Universidad Autónoma de Madrid
Escuela politécnica superior

trabajo de Fin de Grado

DETECCIÓN FORENSE DE ATAQUES USANDO TRAZAS DE RED

Doble grado en Ingeniería Informática y Matemáticas

Miguel Gordo García
Mayo 2015

DETECCIÓN FORENSE DE ATAQUES USANDO TRAZAS DE RED

AUTOR: Miguel Gordo García
TUTOR: Jorge E. López de Vergara Méndez

Dpto. de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo 2015

Resumen

Dentro del desarrollo que ha experimentado la informática en los últimos treinta años ha sido probablemente el campo de la seguridad uno de los que más se ha diversificado debido a las constantes amenazas y la evolución de las mismas. Este campo abarca ahora temas tan dispares como la criptografía y la seguridad de protocolos, así como la defensa frente a ataques a través de Internet o con la reconstrucción de un ataque con técnicas forenses.

El objetivo de este documento, presentado como trabajo de fin de grado, es realizar un análisis de seguridad a posteriori de una red basándonos en flujos o trazas de red. En concreto, este análisis consistirá en la detección de ataques que alteren el volumen de tráfico de la red, tales como DDoS, envío de spam, o escaneos de puerto. Estos flujos resumen transmisiones a través de TCP/IP y UDP que tienen lugar a lo largo de un intervalo de tiempo, siguiendo el estándar de facto implementado por Cisco, Netflowv5.

Para ello, se ha estudiado el estado del arte en el campo de detección de ataques tanto a posteriori como en tiempo real, así como herramientas existentes ya desarrolladas. Se presenta también una herramienta que permite analizar el tráfico en un nodo obtenido con *flow-tools* y presentar la serie temporal en función de los tama nos de los flujos.

Por último se ha realizado un estudio y validacíon de los resultados alcanzados por la herramienta sobre datos obtenidos de varios nodos de la red española *RedIRIS* a lo largo del año 2013.

Palabras Clave

Detección de ataques, RedIris, flujos de red, análisis forense, Netflow, Media Móvil, Holt-Winters, Jacobson

Abstract

In the context of the great developments achieved in Computer Science in the past thirty years, Computer Security is probably the area that has diversified more among its peers, mostly due to the constant evolution of the threats it faces. This area now encompasses fields as different as Cryptography, Protocol Security, defense against network attacks as well as their forensic analysis

The purpose of this document, presented as final degree project, is to make a network security analysis based in flows, also known as network traces. More precisely, this forensic analysis consists in the detection of attacks that produce anomalies in the traffic volume of the network, such as DDoS, spam attacks, or port scanning. This flows, or network traces, summarize TCP/IP and UDP transmissions that take place in a certain period of time, following the de facto standard implemented by Cisco, Netflowv5.

To that end we have studied the state of the art in attack detection, be it forensic or real-time, as well as tools previously developed. We also present a program that allows the network administrator analyze past flow records obtained with *flow-tools*, and presents a temporal series depending on the size of the flows.

Lastly, we hace studied and validated the results obtained by the program by analizing a dataset obtained from several exporters of the Spanish network RedIRIS during the year 2013.

Key words

Ataack detection, RedIris, Network flows, forensic analysis, Netflow, Moving Average, Holt-Winters, Jacobson

Agradecimientos

En primer lugar quiero agradecerle a Jorge su inestimable ayuda y paciencia a lo largo de este proyecto. Él ha estado siempre ahí aportando nuevos puntos de vista, feedback, correcciones y buenos consejos, así como su constante motivación. Sin él no habría tenido ni por dónde empezar. Gracias por introducirme a un campo de la informática que siempre me ha intrigado.

También me gustaría extender mis agradecimientos a Antonio Cuevas por su ayuda, así como al Institute of Computer Science of Masaryk University, por dejarme probar y proveer feedback para su herramienta *Time Series Solver*. No se me olvida agradecer el apoyo de David Muelas, por su inestimable ayuda al recomendarme varios artículos así como al estar estudiando las distribuciones de tráfico de los exportadores.

A mi familia, a quienes no podré pagar nunca todo el bien que me han hecho y que me han transmitido lo más importante: la fe.

Por último quiero dar las gracias a todas las personas que me han acompañado a lo largo de toda la carrera. Ha sido un viaje lleno de buenos amigos y grandes personas. Nombrarlas a todas me resultaría imposible: gracias por vuestra amistad y vuestro apoyo incondicional.

Índice general

Índice de figuras

Índice de tablas

Glosario de acrónimos

- **Flow**: Flujo o traza de red
- **DDoS**: Distributed Denial of Service
- **MAD**: Mean Absolute Deviation
- **Netflow**: Protocolo de red desarrollado por *Cisco Systems* para monitorizar el tráfico de red
- **IDS**: Intrusion Detection System
- **RTO**: Retransmision Timeout
- **TSS**: Time Series Solver
- **CCDF**: Complementary Cumulative Distribution Function

1 Introducción

1.1. Motivación del proyecto

Durante los últimos 30 años hemos visto en la comunidad científica un interés siempre creciente por la seguridad informática. Este interés radica en la contínua expansión de la red y en la creciente cantidad de servicios y utilidades globalmente conectados. Gracias a las estadísticas de la unión Internacional de Telecomunicaciones [1] podemos ver que en España el porcentaje de personas con acceso a Internet ha aumentado desde el 39.93 % en 2003 al 71.57 % en 2013, con otros países experimentando crecimientos similares o mayores. La infraestructura de Internet se ha adaptado para poder servir la mayor cantidad de usuarios y también el aumento en ancho de banda de las conexiones, que hoy en día se sitúan, en los nodos de tier 2 y tier 1, entre 1-10 Gps.

Para monitorizar el tráfico en los nodos de internet han sido necesarios nuevos métodos y nuevos enfoques, especialmente en empresas o universidades, donde el volumen de tráfico en horas puntas del día se alcanzan valores del orden de Gbps. Por otro lado, el número de ataques también ha aumentado, en gran parte al haberse convertido en algo cada vez más rentable económicamente. Expertos calculan que aproximadamente el 90 % de los emails mandados a nivel mundial son ataques de spam [2].

Los ataques informáticos más frecuentes hoy en día alteran el volumen de tráfico usual de un nodo de una manera u otra en el momento en que se producen, y este hecho se puede utilizar en nuestro favor para poder detectar el instante en que comienza un ataque. En ocasiones se altera el número de flujos de manera abrupta (Blaster worm), otras se altera otros parámetros como puede ser el throughput de la red. En cualquier caso siempre dependen del nivel de agregación y de las métricas observadas. Las alteraciones o anomalías pueden ser muy diferentes en su naturaleza: podría tratarse de envío masivo de spam, ataques de diccionario, escaneos de puertos o ataques de denegación de servicio (DDoS).

Por eso es lícito preguntarse y estudiar como se pueden detectar estos ataques desde un sensor que recoge datos de tráfico en un nodo. De esta manera surge el propósito de este proyecto: estudiar a posteriori los datos obtenidos por un sensor para detectar anomalías en el tráfico, es decir, alteraciones en el volumen de tráfico que puedan ser resultado de un ataque.

1.2. Objetivos y enfoque

Los objetivos del proyecto son tres:

- Estudiar el estado actual de ataques que alteren el volumen de tráfico en un nodo: ataques como DDoS, escaneos de puerto, ataques de diccionario y ataques de spam, así como sus métodos de detección, prevención y posterior análisis.
- Diseñar e implementar una herramienta que permita realizar un análisis forense para detectar ataques de red mediante trazas de red (flujos), obtenidos con el paquete *flow-tools*.
- Validar dicha herramienta y estudiar sus resultados ejecutándola sobre datos obtenidos a lo largo del año 2013 de cinco exportadores de la red española de RedIRIS, a los que nos referiremos como exportadores A, B, C, D y E, así como proporcionar indicaciones sobre desarrollos futuros o mejoras de dicha herramienta.

1.3. Trabajo realizado

El trabajo ha consistido en la realización de las siguientes tareas:

- Estudio del estado del arte: Las siguientes tareas se han realizado para el estudio del estado del arte:
  - Lectura del libro *Network Security through Data Analysis* [3] durante el verano de 2014 como introducción al tema tratado.
  - Lectura de varios artículos científicos a fin de entender el estado del arte en cuanto a seguridad mediante analisis de flujos.
  - Práctica de monitorización de tráfico local con *softflowd*
  - Análisis de la herramienta *Time Series Solver* de la Universidad de Masaryk, y estudio de resultados obtenidos tanto con la versión de analisis en tiempo real como plugin de *softflowd* sobre el tráfico local y sobre datos de RedIRIS
- Implementación y validación de un programa en Python que permita visualizar el tráfico a lo largo de un período de tiempo, separando el tráfico en función a su tamaño, y presentar alarmas al usuario cuando distintos métodos de detección basados en filtros de series temporales hayan detectado una anomalía. El trabajo se puede desglosar como:
  - Estudio de los datos y las distribuciones del tráfico en cinco exportadores de RedIRIS
  - Estudio de los filtros y técnicas de detección. Elección de tres métodos de detección.
  - Implementación del programa en Python
- Estudio de los resultados de la herramienta sobre datos de RedIRIS del año 2013 en los cinco nodos antes mencionados

1.4. Estructura del Documento

El documento presenta la siguiente estructura:

- En la sección 2 presenta el estado del arte en cuanto a detección de ataques que alteran el tráfico en la red. Se presentará la herramienta Time Series Solver, así como una introducción al concepto de *flow* (o flujo), así como su formato siguiendo el estándar de facto Netflowv5, con el que se trabajará en el resto del proyecto.
- La sección 3 presentará los objetivos de la herramienta desarrollada para detectar los ataques de manera forense. Se justificarán las decisiones tomadas durante su desarrollo y se dará explicación a los métodos de detección automática de ataques.
- La sección 4 presentará un pequeño estudio de eficiencia, así como un estudio de las distribuciones de tráfico en los exportadores elegidos de RedIRIS.
- La sección 5 presenta las pruebas realizadas sobre datos preliminares para configurar los parámetros de los distintos métodos de detección.
- La sección 6 presenta las los resultados de ejecutar el programa frente a los datos de los cinco exportadores de RedIRIS a lo largo del año 2013. Se presenta asimismo un breve estudio en profundidad de tres anomalías encontradas y comparativas de los métodos de detección empleados.
- La sección 7 presenta las conclusiones del trabajo, así como posibles mejoras del programa y sugerencias para futuras iteraciones

2

Estado de arte

En este capítulo se da una introducción al estándar Netflowv5, del que se hará uso extensivo en el resto del trabajo. A continuación se presenta un pequeño estudio del estado del arte en detección de anomalías en distintas vertientes del estudio de redes.
En último lugar se presenta la herramienta *Time Series Solver*, desarrollada por el Masaryk Institute of Technology, cuyo estudio ha sido motivante para la realización de este trabajo.

2.1. Introducción a trazas de red

A la hora de monitorizar el tráfico en un nodo es necesario especificar a qué nos estamos refiriendo. Tradicionalmente, cuando surgieron los primeros NIDS (*Network Intrusion Detection Systems*) la técnica que se utilizaba es la de captura de paquetes para inspeccionar la carga[4] de cada uno de ellos.

En 1999 *Cisco Systems* publicó la primera versión de su popular estándar de sumarización Netflow. En él se introdujo el concepto de *flow* (en adelante, flujo), que es en esencia una aproximación de una sesión TCP. Cuando un nodo que está configurado con Netflow recibe paquetes, añade los datos relevantes al resumen de esa conexión, por ejemplo, qué banderas se han visto, cuántos bytes se han transmitido... En esencia, un flujo o traza de red es un resumen de un conjunto de paquetes que involucran a dos direcciones IP en un intervalo de tiempo. Normalmente se identifican por la quíntupla (IP origen, IP destino, Puerto origen, Puerto destino y Protocolo)

Los flujos se generan de dos maneras, bien por un router o switch, o por una aplicación que convierte los paquetes en registros de Netflow. Existen ventajas e inconvenientes para hacerlo de una manera o de otra:

- La generación de flujos en un router puede causar problemas de rendimiento. Para solucionar se pueden tomar varias medidas: reducir la prioridad del proceso (perdiendo paquetes), o realizar un muestreo del tráfico[3]
- Por otro lado, la generación de flujos mediante una aplicación tiene el inconveniente que pierde el punto de vista de un router, ya que normalmente sólo monitorizan una interfaz

de red. Por otro lado tienen la ventaja de poder utilizar mayor nivel de procesamiento y producir mejores resultados.

La versión más importante de Netflow ha sido Netflowv5, pero ha quedado obsoleta al estar diseñada únicamente para IPv4. La más moderna v9 permite monitorizar IPv6, elegir que campos se desean muestrear y otras utilidades para administradores de red. Posteriormente se ha estandarizado IPFIX, que es similar en funcionalidad a Netflowv9, pero estándar del IETF.

2.1.1. El estándar Netflowv5

En esta sección presentaremos el estándar Netflowv5 así como la estructura y los componentes necesarios para la captura de flujos para su posterior análisis.
La definición de flujo que vamos a tomar es la dada por IPFIX en la correspondiente especificación del IETF [5]:

> "Un flujo se define como un conjunto de paquetes IP que pasan a través de un punto de observación en la red durante un determinado período de tiempo. Todos los paquetes que pertenecen a un determinado flujo tienen un conjunto de propiedades en común"

Estas propiedades comunes (*common properties*) se llaman *flow-keys*, y son frecuentemente el conjunto de campos (`ip_src, ip_dst, port_src, port_dst, proto`). Los campos de un flujo siguiendo el estándar de Netflowv5 se muestran en la tabla 2.1:

Bytes	Name	Description
0-3	`scraddr`	IP del origen
4-7	`dstaddr`	IP del receptor
8-11	`nexthop`	IP del siguiente salto
12-13	`input`	Índice SNMP de la interfaz de entrada
14-15	`output`	Índice SNMP de la interfaz de salida
16-19	`packets`	Número total de paquetes
20-23	`d0ctets`	Número de bytes de la capa 3
24-27	`first`	sysuptime del primer paquete recibido
28-31	`last`	sysuptime del último paquete recibido
32-33	`srcport`	Puerto TCP/UDP de origen
34-35	`dstport`	Puerto TCP/UDP de destino
36	`pad1`	Padding
37	`tcp_flags`	OR de todas las banderas TCP del flow
38	`proto`	Protocolo IP
39	`tos`	Tipo de servicio IP
40-41	`src_as`	ASN del origen
42-43	`dst_as`	ASN del destino
44	`src_mask`	Mascara del prefijo del origen
45	`dst_mask`	Mascara del prefijo del destino
46-47	`pad2`	Padding bytes

Tabla 2.1: Campos para un flow en Netflow v5

Es importante notar que flujo no equivale a *conexión*, puesto que por ejemplo se generaría un flujo cuando hubiera un intercambio de paquetes sobre UDP. Un flujo no tiene restricciones

de tamaño tampoco, pues aunque haya un único paquete intercambiado se generará un flow.

La generación de flujos sigue dos pasos: *flow exporting* y *flow collection*, realizados por dos agentes: *flow exporter* y *flow collector* (exportador y colector). La figura 2.1 muestra la arquitectura del exportador y colector

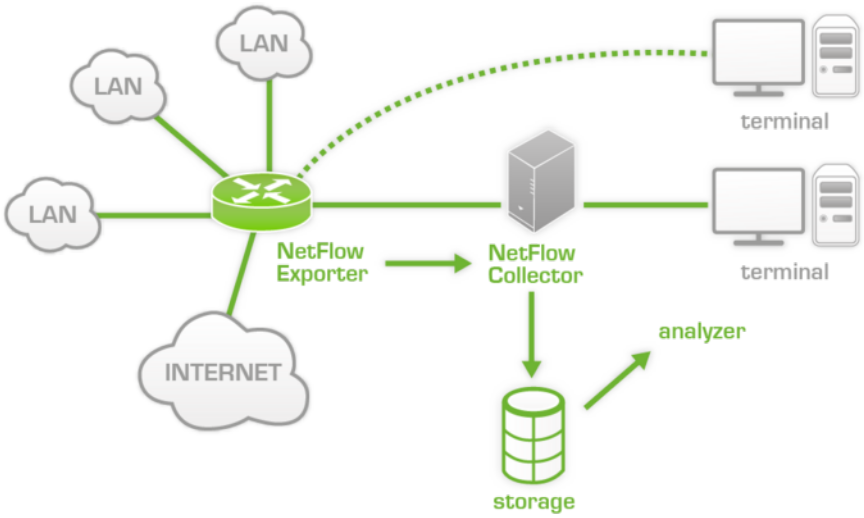


Figura 2.1: Arquitectura del exportador y colector de flujos[1]

El exportador es el nodo de la red que observa el tráfico y crea los flujos a partir de las observaciones. A la llegada de una paquete a la interfaz monitorizada, se guarda su marca de tiempo (*timestamp*). Acto seguido el paquete puede ser descartado por un muestreo (*sampling*), en otro caso es filtrado (*filtered*) como explicaremos más adelante. Cada paquete que llega provoca una llamada a refrescar la cache de flujos, si no existe uno con los mismos datos que la cabecera actual se crea un nuevo flujo y se guarda en la caché, de lo contrario se añade al flujo correspondiente. Cuando el flujo expira se envía al colector para su posterior almacenamiento. Un flujo se considera que ha expirado cuando se cumple una de las siguientes condiciones:

- El flujo ha estado inactivo durante más tiempo que un timeout prefijado de antemano. El tiempo por defecto de Netflowv5 es de 15 segundos.
- El flujo lleva en activo más tiempo que el máximo tiempo de vida permitido. En este caso, el flow se archiva y se manda al colector, y si es necesario se crea un nuevo flow que contiúe el registro. Por defecto, el máximo tiempo de vida de un flujo está configurado en 30 minutos.
- Las banderas `FIN` o `RST` se han visto en uno de los paquetes
- La caché de flujos se llena. En este caso, los flujos elegidos por un algoritmo LRU (*Last Recently Used*, usado más recientemente) se consideran expirados y se exportan al colector

El colector, por su parte, agrupa los flujos que van llegando en un formato adecuado para su futuro análisis.

Si un exportador resulta ser un nodo con un tráfico elevado se dará facilmente la situación de que en un momento dado haya millones de paquetes pasando y la tabla de flujos sea del orden

[1]Pandora FMNS `http://wiki.pandorafms.com`

de cientos de miles de flujos. Como a cada llegada es necesario refrescar la caché de flujos esto puede causar problemas de rendimiento, por eso se utilizan normalmente técnicas de muestreo o filtrado. No existe un estándar o unas técnicas prefijadas, aunque algunos grupos [6] están trabajando para proponer un estándar en este área. Se distinguen dos categorías para los tipos de muestreo:

- **Muestreo de paquetes**: Podemos encontrar técnicas deterministas, como por ejemplo coger un paquete cada t segundos, (*Time Driven*), o coger un paquete cada N paquetes recibidos (*Event Driven*). También se pueden muestrear los paquetes de manera aleatoria, en este caso podemos encontrar muestreo de acuerdo a una función de distribución prefijada (que asigne a cada flujo una probabilidad) o, por ejemplo, elegir n paquetes a muestrear de N paquetes (técnica *n-de-N*)
- **Muestreo de flujos**: Los métodos son similares pero en vez de aplicarse sobre paquetes se aplican sobre los flujos. El más extendido es la técnica *Sample and hold*: esta consiste en que cada vez que recibimos un paquete que no pertenece a ningún flujo en la caché se guarda con probabilidad p, y subsiguientemente todos los paquetes de dicho flujo se añaden al mismo. De esta manera nos aseguramos de que, de cada flujo, se tienen todos los paquetes, al contrario que con muestreo de paquetes donde un flujo puede haber perdido paquetes debido al muestreo.

Con esto queda introducido los conceptos relativos a el estándar Netflowv5.

2.2. Motivación para el estudio de flujos

Es un hecho que el número de ataques en el mundo se ha disparado en los últimos años. El CERT Coordination Center [7] es una organización que publica anualmente una lista de vulnerabilidades conocidas así como una calificación de su gravedad, a fin de ayudar e investigar el panorama de la ciberseguridad mundial. Sin embargo, desde el anõ 2003 dejaron de contabilizar el número de ataques realizados anualmente al seguir un crecimiento exponencial [7]:

> "Debido al uso generalizado de herramientas para realizar ataques automatizados, los ataques contra sistemas conectados a Internet se han vuelto tan frecuentes que contabilizar el número de incidentes que se reportan ofrecen poca información con respecto a evaluar el alcance e impacto de los ataques. Por ello dejamos de proporcionar esta estadística a finales de 2003 "

Por otro lado, también es necesario tener en cuenta el aumento del tráfico de Internet, con enlaces que soportan varios Gbps. Esto ha causado que el enfoque tradicional de análisis de paquetes no sea el más adecuado a la hora de monitorizar el tráfico de la red, pues no es posible efectuar análisis de paquetes a esa velocidad. NIDS como Bro[4] y Snort [8], muestran excesiva carga al enfrentarlos contra volumenes de tráfico actuales.
Esto no implica que el análisis por paquetes sea una técnica obsoleta, sino que ha de entenderse como complementaria: el análisis de flujos puede detectar anomalías que luego pueden ser analizadas con más detalle a nivel de paquetes.

Al ser los flujos un resumen aproximado de transmisiones entre dos direcciones y dos puertos, se le puede achacar alguna desventaja como por ejemplo que contiene poca información. Sería imposible, por ejemplo, tratar de encontrar patrones en *payloads* de diferentes paquetes. La respuesta a esto es doble: en primer lugar depende de los objetivos que se tengan. En segundo lugar,

es cierto que en los flujos falte información porque se limita a ver las interacciones entre hosts. Sin embargo esta información suele ser suficiente para detectar la mayoría de ataques puesto que se pueden encontrar patrones basados precisamente en interacciones entre hosts (alto tráfico entre un servidor de correo y un host exterior, repetidos intentos de conexión en puertos poco usuales para un servidor web, etc.). Por último, hoy en día la mayoría de las comunicaciones entre hosts son cifradas, por lo que puede ser demasiado costoso e ineficiente estudiar el contenido de los paquetes.

En un mundo ideal como en el descrito en [9] el análisis de paquetes siempre será superior al análisis de flujos en cuanto a precisión, pero en la realidad la capacidad de procesamiento necesaria se queda corta para poder ser considerada una opción viable. Igualmente, a nivel de análisis forense, es muy costoso en espacio guardar todos los paquetes, siendo mucho más eficaz guardar los registros de Netflow como un resumen del tráfico observado. Por ello el análisis de flujos se mantiene y seguirá siendo una técnica clave para la seguridad de redes en los años venideros.

2.3. Flujos como series temporales: Detección de anomalías

De acuerdo con [10] una la detección de intrusiones (Intrusion Detection) se define como:

> "La detección de intrusiones es el proceso que consiste en identificar y responder a las actividades maliciosas que tienen por objetivo recursos de computación o de la red"

Una intrusión o ataque es, por tanto, un conjunto de acciones que tienen como objetivo que el atacante adquiera control o información del sistema atacado. Desde que el campo de la detección de intrusiones adquiriera importancia en la década de los 80, muchos enfoques diferentes se han propuesto para su estudio, pero las más usada hoy en día son dos: detección de intrusiones mediante anomalías (anomaly-based IDS), y mediante uso indebido (misuse-based IDS).

- **Basados en uso indebido**: Estos sistemas de detección son basados en conocimiento, de manera que tratan de detectar patrones existentes en su base de conocimiento y detectar una alarma cuando ocurren de nuevo. Efectivamente, un sistema de detección basado en conocimiento no puede detectar ataques nuevos no existentes en la base de conocimiento. Sin embargo su fuerte radica en ser altamente preciso, lo cual quiere decir que el número de falsos positivos es bajo. También son conocidos como IDS basados en firmas o patrones de ataques.
- **Basados en anomalías**: Estos sistemas se basan en analizar el comportamiento actual del tráfico frente a su comportamiento esperado, que se basa en observaciones del comportamiento normal del nodo, de manera que busca desviaciones o comportamientos anómalos. La ventaja frente al los IDS basados en uso indebido es que a priori permite detectar ataques nuevos mientras estos alteren el comportamiento habitual del nodo.

En estos sistemas es inevitable encontrar el problema de optimización entre falsos positivos y falsos negativos, puesto que levantar excesivas alarmas causa que el administrador pertinente se sature con alarmas que podrían ser en su mayoría falsas, mientras que el enfoque contrario podría llevar a ser laxo y no detectar ataques que "no hagan mucho ruido".

2.3.1. Ataques más comunes

En esta sección introduciremos una clasificación de ataques comúnmente aceptada hoy en día para entender a qué nos enfrentamos[11]. Se suele dividir en las siguientes categorías[12]:

- **Ataques físicos**: consisten en dañar el hardware.
- **Buffer overflows**: Ataques que ganan el control o destruyen un proceso haciendo overflow de alguno de sus buffers.
- **Ataques de contraseña**: Ataques que intentan robar identificaciones de usuarios de un sistema. Un ataque de diccionario entraría en esta categoría.
- **Ataques de denegación de servicio (distribuidos)**: Ataques en los que el atacante satura un servidor con el fin de que los usuarios no puedan acceder a él o se incremente el tiempo de respuesta.
- **Obtención de información**: Estos ataques tratan de obtener información de un sistema, para posiblemente usarlo en un ataque futuro. Los escaneos de puertos y sniffers entrarían en esta categoría.
- **Troyanos**: Un programa que se hace pasar por una aplicación que no es en realidad, puesto que de espaldas realiza acciones no deseadas por el usuario.
- **Worms (gusanos)**: Programas que se auto propagan a través de una red. Esta autopropagación es la que la diferencia de los virus.
- **Virus**: Un programa que sólo se replica en el host que ha infectado. Su velocidad de propagación no se puede comparar a la de un gusano.

Como [11] hace notar, estos ataques no son mutuamente exclusivos, puesto que existen ataques de denegación de servicio y guanos que se sirven de buffer overflows para cumplir su cometido.

Mención especial merece la aparición de Botnets, que son la infraestructura necesaria para lanzar un ataques distribuidos. Esta red de bots infectan numerosos hosts sin que estos se den cuenta, y obedecen las órdenes de uno o más *bot-masters*, que coordina la botnet para lanzar ataques como DDoS o envío masivo de spam.
Los ataques que se pueden detectar con detección mediante flujos son un subconjunto de los arriba presentados, al sólo poder acceder a las cabeceras de los paquetes. Estos son: DDoS, Escaneos, Gusanos y Botnets. De este subconjunto, los más modernos y aún en un estado precario son el estudio de detección contra botnets y ataques distribuidos. Los otros, en mayor o menor medida, se han estudiado y ya presentan un cierto desarrollo. En las dos siguientes subsecciones presentaremos el estado del arte en cuanto a detección de botnets y ataques distribuidos usando detección con flujos.

2.3.1.1. Denegación de Servicio

En esta sección presentaremos varios trabajos punteros en el área de ataques DDoS.
Los ataques de denegación de servicio llevan tiempo siendo estudiados desde la perspectiva del análisis de flujos, puesto que normalmente son visibles a este nivel al producir variaciones en el volumen de tráfico durante un intervalo de tiempo. Con este enfoque nos dejamos fuera los ataques de denegación de servicio *semánticos*, que buscan obtener los mismos fines pero con los contenidos del payload. A nivel de flujos, no se pueden analizar los contenidos de los paquetes.

Un ejemplo de este tipo de ataques sería el famoso *Ping of Death.*

Los primeros ejemplos de trabajo son de *Li* [13] y *Gao* [14]. Su enfoque consiste en estudiar los flujos con unas estructuras de datos especiales llamadas *sketches*. Estos *Sketches* son en esencia tablas hash que guardan la información y sirven para contar el número de ocurrencias de un evento. Los *sketches* con los que trabajan los autores son de dos dimensiones. Estas estructuras permiten caracterizar como varía el tráfico en un período de tiempo dado simplemente estudiando si un determinado flujo está presente en durante dicho intervalo de tiempo. En base a esto se realizan predicciones y cuando el tráfico actual difiere de la media esperada se levanta una anomalía. Por ejemplo, para el caso de detección de SYN flooding, el *sketch* guarda para cada intervalo de tiempo y tupla (`dest_ip`, `src_ip`), la diferencia entre el número de el número de SYN y el número de SYN/ACK. Cuando el valor actual se dispara del esperado, se activa la alarma que indica que un ataque está ocurriendo.

Otro enfoque propuesto por Zhao en [15] sigue un enfoque similar. En este caso, utilizan un algoritmo para identificar direcciones IP que tengan un número excesivo de conexiones abiertas en un momento dado. Puede ser que una dirección tenga un gran número de conexiones entrantes (*large fan-in*) o gran número de conexiones salientes (*large fan-out*). En el primer caso nos encontramos ante una búsqueda de víctimas para un ataque DOS, mientras que en el segundo se trata de un patrón típico del escaneo de puertos.
El método empleado utiliza de nuevo hash tables, como en el enfoque anterior, para moder medir el ratio *fan-in/fan-out* de un host determinado y poder determinar, de nuevo, si está siendo víctima de alguno de los dos ataques anteriores.

Por último, el enfoque de Kim [16] es clasificar los distintos tipos de ataques de denegación de servicio en función de sus características que presentan a nivel de flujos. Así, mirando estadísticas como el número de flujos y paquetes, el tamaño de los flujos y los paquetes, ancho de banda consumido, media de paquetes por flujo y otros más permite detectar patrones en su base de conocimiento de ataques en tiempo real. Así, por ejemplo, en un SYN flooding attack, se dispara el número de flujos pero el número de paquetes no, son flujos y paquetes pequeños y sin restricciones de ancho de banda. En su trabajo se establecen claramente los patrones de cada ataque y se formalizan en funciones de detección que determinan la probabilidad de que un patrón de tráfico sea en realidad un ataque.

2.3.1.2. Botnets

En esta sección se presentarán varios trabajos punteros en detección de botnets.
Como se explicó en la sección 2.3.1, una Botnet es un conjunto de hosts infectados sin conocimiento de sus dueños que obedecen las órdenes de uno o más bot-masters. Esta infraestructura permite toda clase de ataques distribuidos. También es cierto que eliminado el botmaster o botmasters correspondientes la red queda inutilizada, pero en general el campo de defensa contra botnets aún es muy precario y requiere más investigación.[17]

Varias redes de botnets solían utilizar IRC para las comunicaciones entre el botmaster y la red. De esta manera, Karasaridis [18] propone un método para detectar botnets que no depende de los puertos IRC. Su solución consiste en dos partes: primero presentan un proceso de varias etapas para detectar a los controladores de una botnet. De esta manera, a partir de logs, viruses e informes de spam se generan conjuntos de flujos sospechosos candidatos a pertenecer a una botnet (*candidate controllers conversation*). Estas conversaciones pueden tener lugar entre un

posible servidor (*controller*) que usa IRC u otro protocolo para enmascarar su tráfico. En la segunda etapa la conversación se chequea contra los modelos de flujo.
Una vez que los *controllers* han sido identificados, el siguiente paso es agrupar los bots en grupos de comportamiento (*behavioural groups*), es decir, grupos que presentan los mismos patrones de comportamiento, basado en los puertos utilizados. Sin embargo, los autores hacen constar que, al contrario que en escaneos de puerto u otro ataque al que se pueda responder en tiempo real, las observaciones necesarias para identificar *controllers* y grupos de comportamiento necesitan realizarse sobre un largo período de tiempo.

Un enfoque diferente es la herramienta *Botminer*[19]. Este detector de botnets actúa de dos maneras: la primera, utiliza análisis de flujos para encontrar grupos que presenten los mismos patrones en sus comunicaciones. La segunda, utiliza inspección de paquetes para detectar actividades ilegales en las comunicaciones. Los resultados de ambas etapas se mezclan para crear grupos con la misma actividad maliciosa. Un host que no se comporta como debiera no indica necesariamente la presencia de una botnet, por lo que es necesario realizar correlación cruzada para juntar los resultados de ambas etapas.

En cualquier caso, este campo de seguridad aún está en un estado primitivo y lleva varios años siendo objeto de estudio por parte de la comunidad científica, lo que esperamos que produzca valiosos resultados en el futuro cercano.

2.3.2. Time Series Solver

Un grupo de investigación del Institute of Computer Science of Masaryk University, República Checa, presentó en 2014 su herramienta *Time Series Solver* [20], [21]. TSS es un plugin para *softflowd* [22], un analizador de tráfico que utiliza el paquete de Cisco Netflow. Este plugin, basado en el análisis de flujos permite la visualización en tiempo real del tráfico en un nodo. La visualización presenta el número de flujos en el eje y a lo largo del tiempo, y divide los flujos en varias categorías en función de su tamaño, de modo que es fácil comprobar a simple vista que familia de flujos es responsable de la mayor parte del tráfico en un momento dado. Las familias en las que divide el tráfico son (en bytes): 0-100, 100-200, 200-500, 500-1000, 1000-5000, 5000-10000, 10000-∞. Un ejemplo de visualización lo podemos ver en la figura 2.2. La herramienta permite no sólo la visualización del tráfico en tiempo real sino que también alerta al usuario si el volumen de tráfico (medido en número de flujos activos) varía sustancialmente en un momento dado. Esta variación podría deberse a un ataque que se esté produciendo en ese momento, como podría ser un ataque de denegación de servicio, robo de datos a gran escala, o un ataque de spam.

Para medir esta variación de tráfico el programa utiliza varios métodos de detección por separado. Estos métodos son varios filtros de series temporales, que en esencia procesan el tráfico pasado y realizan una predicción en tiempo real. Si la predicción se sale de las bandas de confianza, crea una alerta y la muestra al usuario. Esta alerta son los círculos rojos que se ven en la figura 2.2.

Los responsables del proyecto en el Institute of Computer Science of Masaryk University nos permitieron probar su herramienta en versión alfa. Asimismo, también nos proveyeron con una versión en terminal del programa que no necesita estar monitorizando en tiempo real una interfaz de red, sino que puede ser ejecutada sobre archivos de tráfico obtenidos con anterioridad.

La herramienta aún está en un estado muy básico, y ambas versiones obtuvieron resultados

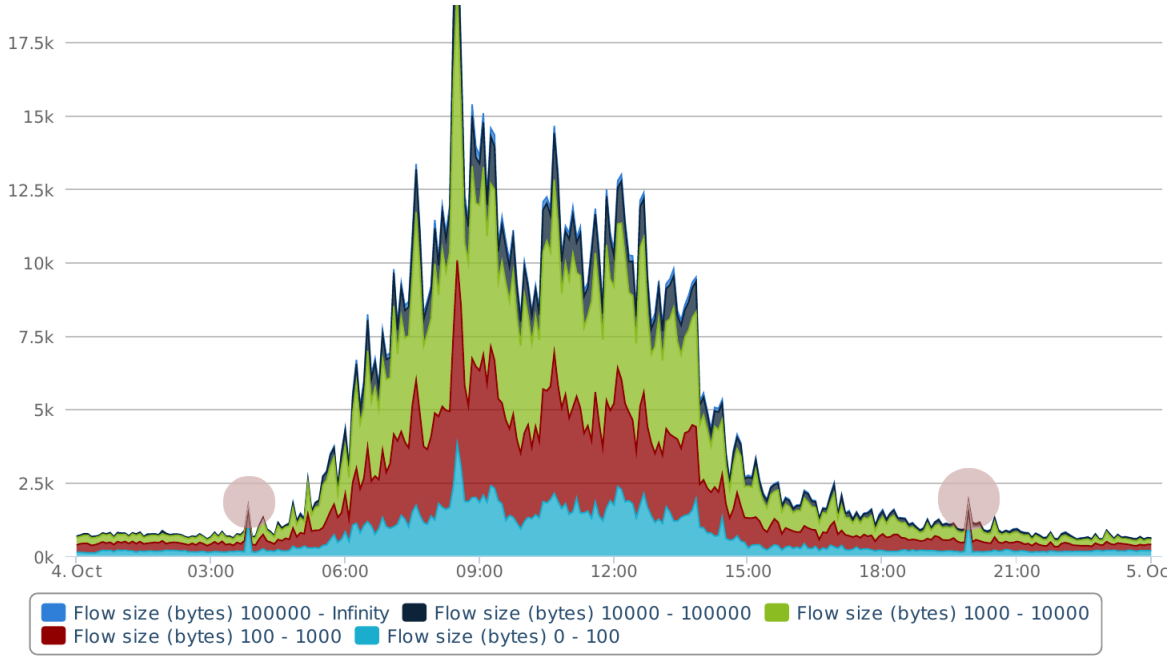


Figura 2.2: Un ejemplo de visualización de la herramienta Time Series Solver [1]

mixtos a la hora de analizar el tráfico, tanto en tiempo real como con archivos antiguos. Asimismo varias decisiones de diseño no tienen un fundamento claro, como por ejemplo cuál es el criterio para la separación del tráfico en las categorías utilizadas. Asimismo una lectura atenta del código fuente revela que los intervalos de confianza utilizados en todos sus métodos de detección vienen dados exclusivamente por múltiplos de la predicción, y no se tiene en cuenta ni la media ni la desviación típica de las observaciones. De esta manera la detección de anomalías queda seriamente perjudicada, pues los límites son arbitrarios e inconsistentes. Sin embargo el concepto es muy interesante y fue el estudio de esta herramienta la que motivó en gran parte el desarrollo finalmente realizado.

2.4. Conclusiones

En este capítulo se ha hecho una presentación del estado del arte en detección de anomalías para redes de alta velocidad, presentando ataques, y escenarios de interés actual. Se ha presentado asimismo el estándar de facto Netflowv5, que es la base para el resto del trabajo desarrollado. Por último, se ha presentado la herramienta *Time Series Solver*, cuyo estudio es la motivación del trabajo actual y ha llevado al desarrollo del trabajo que se sigue en los próximos capítulos.

En concreto, de la herramienta Time Series Solver hemos observado su arbitrariedad a la hora de elegir intervalos de tamaños para los flujos. Esta decisión puede ocultar información que, presentada de otra manera, permitiría a un administrador de red adquirir más información sobre lo que está pasando en la misma. Asimismo, se ha observado que los intervalos de confianza de los métodos de detección están poco fundamentados, lo que inevitablemente se traduce en una peor detección de anomalías. Las mejoras que se proponen en este trabajo se enfocarán por estas dos líneas.

[1] *Time Series Solver* `http://www.muni.cz/ics/services/csirt/tools/tss`

3 Requisitos

En este capítulo se procederá a la definición de objetivos del programa implementado, así como el análisis de requisitos del mismo, tanto funcionales como no funcionales. Entre ellos se encuentran los objetivos relativos a la presentación visual de la herramienta y la detección de anomalías. Se presenta también un pequeño estudio de eficiencia entre varios lenguajes de programación para motivar la elección tomada al respecto.
Se presentan también los métodos de detección de los que se hará uso: Media Móvil, Holt-Winters de segundo orden y método de Jacobson. Se hará la justificación de sus intervalos de confianza, y por último se presentarán dos métodos de detección adicionales que se han estudiado pero que al no adecuarse al caso de estudio no se han implementado.

3.1. Requisitos funcionales

El objetivo de este proyecto es diseñar, implementar y validar un programa que permita el análisis de flujos a posteriori así como detectar anomalías en el volumen de tráfico por flujos y presentarlas visualmente al usuario. Esta presentación visual representará, en el eje x el tiempo, y en el eje y el número de flujos activos. Los métodos de detección de anomalías estarán basados en filtros de series temporales, y la visualización mostrará al usuario el tráfico a lo largo de un día, dividiendo el tráfico en diferentes gráficas en función de los tamaños de los flujos. Los límites de los intervalos vienen dados por los deciles de los tamaños de los flujos del exportador correspondiente. La visualización presentará estas gráficas de forma acumulada, es decir, que el número de nodos activos para un intervalo de tamaño de flujos en un momento dado incluye los de esa familia junto con los flujos activos de familias de tamaños más pequeños.

De esta manera se propone una mejora a la herramienta *Time Series Solver* en dos sentidos: por una lado, se tienen en cuenta las características particulares de la red para su visualización, de manera que no se haga con intervalos de tamaño arbitrarios y prefijados de antemano (como es el caso de *TSS*), sino que los intervalos vengan delimitados por los deciles de la muestra de tamaños de flujos. Por otro lado, los métodos de detección utilizados por la herramienta *Time Series Solver* no tienen en cuenta la media ni la varianza muestral en sus intervalos de confianza, que son los mismos sea cual sea el intervalo, y esta sí.

Concretamente, se implementarán tres métodos de detección: método de Holt-Winters de segundo orden, método de Media Móvil y aproximación de Jacobson, que serán explicados en profundidad en la siguiente sección. Se compararán los tres métodos, y se compararán los resultados obtenidos por cada uno de ellos.

El programa aceptará como entrada un archivo de flujos basado en el estándar Netflowv5 convertido a texto con el comando *flow-print* del paquete *flow-tools*. Así mismo, el usuario podrá cambiar los parámetros asociados a los métodos de detección a fin de hacer la detección más estricta o permisiva según sea necesario.

La validación de resultados se llevará a cabo ejecutando el programa sobre flujos obtenidos de la red española RedIRIS a lo largo del año 2013 para cinco de sus exportadores, que denominaremos A, B, C, D y E. Se presentará por cada nodo el número de anomalías obtenidas a lo largo del año según cada método de detección, y se estudiará algunas de ellas para investigar el comportamiento sospechoso que la ha producido, y de tratarse de un ataque, identificarlo.

Las características de la red española RedIRIS, en particular, hacen que este trabajo adquiera especial relevancia. En cada uno de sus nodos, RedIRIS realiza un muestreo intensivo, como se explicó en la sección 2.1.1. De esta manera será necesario observar si las anomalías se pueden detectar a pesar del muestreo, y si el mismo es tan excesivo como para evitar que se detecte nada, o que no se detecten ciertos tipos de ataques.

3.1.1. Requisitos no funcionales

- El programa será implementado en Python, por lo que será multiplataforma.
- Extensibilidad: Se podrán añadir nuevos métodos de detección al programa.
- Se podrá cambiar la granularidad de la visualización: un punto por minuto, por segundo, por cinco minutos, etc...
- Puesto que los ficheros de flujos son muy grandes y su procesado lleva tiempo, se buscará la mayor eficiencia posible en tiempo a la hora de procesarlo.

3.1.2. Eficiencia

El programa está implementado en Python y por tanto es multiplataforma. No obstante, al inicio del proyecto se realizó un pequeño estudio comparativo entre R, Python y AWK para comparar tiempos de ejecución. En esta sección se expondrán los resultados del estudio. Las pruebas se realizaron en un portátil ACER Aspire 5738G, con procesador Intel Core 2 Duo P8700 (2.53GHz) y 4Gb de RAM, leyendo un archivo de 1.7Gb.

Claramente el mayor coste computacional viene dado por la lectura de fichero. Los ficheros de flujos en formato texto (una vez descomprimidos de su formato original pcap) ocupan un tamaño entre 2-9Gb, luego es necesario una librería que soporte lectura a gran velocidad de ficheros de estos tamaños.
El tiempo de ejecución de R para la lectura de un archivo de 1,7Gb con la mejor librería disponible (ff) resultó ser de 680 segundos, es decir, 10 minutos. Además, a fin de optimizar la lectura la libreriá requiere de información adicional costosa de conseguir, como el número de líneas del

fichero. Por tanto R quedó descartado.

AWK demostró tener el mejor tiempo, empleando tan sólo 1m57s en procesar el fichero. Sin embargo, la carga extra que supone implementar todos los métodos de detección en AWK, unido a tener que utilizar gnu-plot u otra utilidad similar para realizar la visualización acabó siendo el factor determinante a favor de Python, que aunque empleó para la misma prueba un tiempo total de 2m45s, con la inmensa cantidad de librerías disponible permite una mayor facilidad tanto de implementación como de interoperabilidad, sumado a la multiplataforma. El tiempo añadido no es tan grande como para menospreciar las ventajas adicionales que conlleva Python.

3.2. Métodos de detección

En esta sección se presentan en detalle los métodos de detección que usa el programa para detectar anomalías y que se compararan entre sí en la sección de resultados.

3.2.1. Media Móvil

La media móvil es útil para estimar conjuntos de datos estables, sin tendencia ni estacionalidad. No es el caso para una variación del tráfico durante el día, pero sí lo es para cortos períodos de tiempo. Sin embargo, si una anomalía se produce durante un breve espacio de tiempo relativo a la granularidad, e.g. si estamos visualizando con granularidad de un minuto y la anomalía dura 1 minuto o pocos minutos, entonces el método funciona con precisión.

Por el contrario, si la anomalía se mantiene durante un período de tiempo más largo entonces es predecible que este método devuelva muchas anomalías para un supuesto ataque que, simplemente, se extiende más en el tiempo. Incluso es posible que al volver al tráfico normal devuelva anomalías que sean falsos positivos puesto que a la media móvil es lenta a cambios.

Por defecto en el programa la media móvil utiliza los 30 términos anteriores para calcular la media móvil. Es de predecir que este método devolverá un número elevado de falsos positivos comparado con los otros dos métodos de detección.
Por defecto, se considerará que un valor observado es anómalo si dista $(1-0{,}5)*$`valoresperado` o $(1+0{,}7)*$`valoresperado`. Al hacer más restrictivo el límite superior evitamos levantar anomalías por muchas subidas normales del tráfico (por ejemplo, la subida natural que se produce en la primera hora laboral del día).

3.2.2. Método Holt-Winters de segundo orden

El método de Holt-Winters de segundo orden también se conoce como doble alisado exponencial. Este método es útil para estimar series temporales que presentan una media variable a lo largo del tiempo.

Denotamos x_t la observación t-ésima, s_t la aproximación en el momento t, b_t la aproximación de la pendiente en el momento t, y F_{t+m} la predicción del valor esperado en el momento $t+m$. De esta manera, el método de Holt-Winters de segundo orden se define como:

$$s_1 = x_1$$

$$b_1 = x_2 - x_1$$
$$s_t = \alpha x_t + (1-\alpha)(s_{t-1} + b_{t-1})b_t$$
$$b_t = \beta(s_t - s_{t-1}) + (1-\beta)b_{t-1}$$

donde $\alpha, \beta \in [0,1]$. A α le llamaremos factor de alisado de los datos, mientras que a β le llamamos factor de alisado de la media.

La predicción para el momento $t+m$ viene dada por:

$$F_{t+m} = s_t + mb_t$$

Viendo el método queda claro que se trata de un doble alisado exponencial, uno para los datos y otro para el estimador de la media.
La predicción en el momento F_0 no está definida, y no ha de importarnos pues no es necesario predecir para el instante 0.
La ventaja que nos proporciona este método frente a la media móvil es que no se necesitan n observaciones para poder predecir el valor $n+1$, sino que basta una observación para poder empezar a predecir. Además, permite controlar la sensibilidad de los nuevos valores frente a los viejos. Para nuestro caso de estudio preferiremos valores de α bajos para dar más peso a las observaciones pasadas.

Las bandas de confianza de predicción para el método Holt Winters de segundo orden utilizan la desviación media absoluta (Mean Absolute Deviation). Esta se define como:

$$\text{MAD}(0) = \frac{1}{n}\sum_{t=1}^{n}|s_t - x_t|$$

$$\text{MAD}(t) = (1-\delta) * \text{MAD}(t-1) + \delta * |s_t - x_t|$$

donde n es el número de términos que elijamos para la inicialización, y $\delta \in [0,1]$ es el parámetro que permite pesar diferentemente los nuevos valores frente a las nuevas observaciones de las desviaciones.
De esta manera, los intervalos de confianza vienen dados por:

$$\text{Intervalo\_de\_confianza}(t) = s_t \pm \sigma\text{MAD}(t)$$

Donde $\sigma \in \mathbb{R}$ es un parámetro que permite ajustar el ancho del intervalo de confianza.
La elección de un intervalo de confianza correcto es crucial para la efectividad de estos métodos de detección. Este método refina en este sentido la media móvil, puesto que tiene en cuenta las observaciones pasadas y es "lógicamente"más robusto: para estudiar si un valor es anómalo sería lógico tener en cuenta un estimador de la desviación típica (sin realizar la raíz cuadrada, pues así es menos costoso de computar), en vez de simplemente tomar valores "fijos". Esta búsqueda de un mejor intervalo de confianza es la motivación de la inclusión en el programa diseñado de una variante del algoritmo de Jacobson para el cálculo del RTO en TCP. En la sección de diseño se harán justificaciones de los valores que se han dado a estos parámetros de cara a su validación con los datos de RedIRIS.

3.2.3. Método de Jacobson

El algoritmo de Van Jacobson [23] para la retransmisión de paquetes usando TCP también puede aplicarse para este entorno. La ventaja de este método de detección frente a los otros es

que las bandas de confianza tienen en cuenta no sólo la estimación de la desviación típica, sino también la varianza. Este método responde mejor a la fluctuaciones grandes para la retransmisión en TCP, evitando retransmisiones innecesarias.
Cabe esperar que la elección de parámetros utilizada por Van Jacobson no aplique para este entorno, pero no dista mucho de las recomendaciones originales. La justificación de los parámetros del método se hará en la sección de diseño.

El algoritmo de Jacobson se especifica de la siguiente manera:

$$\text{ERR} = M - A$$

$$A = A + g * \text{ERR}$$

$$D = D + h(|\text{ERR}| - D)$$

$$\text{RTO} = A + 4D$$

donde ERR mide el error entre la medición M y el estimador de la media A (inicialmente, la media muestral hasta un valor prefijado n), D es el estimador de la desviación media absoluta (de nuevo hasta un valor prefijado n), y RTO es el Retransmission Timeout. El $\pm$RTO en nuestro caso se convertirán en los límites de las bandas superior e inferiormente. Sin embargo, el factor 4 que multiplica a D también será modificado acordemente.

Los parámetros g y h son constantes elegidas para dar mayor o menor peso a la nueva estimación de la media o de la desviación media absoluta respectivamente. La recomendación de Jacobson para estos parámetros es $g = \frac{1}{8}$ y $h = 0{,}25$ respectivamente, pero veremos en la sección de diseño los cambios necesarios para ajustar los parámetros a nuestro caso de estudio.

3.2.4. Métodos estudiados no implementados en el programa final

Durante el transcurso del trabajo, varios métodos de deteccíon adicionales a los ya presentados fueron evaluados para su uso. Sin embargo, por unas razones o por otras los métodos resultaron no ser efectivos, o bien requieren suposiciones no presentes en nuestro caso. Algunos de ellos pueden resultar de interés como futuras mejoras del proyecto, como se indicará en la sección 7

3.2.4.1. Metodo de Holt-Winters de tercer orden

El método de Holt-Winters de tercer orden es en esencia un triple alisado exponencial[24]. Es útil cuando la muestra observada se le supone media y además una componente estacional.

Este método será útil para una variante del programa que presente gráficas temporales para una semana entera, en lugar de para un día como está implementado actualmente. Esto es así porque entre días laborales sí que hay presente una componente de estacionalidad: los patrones de comportamiento del tráfico en la red se repiten durante los días laborables (subida del tráfico a primera hora, bajada durante las horas de comer, repunte, decrecimiento hasta la hora de salida). Una futura mejora del programa que permitiera la visualización del tráfico por semanas debería implementar este método al estar específicamente adaptado para el problema tratado.

3.2.4.2. Método de Hodrick-Prescott

El método de Hodrick-Prescott es un filtro de series temporales usado especialmente en macroeconomía para eliminar la componente cíclica de los datos de una serie temporal dada. Concretamente, el método trata de encontrar la minimización de

$$\min_{\tau} \left(\sum_{t=1}^{T} (y_t - \tau_t)^2 + \lambda \sum_{t=2}^{T-1} [(\tau_{t+1} - \tau_t) - (\tau_t - \tau_{t-1})]^2 \right)$$

suponiendo que la serie temporal y_t puede escribirse como $y_t = \tau_t + c_t + \epsilon_t$, donde τ es la componente de la media, c_t es la componente cíclica y ϵ_t es el error en el instante t. El término λ penaliza las variaciones de crecimiento de la componente de la media, y es mayor la penalización cuanto mayor es el tamaño de λ. En función del tamaño de la muestra (e.g. si son datos diarios, semanales, mensuales..) se recomiendan diferentes valores para λ

Sin embargo la asunción de que nuestra serie temporal se puede escribir así no podemos garantizar que se cumple. El método también requiere garantizar que los datos existen en una media I(2), si se produce una subida repentina de los datos no se puede garantizar que el método funcione porque falla al estimar la componente cíclica. Esto se da en nuestros datos (por ejemplo, cuando en un día empieza la jornada laboral).

Este método se implementó en una versión inicial del programa pero resultó ser inadecuado por las razones expuestas arriba. Además, el hecho de que la minimización utilice términos posteriores al momento actual hace que "prediga", en cierto sentido, las anomalías. Por todo esto su utilización fue descartada.

3.2.4.3. Filtro de Kalman

El filtro de Kalman opera recursivamente en flujos de datos con ruido para producir estimaciones de diversas variables desconocidas, y éstas suelen ser más precisas que las basadas en una única observación.
Este método no se puede aplicar puesto que requiere

- Conocimiento de antemano de cómo funciona el sistema a modelar, es decir, si sigue una ecuación o sistema de ecuaciones
- Una estimación del error a priori
- Suponer que el ruido está normalmente distribuido

En nuestro caso, ni siquiera disponemos de varias variables pues sólo estamos observando los flujos activos en un momento dado. Tampoco conocemos qué modelo sigue la distribución de tráfico ni una estimación del error a priori. En resumen, el filtro no está pensado para la tarea que queremos realizar.

3.3. Conclusiones del capítulo

En este capítulo se han presentado los requisitos funcionales del programa implementado, así como diversos requisitos no funcionales. También se ha especificado cuales son los objetivos

del programa implementado, un pequeño estudio de eficiencia a fin de justificar la elección de Python como lenguaje en el que se ha implementado el programa, y como será la presentación de resultados. El programa presentará en la visualización los flujos activos por minuto, creando diez gráficas en función de los tamaños de los flujos de la red bajo estudio. Estas diez gráficas se obtendrán a partir de los deciles de los tamaños de los flujos.

Se ha realizado una explicación de los métodos de detección implementados: media móvil, Holt-Winters de segundo orden y Jacobson, y se han dejado los detalles de la implementación para el capítulo 4. También se han presentado métodos de detección estudiados durante el trabajo pero finalmente no presentes en el programa por diferentes razones, como son el Filtro de Kalman, el método de Hodrick-Prescott, y el método de Holt-Winters de tercer orden.

Una cuestión legítima es la pregunta de si el método de Holt-Winters es en el fondo lo mismo que el método de Jacobson, puesto que ambos utilizan regresión exponencial. No son lo mismo por varias razones: primero, los intervalos de confianza se construyen de manera diferente. En HW se utiliza el estimador de la MAD directamente, mientras que en el método de Jacobson el estimador de la desviación media absoluta se calcula de manera diferente una vez inicializado. La predicción, de ambo métodos es distinta, puesto que HW utiliza dos términos, uno para estimar la pendiente y otro la componente constante, mientras que el método de Jacobson utiliza la media de las observaciones sumada al error obtenido en la última observación.

4

Diseño del programa

En este capítulo se presenta la información necesaria y las decisiones tomadas respecto al diseño del programa implementado.
Concretamente, se explica al lector el funcionamiento del programa implementado para este proyecto, así como el por qué de de las decisiones tomadas. Se repasan los métodos de detección expuestos en la sección anterior y se explican los valores por defecto de los parámetros de los mismos. Por último, se presenta un pequeño estudio de las distribuciones de tráfico en los nodos de RedIRIS analizados, a fin de proporcionar una justificacción adecuada a las categorías en las que se clasificarán los flujos.

4.1. Diseño del programa

Esta sección cubrirá las decisiones relativas al diseño del programa.

Al ser una herramienta específica y desarrollada sin propósito de ser distribuida al ámbito comercial y estar en un estado experimental, no se proporciona interfaz gráfica. Los cambios han de ser realizados en el mismo código fuente.
Así, el programa permite modificar la granularidad de la visualización, en el parámetro correspondiente. Los parámetros de los métodos también están debidamente indicados, así como los parámetros de entrada del programa.

El programa recibe una lista de días, un mes y un año, así como una lista con los nombres de los nodos a buscar. La descripción que ahora sigue se realiza para cada nodo, y para entender el funcionamiento del programa en su totalidad sólo hay que iterar el procedimiento siguiente para cada nodo, y luego repetir el proceso para el siguiente día a procesar. Para cada día, lee el archivo de cada nodo en la ruta especificada y guarda todos los datos en un array de diccionarios. Por cada intervalo de tamaño de flujos se tiene un diccionario, y cada uno de esos diccionarios tiene por claves las horas del día, de acuerdo a la granularidad especificada. Para cada flujo se observa su tamaño en bytes y el tiempo de inicio y de finalización. De esta manera se escoge el diccionario cuyo intervalo contenga al tamaño en bytes del flujo siendo leído, y se suma +1 en todas las entradas cuyas claves estén contenidas en el intervalo de duración del flujo. Este

proceso se repite para cada nodo. En total hay 10 intervalos de tamaños de flujo, por lo que manejamos 10 diccionarios, que llamaremos `datos`. Las razones para la elección de 10 intervalos de tamaños para los flujos se presenta en la sección 4.2

A continuación se produce el cálculo de las predicciones de los distintos métodos. Para cada método, se crea un diccionario por decil (en total 10×3 diccionarios) que tienen como clave el tiempo del día, de acuerdo a la granularidad elegida. Para los métodos que requieren una inicialización previa, estos diccionarios empiezan en la hora correspondiente (por ejemplo, si el método de la media móvil necesita 30 términos para su inicialización, y se eligió granularidad de segundos, la primera clave del diccionario será `'00:00:30'`). A continuación se calculan las predicciones de los tres métodos de detección para cada uno de los 10 diccionarios con los flows. Tras calcular las predicciones, se buscan anomalías en las predicciones en cada uno de ellos. Para cada decil y método de detección se crea un diccionario cuyas claves son el tiempo del día, de acuerdo con la granularidad elegida. Cuando se encuentra una anomalía en el instante `X` para el decil `Y` con el método `Z`, se actualiza la entrada del diccionario a `anomalia[Y][Z][X] = datos[Y][X]`, es decir, a el número de flujos en el momento de la anomalía. Esto nos ayuda posteriormente para dibujar las anomalías.

Realizado este proceso, se grafica cada diccionario `datos` y se hace de manera acumulativa, es decir que para el instante `x` y el decil `i` se representa el punto `(X,Y)` dado por `(X,Y)`$= (x, \texttt{datos[i]} + \sum_{j=1}^{i-1} \texttt{datos[j]})$. Posteriormente se añaden las anomalías para cada método y se guarda el fichero. También se guarda un fichero de texto con el recuento de anomalías en total para el día analizado.

4.1.1. Diagrama de Actividad

En la figura 4.1 se encuentra el diagrama de actividad que ilustra el comportamiento del programa descrito en la sección anterior.

4.2. Análisis de los datos de RedIRIS

4.2.1. Objetivo

Para la realización de este trabajo se han utilizado los datos de tráfico obtenidos en cinco nodos de la red española RedIRIS a lo largo del año 2013. De la muestra se excluirán la segunda quincena de Agosto y la primera quincena de Septiembre porque no se disponen datos de los mismos en los colectores de la UAM. Estos archivos han sido primero descomprimidos y luego filtrados para obtener los datos de los nodos necesarios, de manera que se obtiene un archivo por día y nodo en formato texto. Este archivo es luego procesado para producir la visualización.

Se ha decidido que la visualización divida el tráfico en varias gráficas en función de los tamaños de los flujos. La primera pregunta que surge naturalmente es cómo hacer esta división, es decir, en qué intervalos particionamos los tamaños de los flujos de manera que el resultado sea lo más informativo posible del tráfico de la red.
La solución adoptada consiste en dividir el tráfico en intervalos de acuerdo a los deciles de los tamaños en bytes de los flujos, de manera que se dibujen 10 gráficas con los intervalos del conjunto $G = [0, d_1] \cup_{i=1}^{9} (d_{i-1}, d_i] \cup (d_9, \infty)$ con d_i el decil i-ésimo para el día y nodo estudiados.

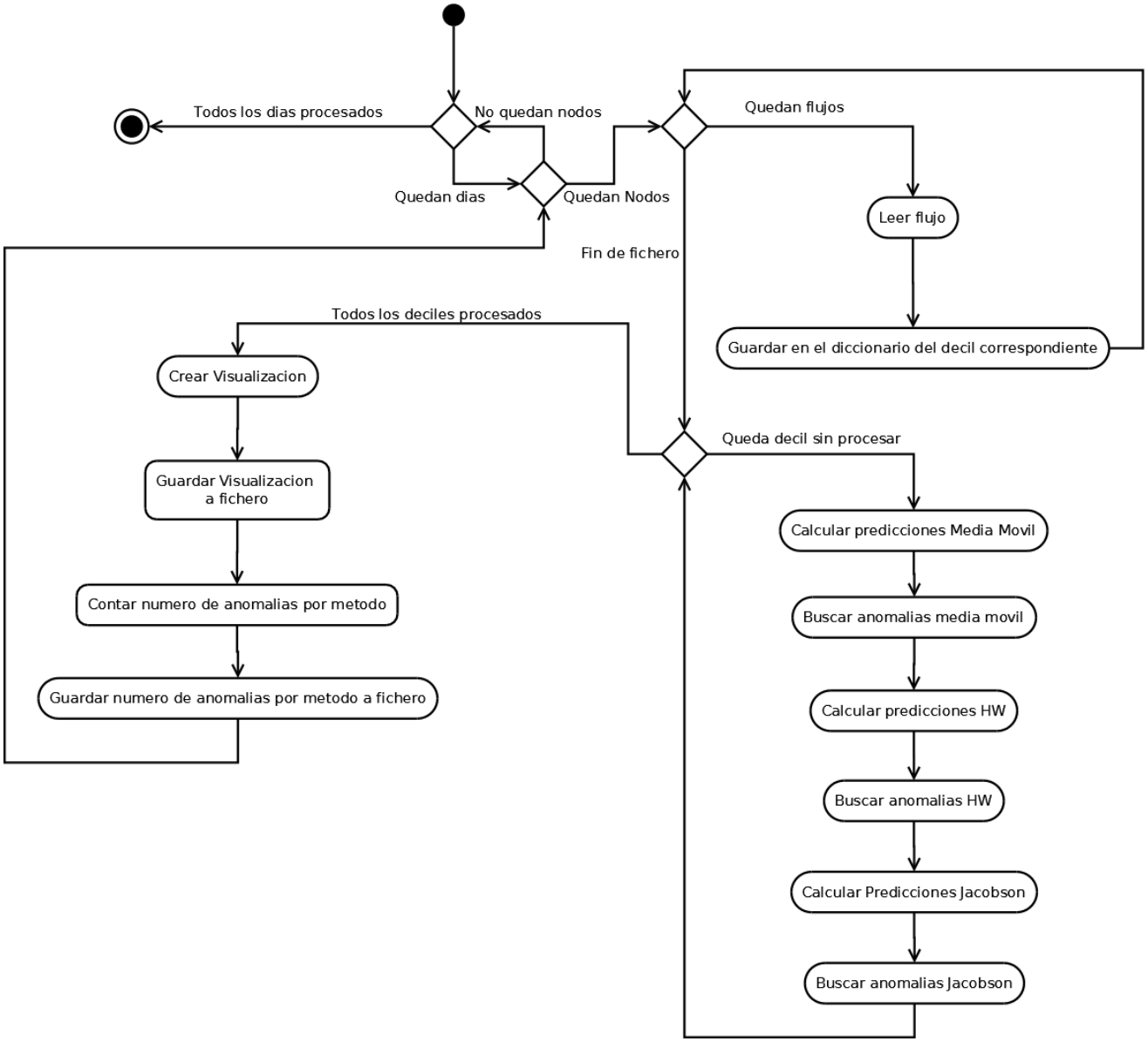


Figura 4.1: Diagrama de Actividad del programa

Este enfoque es adecuado, pero requiere obtener los deciles para cada muestra, por día y nodo de RedIRIS. Esto es computacionalmente costoso puesto que el tamaño de la muestra es elevado (en un sólo día se registran miles de millones de flujos).
La medida que nos gustaría adoptar es que los deciles fueran los mismos para cada nodo de RedIRIS a lo largo del año, es decir, que la distribución del tráfico es aproximadamente la misma independiente del día que estemos observando (a todos los efectos, nos restringimos a los días laborables).

El objetivo de esta sección es por tanto analizar las distribuciones de tráfico para los cinco nodos de redIRIS estudiados y validar la afirmación de que es posible elegir un valor fijo para estos deciles por cada nodo para no tener que calcularlos cada día del año.

4.2.2. Análisis de las distribuciones de tráfico

El planteamiento real del problema se presenta a continuación: dada la muestra de distribuciones para un cierto nodo i durante un año $\{d_i^j : j \in [0, 364]\}$, ¿se cumple que provienen de la muestra de una variable aleatoria X tal que $X_i(\omega_j) = d_i^j$ para un cierto ω_j? En otras palabras, ¿provienen las muestras de los distintos días de un mismo proceso estocástico X? La respuesta a la pregunta con este planteamiento no es trivial, y requiere conocimientos que escapan a la realización de este trabajo. Cabe destacar el trabajo de los autores en [25], que introducen técnicas de análisis funcional (Functional Data Analysis) para el estudio de diversas características de red que actualmente requieren de métodos diversos tanto para el análisis como para su administración. Algunas de estas ideas son aplicables para el trabajo desarrollado en esta sección, como el concepto de *profundiad funcional*.

Por tanto las preguntas a responder en esta sección son dos: ¿ podemos afirmar que son muestras de un mismo proceso estocástico y sus distribuciones son, por tanto, parecidas? si así lo fuera, podríamos tomar los $d_i^j \equiv d_i$ y no sería necesario calcularlos para cada nueva muestra diaria. Así mismo, ¿ cuantas observaciones son necesarias para extender esta justificación a todo el año?

Para reducirlo a un problema abarcable a este nivel nos ceñiremos a métodos estadísticos menos potentes e intentaremos obtener conclusiones que nos sirvan para conseguir respuestas a las preguntas planteadas. Con este fin a continuación se presentan los datos y las técnicas utilizadas en el análisis. En cada apartado se especificarán los datos utilizados para cada experimento.

4.2.2.1. CCDF de los datos

Lo primero a comprobar es que nuestra suposición tiene fundamento. Para ello observamos las funciones de probabilidad acumulada de los días estudiados. En las figuras 4.2, 4.3, 4.4, 4.5, 4.6 se muestran algunos de los resultados obtenidos observando varios días en meses distintos para cada nodo:

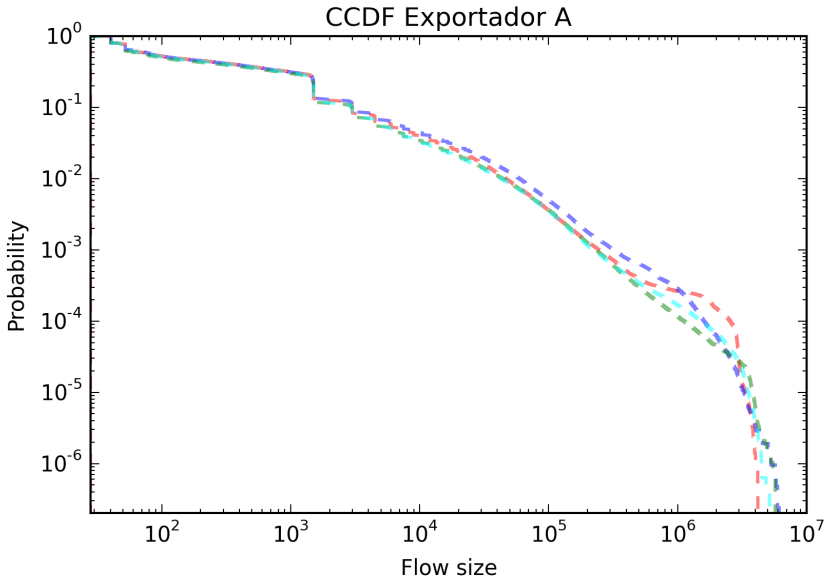


Figura 4.2: CCDF Exportador A para varios días del año 2013

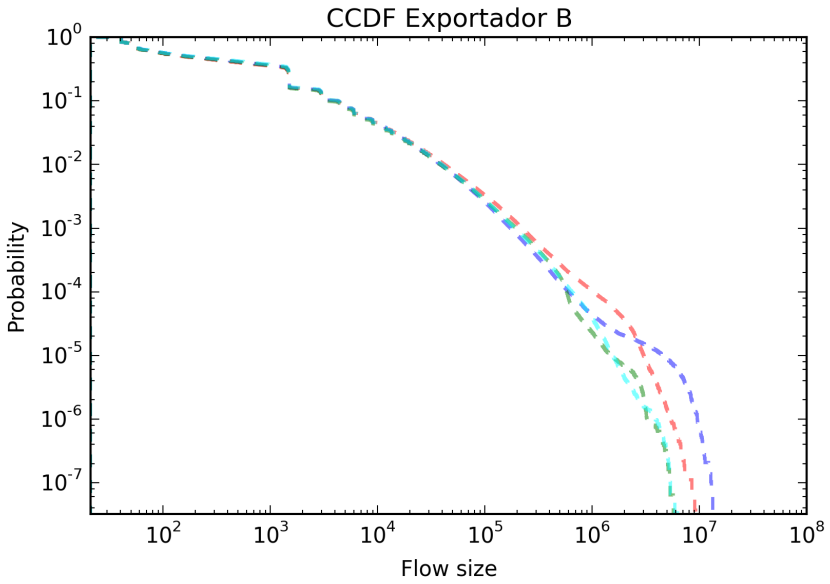


Figura 4.3: CCDF Exportador B para varios días del año 2013

El resto de muestras obtenidas permiten observar que la forma de las CDF y las CCDF se mantiene similar a las mostradas en las figuras anteriores. De estas gráficas se puede obtener más información, como que por ejemplo el muestreo al que se ve sometido RedIRIS en sus nodos provoca que más de la mitad del tráfico registrado es de tamaños inferiores a 100 bytes, y que los flujos de tamaños ≥ 3000 constituyen apenas un 10 % del tráfico registrado. Además apunta a que, al ser casi idénticas para flujos de tamaño ≤ 3000 bytes los deciles serán razonablemente estables. También nos indica que nos encontramos con datos con una gran variabilidad en la cola pesada, pero el último decil está mucho antes de que empiecen las desviaciones debidas a la cola pesada en un mismo nodo. En el siguiente apartado continuaremos el estudio estimando la densidad y calculando los deciles para cada nodo.

4.2.2.2. Estimacion de la densidad y Deciles

A priori no sabemos qué distribución siguen los flujos, pero una estimación de la densidad en cada nodo muestra que las distribuciones son prácticamente idénticas visualmente. La figura 4.7 nos muestra las estimaciones de las distribuciones. Como se ve, la mayor parte se concentra en los flujos de tamaño menor que 200 Bytes. En las figuras 4.7(b), 4.7(c) y 4.7(d) se han llevado a cabo magnificaciones para poder apreciar las diferencias en el intervalo 0-200.

El exportador D es el que presenta, en (d), una diferencia notable respecto al resto, y es una acumulación en torno a los 29 bytes.

Para el siguiente paso se han calculado los deciles de los 20 días laborables del 2013 para todos los nodos, y se ha calculado la media y la desviación típica de cada decil para cada nodo. Los resultados pueden verse en las tablas 4.1, 4.2, 4.3, 4.4 y 4.5.

De un primer vistazo de estas tablas podemos ver que al menos los primeros deciles son bastante estables para todos los nodos. Bastantes, especialmente en los primeros deciles, tienen desviación típica cero. En varios nodos el octavo decil suele ser 1500 bytes o un número cercano.

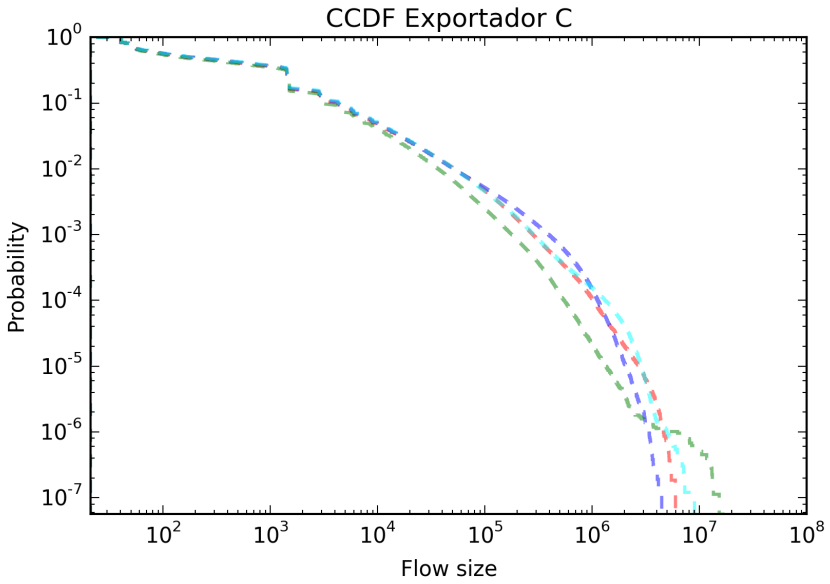


Figura 4.4: CCDF Exportador C para varios días del año 2013

Decil	Media	Desv. Típica
1	40	0
2	44	2.1249
3	52	0
4	63	1.8294
5	111	6.0781
6	339	27.6553
7	1118	72.8047
8	1500	0
9	2994	7.6665

Tabla 4.1: Medias y desv. típicas para los deciles del exportador A

Decil	Media	Desv. Típica
1	40	0
2	48	0
3	52	0
4	79	0.2575
5	170	17.8462
6	604	81.9191
7	1467	11.0951
8	1500	0
9	3871	515.4617

Tabla 4.2: Medias y desv. típicas para los deciles del exportador B

Esto ocurre con flujos en los cuales Netflow sólo ha observado un paquete con máximo *payload*, y es prácticamente consistente en todos los nodos.
De estas podemos afirmar que en los deciles que se tiene desviación típica 0 o una desviación típica pequeña ($\sim$ 10bytes) es justificable elegirlos para representar la visualización.
No está tan claro con deciles que presentan mayor desviación. Para la media deciles que muestran una desviación mayor nos preguntamos si la muestra de deciles a partir de la cual se han obtenido está normalmente distribuida. De ser así, también podremos elegir la media obtenida para la visualización.
A tal efecto se presentan varias de las gráficas cuantil-cuantil (QQ) frente a los puntos de la muestra que se ajustan a una normal. Algunos de ellos se muestran en la figura 4.8,

Sin embargo, las muestras de varios deciles, en especial los del exportador D y otros, no se ajustan a una normal, como se muestra en la figura 4.9. Ante esta disparidad de resultados necesitamos otro enfoque que nos permita tomar una decisión al respecto.

A tal efecto se han efectuado tests de Kolmogorov-Smirnov dos a dos entre las muestras de

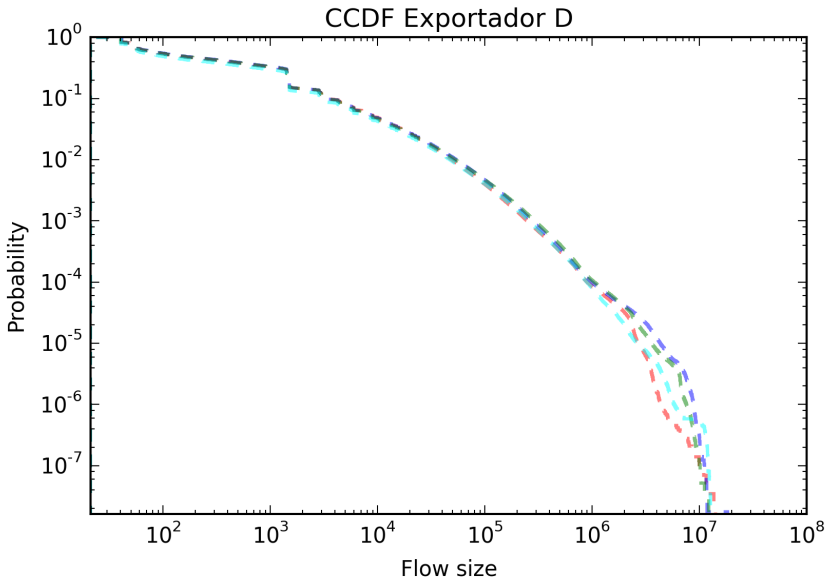


Figura 4.5: CCDF Exportador D para varios días del año 2013

Decil	Media	Desv. Típica
1	40	0
2	48	0.9819
3	52	0.2575
4	84	8.4164
5	197	49.1036
6	781	275.1791
7	1435	23.8258
8	1500	0
9	4050	555.8708

Tabla 4.3: Medias y desv. típicas para los deciles del exportador C

Decil	Media	Desv. Típica
1	40	0.0
2	48	1.6629
3	52	2.8458
4	79	11.8300
5	175	55.8296
6	611	263.9604
7	1395	81.3630
8	1500	0.0
9	3207	436.9017

Tabla 4.4: Medias y desv. típicas para los deciles del exportador D

deciles para un mismo nodo. De esta manera queremos comprobar si los deciles obtenidos cada día siguen o no la misma distribución. En este caso, el objetivo que estamos buscando es no obtener evidencia estadística suficiente en contra de la hipótesis nula del test H_0*: D_i y D_j son homogéneas*, donde D_i y D_j son las muestras de deciles para un mismo nodos en días i y j.
Los resultados del los tests dos a dos de Kolmogorov-Smirnov son que en **ningún caso** hay suficiente evidencia estadística para rechazar H_0 a un nivel de significación aceptable. El p-valor mínimo obtenido para esta batería de tests es de 0,95747, un valor altísimo que, aunque no prueba en el sentido estricto de la palabra que las distribuciones sean las mismas, da la confianza suficiente como para asumirlo al nivel que estamos trabajando.

A pesar de estos resultados, la alta variabilidad del último decil en todos los nodos nos indica que nos encontramos ante una cola pesada. Sin embargo, si tenemos en cuenta que nuestro objetivo es detectar comportamientos anómalos en la red proveniente de ciertos ataques, no es tan relevante si el último decil se elige con ±200bytes. Un ataque en los flujos de mayor tamaño sería probablemente un escenario de robo masivo de datos. En este supuesto, que el último decil sea 3000 Bytes o 2740 Bytes nos va a importar poco porque el ataque se detectará de todas maneras. La división por deciles da más precisión a la detección a para los flujos pequeños, y

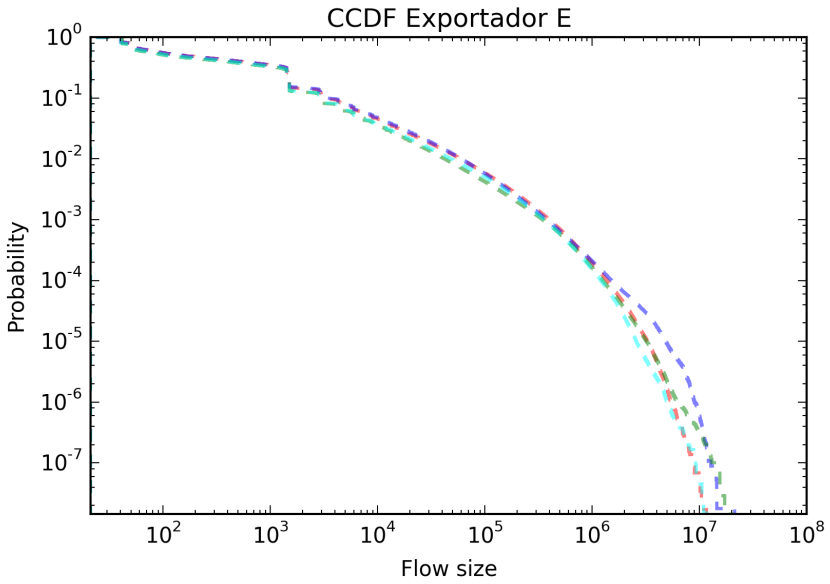


Figura 4.6: CCDF Exportador E para varios días del año 2013

Decil	Media	Desv. Típica
1	40	0
2	42	2.6178
3	51	2.8329
4	67	8.8074
5	121	26.5103
6	410	112.9300
7	1289	252.6258
8	1488	22.4950
9	3030	302.7147

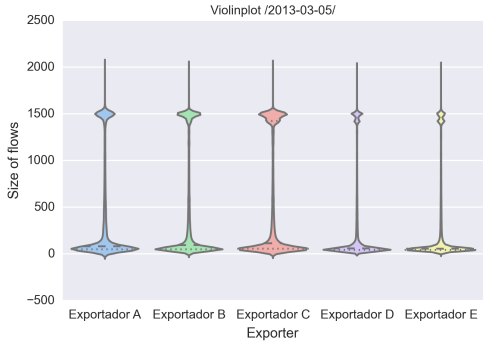
Tabla 4.5: Medias y desv. típicas para los deciles del exportador E

es ahí donde un analista tiene que poner su atención ya que los flujos que causan anomalías a causa de una ataque suelen ser flujos pequeños.

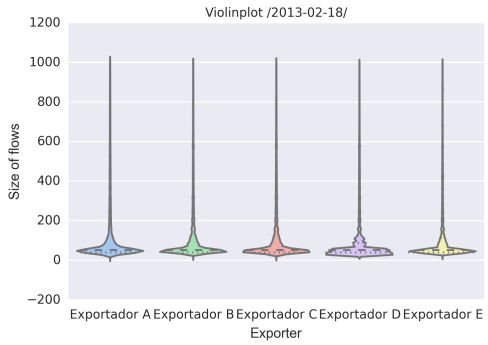
4.2.2.3. Pruebas Adicionales

Además de las pruebas realizadas y ya expuestas, se han realizado una prueba adicional, en esta ocasión no con muestras de deciles, sino con las muestras de tráfico obtenidas durante los 20 días laborables de 2013.

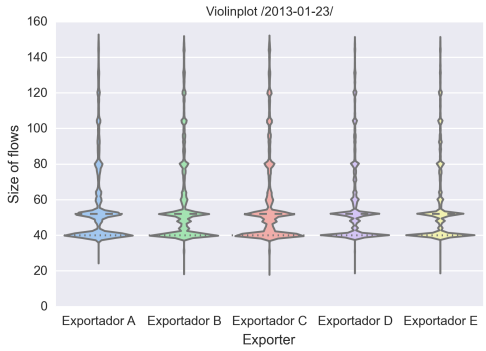
- **Medición de la distancia de Hellinger**: Con los datos de los que disponemos se ha medido la distancia de Hellinger dos a dos entre los días de un mismo nodo. El máximo valor obtenido para todas ellas es de 0.11. No es sorprendente, puesto que la distancia de Hellinger mide en esencia lo mismo que el test de Kolmogorov-Smirnov, es decir, que las distancias entre las funciones de distribución de los días realizada dos a dos es pequeña, y suficientemente pequeña para no dar evidencia en contra de la hipótesis nula en el test de KS, como se explicó en la sección anterior.



(a) Est. Densidades 03/05. Maximo 2000 Bytes



(b) Est. Densidades día 05/07. Máximo 1000 Bytes



(c) Est. Densidades 01/23. Máximo 150 bytes

(d) Est. Densidades 03/05. Máximo 200 bytes

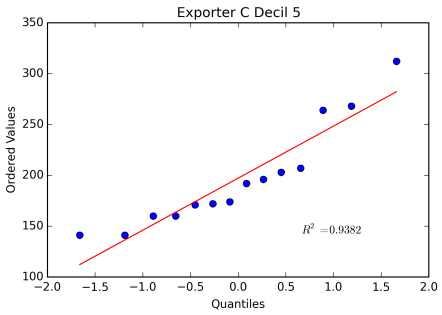
Figura 4.7: Estimaciones de la densidad, para varios días con varios máximos sobre el tamaño de los flujos

4.2.3. Conclusiones del análisis

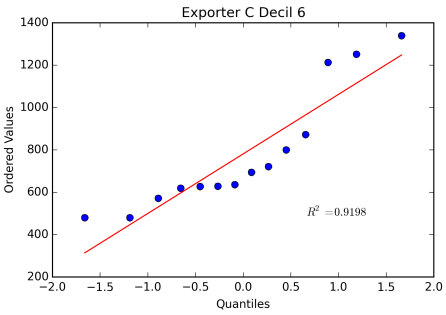
Las medias obtenidas de la muestra utilizada pueden ser utilizadas como los límites de los intervalos d_i^j que introduciamos al inicio de la sección. Bien por presentar una desviación típica muy baja, bien por ajustarse las muestras de dicho decil a una normal, o bien por observar que las muestras de los deciles de un mismo nodo dos a dos no presentan evidencia estadística significativa en contra de la hipótesis nula del test de Kolmogorov-Smirnov para ningún nivel de significación relevante. De este modo, las medias obtenidas en las tablas 4.1, 4.2, 4.3, 4.4 y 4.5 serán los utilizados por el programa a la hora de presentar las visualizaciones de los nodos correspondientes.
La muestra que se ha utilizado es la de los veinte días laborales del año 2013. La legítima pregunta de si es suficiente una muestra de veinte días para estimar los deciles a lo largo del año no está del todo clara. Es asumible que para los meses a lo largo del curso académico los resultados obtenidos habrían sido los mismos. Los meses de verano podrían a priori presentar una distribución diferente, pero precisamente no se disponen de datos de Netflow para los cinco nodos entre la segunda quincena de Agosto y la primera quincena de Septiembre de 2013, por lo que no debería de suponer un problema.

4.3. Datos de RedIRIS 2013

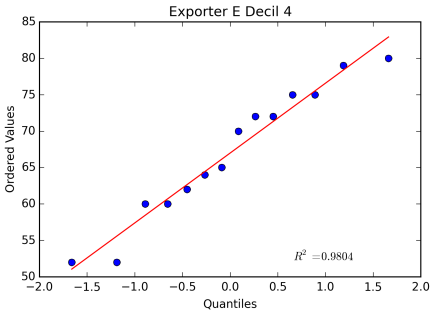
Para la validación del programa desarrollado se utilizarán los datos de tráfico de una selección de 5 nodos de RedIRIS. Los datos de tráfico están en formato *pcap*. Para pasarlos al



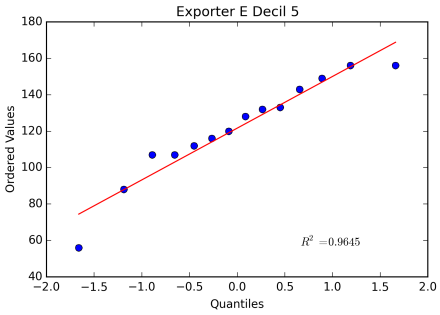
(a) QQ-Plot Exporter C Decil 5



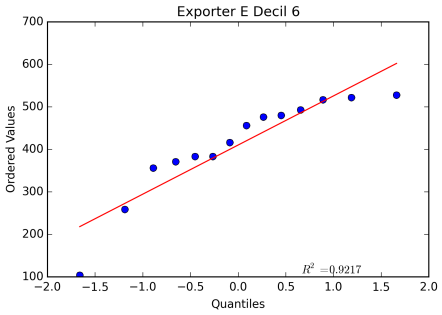
(b) QQ-Plot Exporter C Decil 6



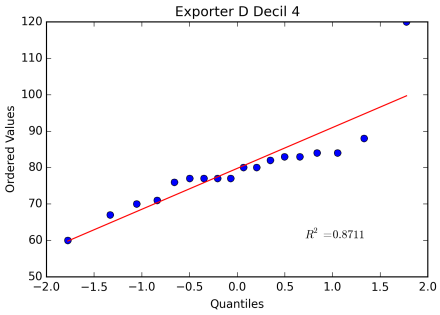
(c) QQ-Plot Exporter E Decil 4



(d) QQ-Plot Exporter E Decil 5



(e) QQ-Plot Exporter E Decil 6



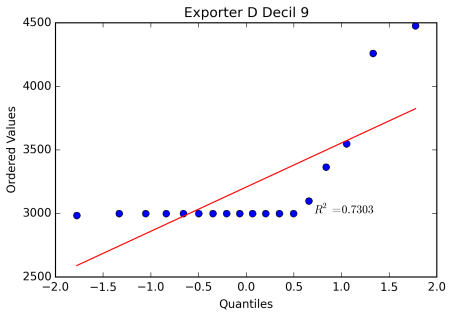
(f) QQ-Plot Exporter D Decil 4

Figura 4.8: QQ-Plots de Varios Deciles contra una normal a la que se ajustan

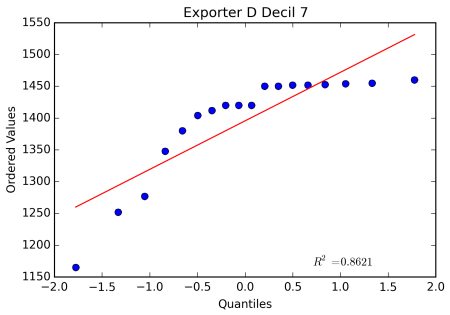
formato texto que lee el programa implementado, es necesario un pequeño script usando *flow-tools*. Debido al tamaño que ocupan en disco (1Tb todo el año comprimido, y $\sim$ 600Gb un mes descomprimido) ha sido necesario descomprimir los archivos mes a mes, procesarlos y luego borrarlos.

Los registros de tráfico de Agosto y Septiembre no se incluirán en el estudio puesto que están incompletos o defectuosos. No todos los días tienen registros y los días que sí tienen resultan incompletos. Se deduce por tanto que en estas fechas el colector, el exportador o ambos estuvieron fuera de servicio.

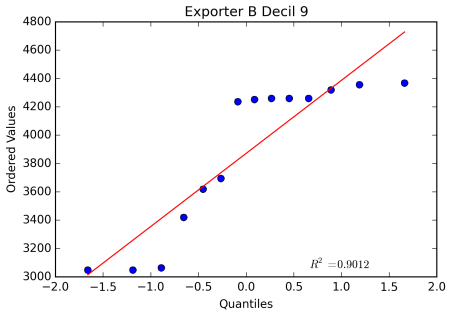
El programa requiere que el fichero de entrada tenga la siguiente estructura por columnas: `Start, End, Sif, SrcIPaddress, SrcP, DIf, DstIPaddress, DstP, P, Fl, Pkts, Octets`. El significado de los campos se encuentra en la 2.1 de la sección 2



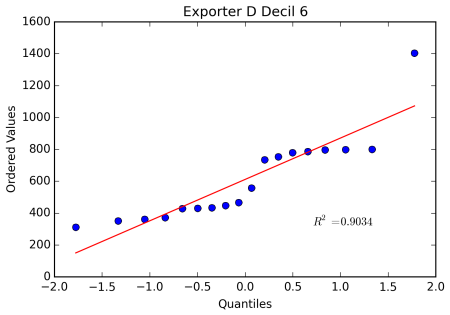
(a) QQ-Plot Exporter D Decil 9



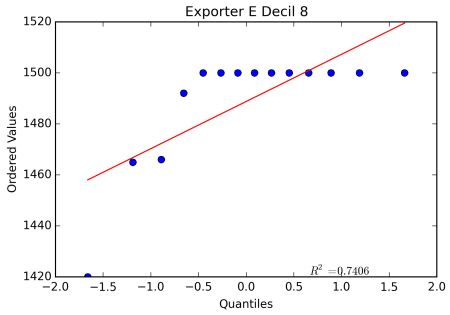
(b) QQ-Plot Exporter D Decil 7



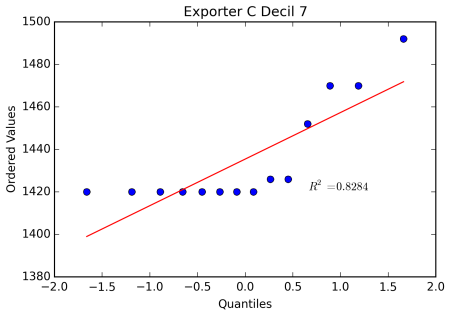
(c) QQ-Plot Exporter B Decil 9



(d) QQ-Plot Exporter D Decil 6



(e) QQ-Plot Exporter E Decil 8



(f) QQ-Plot Exporter C Decil 7

Figura 4.9: QQ-Plots de Varios Deciles contra una normal a la que no se ajustan

4.4. Conclusión

En esta sección se han presentado las cuestiones relativas al diseño del programa desarrollado. Se ha presentado mediante una explicación y el diagrama de actividad el funcionamiento del programa. También se han expuesto las razones de los valores de los párámetros por defecto para los distintos métodos de detección. Asimismo se ha realizado un estudio de las distribuciones de tráfico en los nodos a fin de justificar las elecciones de intervalos de tamaño, de acuerdo a los deciles de dichas distribuciones. Por último, se ha dado una pequeña presentación de los datos utilizados en la validación de la herramienta, que se presenta en la sección siguiente.

5

Pruebas y Validación de resultados

En este capítulo se presentan las pruebas realizadas con los datos de RedIris 2013 para los cinco exportadores de RedIris que estamos estudiando, así como las conclusiones que se obtienen de observar los resultados. Se presentan conclusiones a nivel de cada nodo, así como un pequeño estudio en profundidad de algunas anomalías encontradas, y conclusiones a nivel de cómo afecta el muestreo a la detección de anomalías.

5.1. Pruebas Realizadas

5.1.1. Pruebas unitarias de caja negra

El programa se ha sometido a pruebas unitarias para comprobar entradas mal formadas y el correcto comportamiento de los métodos de predicción. Asimismo también se ha comprobado su correcto funcionamiento al modificar la granularidad seleccionada.

5.1.2. Pruebas Métodos de detección

El objetivo del programa diseñado es ejecutarlo sobre los datos de tráfico de días laborables de RedIRIS del año 2013 para 5 exportadores. Previo a este proceso se han realizado varias pruebas para calibrar los parámetros de los distintos métodos y comprobar su correcto funcionamiento. Con este fin se han usado varios ficheros de tráfico elegidos de forma aleatoria: un fichero del exportador B con fecha del 26/12/2011, y dos ficheros del exportador C del 11/02/2013 y 18/02/2013, respectivamente. Estos ficheros contienen anomalías en diferentes momentos del día.

Como se expuso en la sección 3.2, los diferentes métodos implementados necesitan de varios parámetros que ajustar antes de poder ejecutarlos sobre nuestro dataset. Así, estos ficheros han servido para observar los cambios que se producen en la detección al modificarlos.
En esta sección expondremos explicaremos los motivos de la elección de los parámetros en función de estas pruebas.

5.1.2.1. Ficheros utilizados para las pruebas

El primer fichero utilizado para las pruebas es del exportador B con fecha del 26/12/2011. Este fichero se muestra en la figura 5.1

A fin de entender el fichero de prueba se ha investigado cual es la anomalía que causa ese aumento de flujos. Para ello se ha utilizado AWK. Las anomalías que se producen entre las 17 y las 19 son dos eventos mezclados: escaneos de puertos de varios hosts, y un ataque masivo de diccionario. Los escaneos de puertos son tanto horizontales como verticales, por lo que se puede recomponer de los datos: la misma IP escanea puertos distintos y no asignados de varios hosts internos. No es sólo una IP de origen la causante del tráfico de escaneo, sino que son varias. La lista de puertos de destino accedidos durante esas dos horas muestra muchos números de puertos no asignados con un sólo acceso. Respecto al ataque de diccionario, durante ese intervalo se producen un total de 283453 flujos de SSH, de los cuales prácticamente todos se originan en la IP `x.y.203.193`. Filtrando estos flujos se observa que intentan sucesivamente conectarse a un amplio rango de hosts de destino de manera muy consecutiva. Sin mirar el payload de los paquetes, podemos decir que se trata o bien de un ataque de diccionario contra varios hosts, o bien un intento de exploit via SSH para los hosts afectados.

Estos eventos son precisamente los que queremos detectar con la herramienta diseñada. Un observador hábil, además, observará que las anomalías se producen en el intervalo de 40-48 Bytes de tamaño de flujo, y el tamaño de un paquete SSH de inicio de sesión es exactamente 48 bytes, de manera que un incluso a primera vista uno puede hacerse una idea de qué tipo de ataque puede ser. La visualización por deciles de tamaños de flujo permite, como valor añadido, comprobar cual es el intervalo de tamaño en el cual se origina una anomalía que afecta al total de tráfico de la red. En otros ejemplos, puede ser que las anomalías se den en varios deciles a la vez.
Los otros dos ficheros utilizados para la calibración de parámetros presentan las mismas características: tráfico normal durante todo el día y un periodo que produce anomalías bien localizadas. Estos dos ficheros se muestran en las figuras 5.2 y 5.3. A continuación se presentan las decisiones de diseño de los parámetros de los distintos métodos de detección, que han sido ajustados con los ficheros arriba mencionados.

5.1.2.2. Método de Media Móvil

En este método los parámetros libres que se pueden ajustar son el tamaño de la ventana (n) para el número de términos usados en calcular la media, y los coeficientes para los límites superior e inferior.

El tamaño de la ventana es preciso ajustarlo de acuerdo con la granularidad a la que ejecutemos el programa. A efectos de este proyecto, se ha ejecutado con granularidad de minutos y se ha tomado como tamaño de la ventana $n = 30$. De esta manera, se tienen en cuenta la primera media hora de cada día antes de empezar a realizar predicciones. En el caso en que la granularidad fuera mayor (un segundo, cinco segundos) entonces sería recomendable aumentar el tamaño de la ventana para tener una mejor apreciación de la media muestral hasta ese momento.

Los coeficientes de los límites superior e inferior se han dejado en superior = 0,6 e inferior = 0,5. Se ha aumentado ligeramente el coeficiente superior para evitar que subidas naturales del tráfico sean erróneamente catalogadas como anomalías, aunque dada la naturaleza del método, seguirá ocurriendo ya que no soporta bien cambios pronunciados en la media.

El límite inferior se mantiene a un valor medio por defecto, pues a priori estamos interesados en los picos y no en los valles de la gráfica, aunque estos también pueden ser indicios de ciertos comportamientos, como por ejemplo un vaciado repentino de la caché de flujos, una caída del servicio o un apagado del exportador.

5.1.2.3. Método de Holt Winters de segundo orden

Los parámetros ajustables de este método son varios, como se explicó en la sección 3.2.2. Los valores elegidos para la ejecución del proyecto han sido los siguientes:

- $\alpha = 0{,}2$, dando mayor peso por tanto a las observaciones anteriores.
- $\beta = 0{,}1$. Este parámetro da menor peso a la nueva observación de la pendiente frente al valor pasado.
- $n = 30$ para el número de muestras necesario para el cálculo inicial de la desviación media absoluta. Este parámetro ha de ser modificado en función de la granularidad.
- $\delta = 0{,}3$. Este valor da más peso al valor de la MAD en el paso anterior que a la nueva observación, sin ser excesivamente conservador.
- $\sigma = 2{,}6$. Éste parámetro ha sido el más difícil de ajustar para la validación de los resultados con los datos de RedIRIS. El primero valor probado fue de 0.6, pero el método devolvía demasiados falsos positivos. Fueron necesarias varias iteraciones de prueba y error hasta encontrar un valor adecuado que no arruinara la validez del método. Se recomienda encarecidamente no cambiarlo.

5.1.2.4. Método de Jacobson

Para el método de Jacobson se necesita ajustar en primer lugar, el parámetro n para la inicialización de la media muestral y de la desviación media absoluta. Este parámetro ha sido ajustado a 30 de igual manera que en los anteriores métodos, y también es necesario modificarlo en función de la granularidad de acuerdo con lo explicado anteriormente.

De los valores recomendados por Jacobson para g y h sólo ha sido modificado $h = 0{,}4$, mientras que g se ha mantenido a su valor recomendado $\frac{1}{8}$.Es de esperar, puesto que los valores originales están pensados para los tiempos de retransmisión en TCP, una situación completamente diferente a la que nos enfrentamos ahora. De esta manera damos más peso a la estimación de la desviación típica.

Esta misma razón es la que ha llevado a modificar el factor 4 de la expresión:

$$\mathrm{RTO} = A + 4D$$

por el valor 2

$$\mathrm{RTO} = A + 2D$$

Para los valores altos que estamos gestionando (en un instante dado el número de flujos activos de una determinada familia es del orden de 10^3) no es adecuado utilizar el factor 4. Es excesivo y por tanto no detecta anomalías. Un valor de 2 es más razonable y produce buenos resultados.

5.2. Conclusiones

En esta sección se han presentado las pruebas que se han llevado a cabo sobre el programa. Se han presentado, en primer lugar, las pruebas unitarias de caja negra sobre el programa. A continuación se han explicado como se han realizado las pruebas para ajustar los valores predeterminados de los métodos de detección: mediante prueba y error frente a archivos cuyo contenido nos permite diferenciar claramente los comportamientos anómalos frente al tráfico normal. Cuánto afectan estos parámetros a la cantidad de anomalías que detectan (comparativamente hablando) se estudiará en la sección 6.

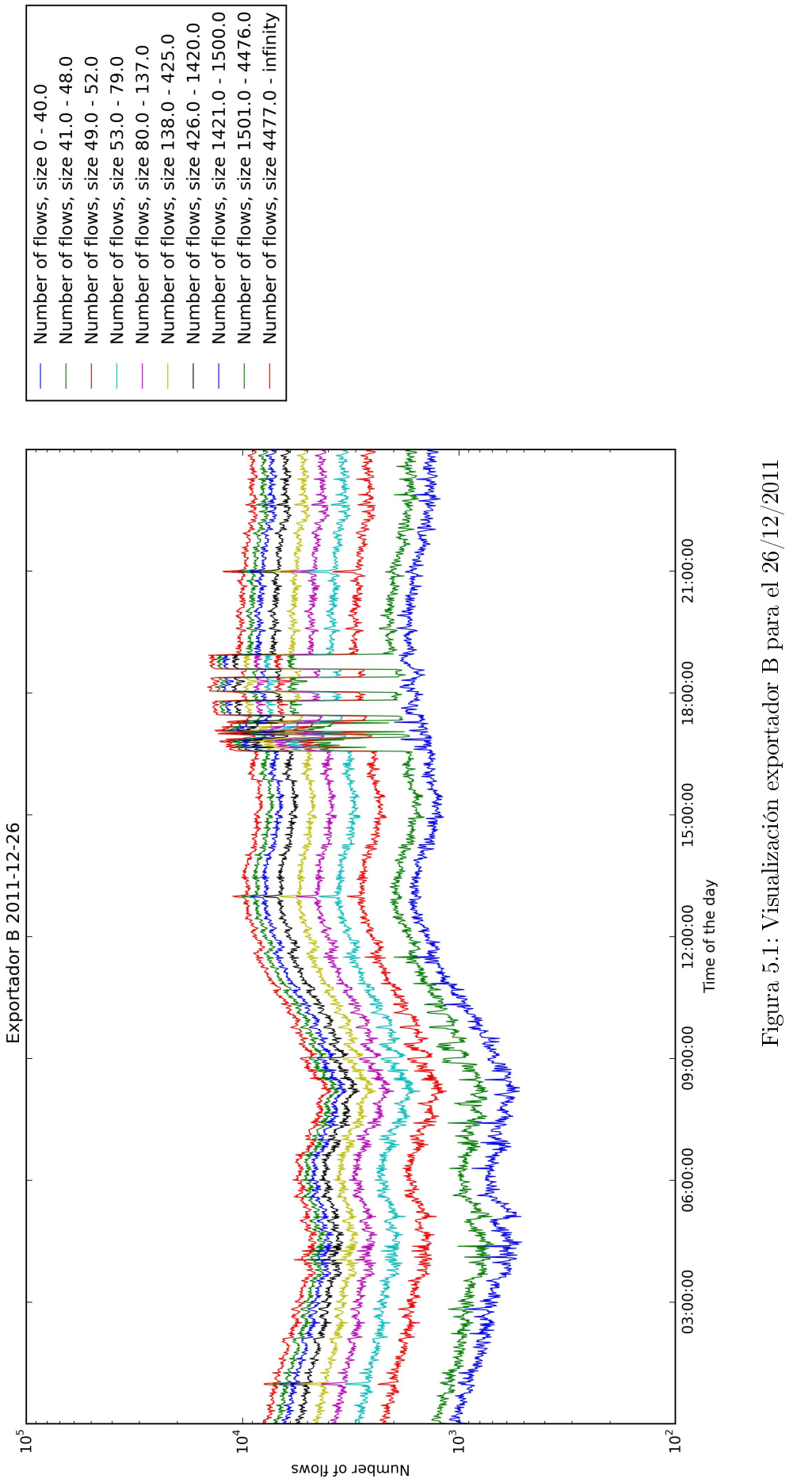


Figura 5.1: Visualización exportador B para el 26/12/2011

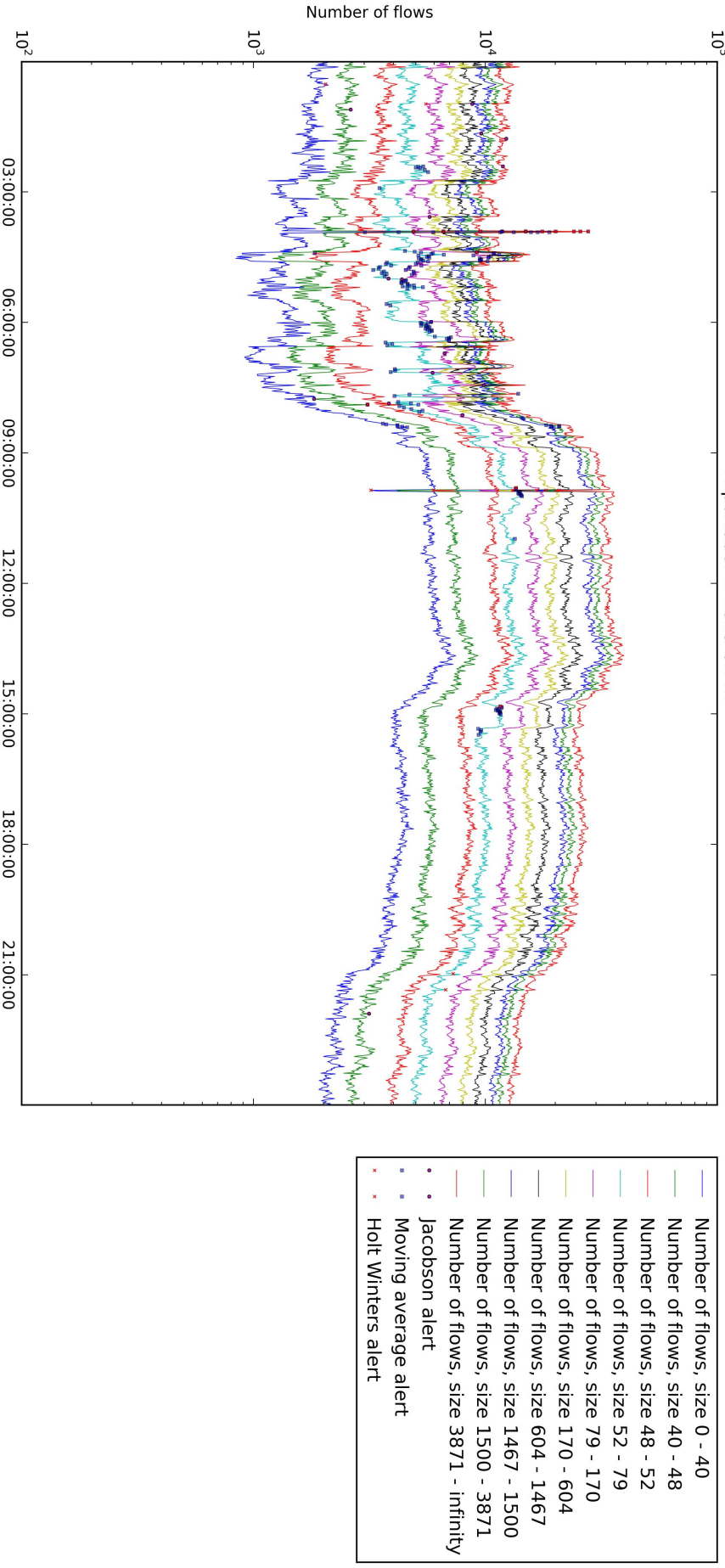


Figura 5.2: Visualización exportador B para el 27/02/2013

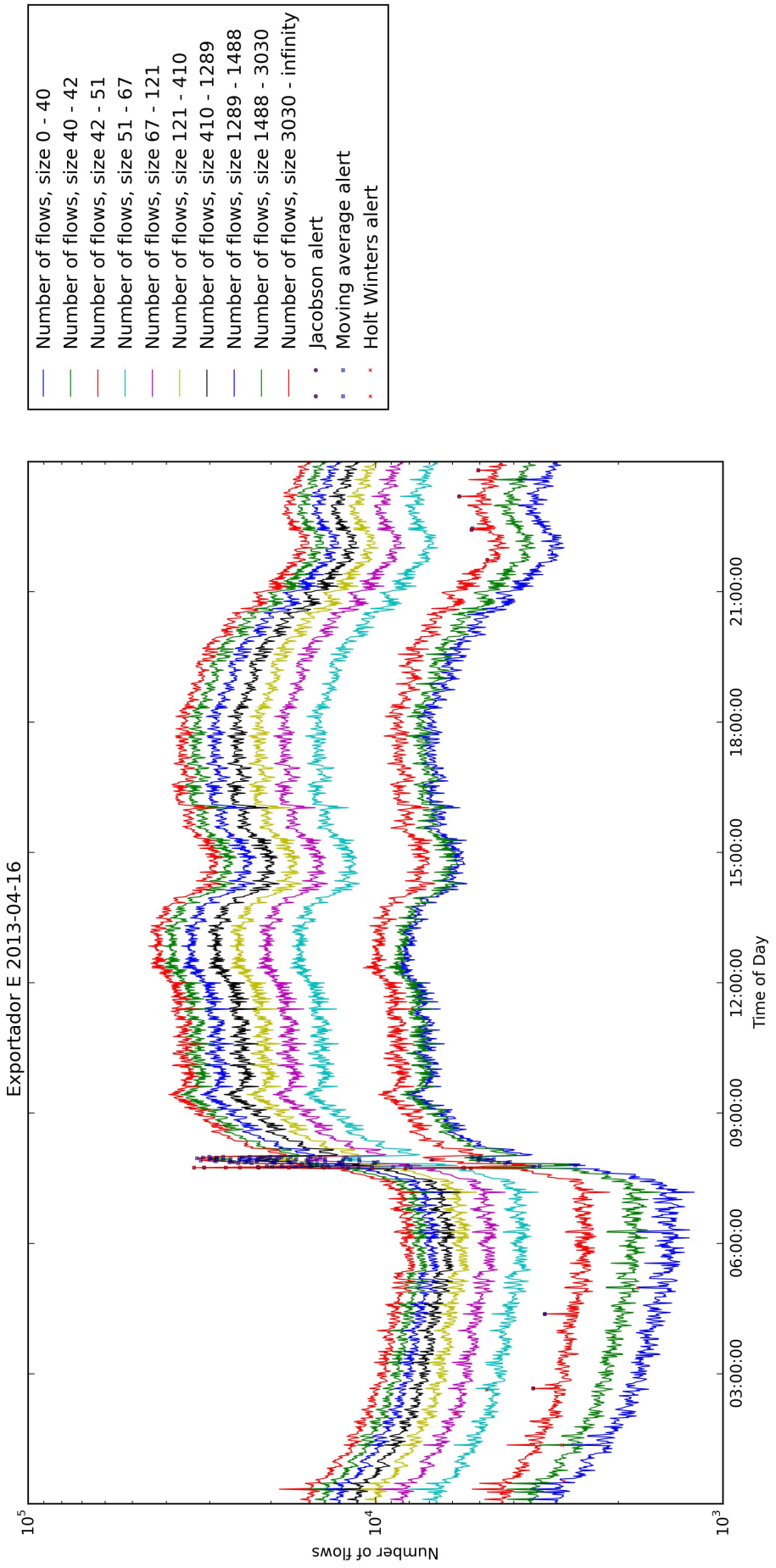


Figura 5.3: Visualización exportador E para el 16/04/2013

6

Resultados y discusión

En este capítulo se presentan los resultados de ejecutar el programa implementado frente a los datos de tráfico de cinco exportadores de RedIRIS durante el año 2013. Se hará un pequeño estudio de cada nodo, observando las carácterísticas particulares de cada uno, así como un análisis en profundidad de tres anomalías elegidas a lo largo del año.

6.1. Resultados Generales

Las pruebas de todo el año han tardado un total de 24 días. Durante este tiempo se han analizado todos los días laborables de los exportadores. En las siguientes subsecciones veremos como se ha comportado cada exportador en particular.
Como observación general se puede comprobar que el número de anomalías detectadas con el método de media móvil es mucho mayor a los números de los otros dos métodos, confirmando la hipótesis que habíamos establecido en la sección 3.2.1, a saber, que al reaccionar de manera lenta frente a cambios su número iba a ser mucho mayor, además de aportar muchos falsos positivos fruto de las subidas y bajadas naturales del tráfico en la red.

En las gráficas también aparecen repentinas bajadas puntuales en el tráfico, que al instante siguiente vuelven al volumen de tráfico normal. Estas anomalías se producen cuando el registro de flujos alcanza su límite y en exportador empieza a catalogar los más antiguos como finalizados, provocando un vaciado de la caché. Si este comportamiento dura más en el tiempo, entonces puede ocurrir que sea una parada por mantenimiento u otras razones.
Al no disponer de los datos sin muestreo es importante notar que no se pueden dar métricas de aciertos de las predicciones. Ni siquiera si estuvieran sin muestrear sería posible, pues en esencia se necesitaría que las trazas de red estuvieran etiquetadas.
También es importante hacer notar que en las fechas de agosto y septiembre no se disponen de datos, puesto que a partir del 12 de Agosto hasta el 19 de Septiembre se apagaron los colectores de datos de la UAM no estuvieron en funcionamiento.

6.1.1. Exportador A

Los ficheros del exportador A son los más pequeños de todos los disponibles, con un tamaño medio en torno a los 500Mb. Por ello, las visualizaciones resultan ser gráficas con muchos picos y valles, lo cual causa demasiados falsos positivos, incluso de los métodos de Holt Winters y Media móvil.
Al ser una red más pequeña, resulta ser el exportador con subidas más abruptas a las 8 de la mañana. En la figura 6.1 se observa el número de anomalías a lo largo del año y en la figura 6.2 se observa el número de anomalías por método mensuales.

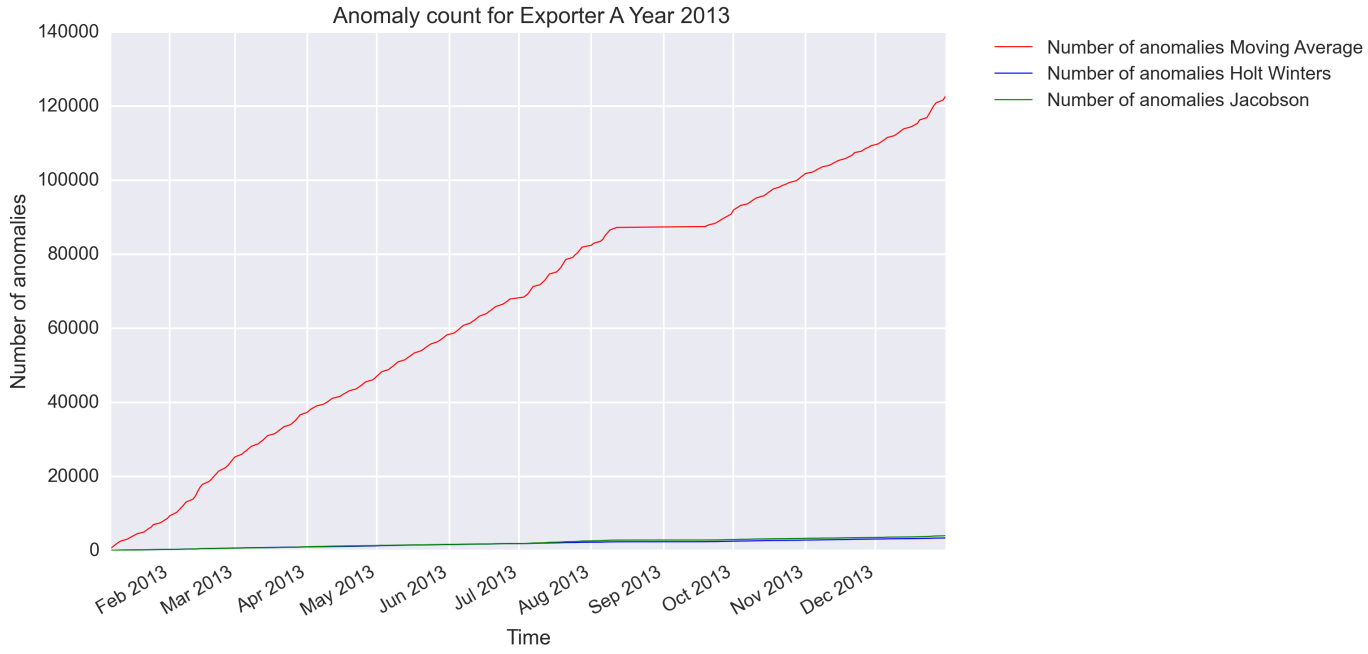


Figura 6.1: Anomalías a lo largo del año para el exportador A

En este caso el número de anomalías de la media móvil ha sido excesivo comparado con los de los demás, en parte por el menor tamaño de los ficheros. Algunas de las gráficas obtenidas están en las figuras 6.3, 6.4 y 6.5

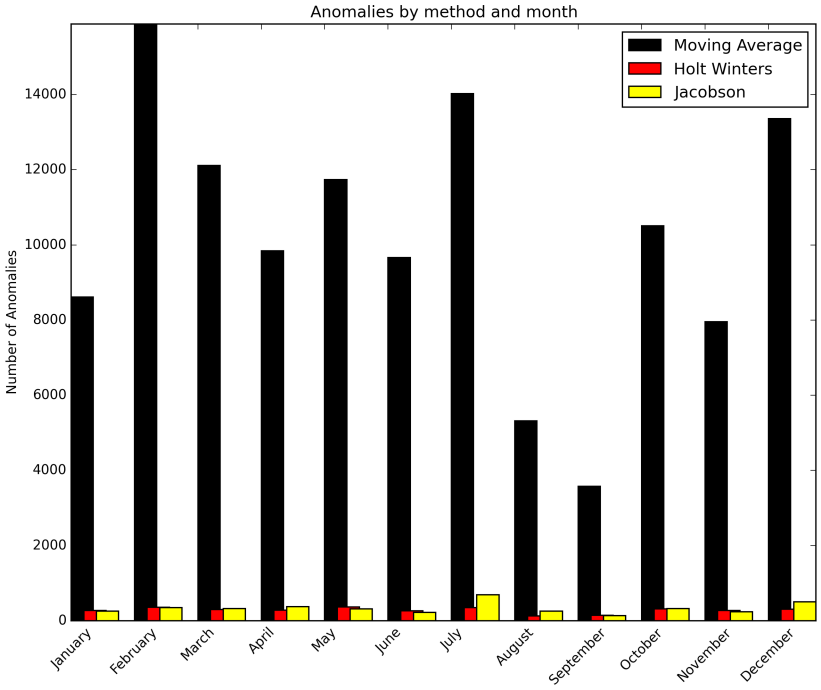


Figura 6.2: Anomalías por mes y método para el exportador A

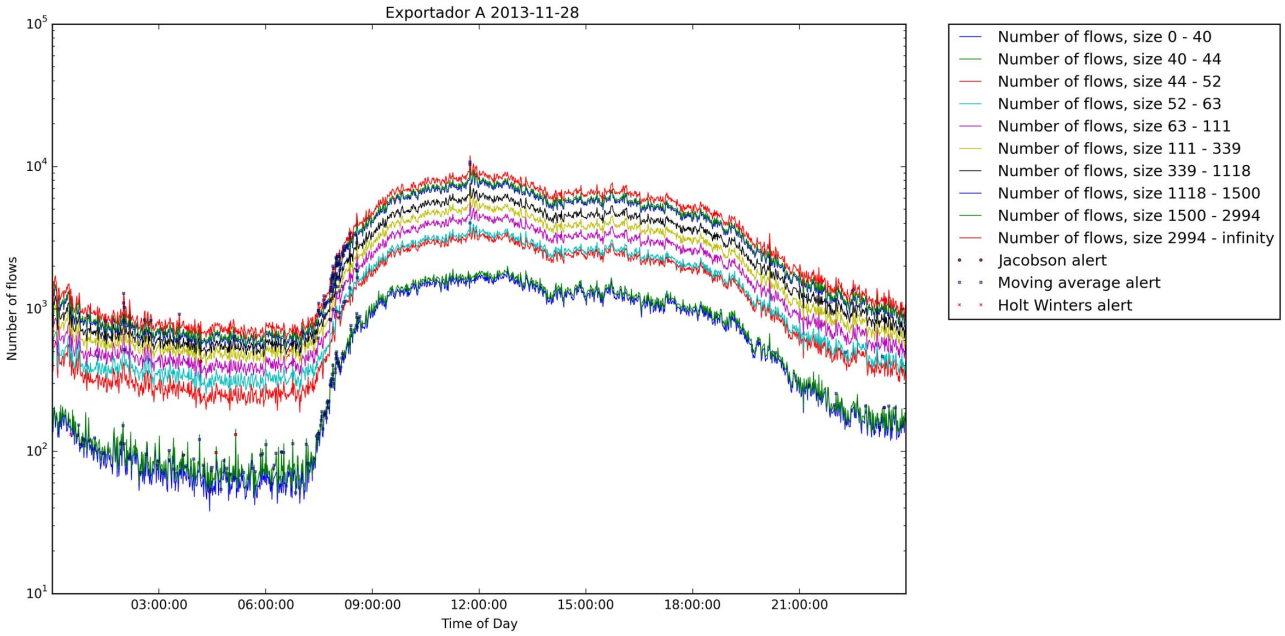


Figura 6.3: Visualización del 28/11 para el exportador A

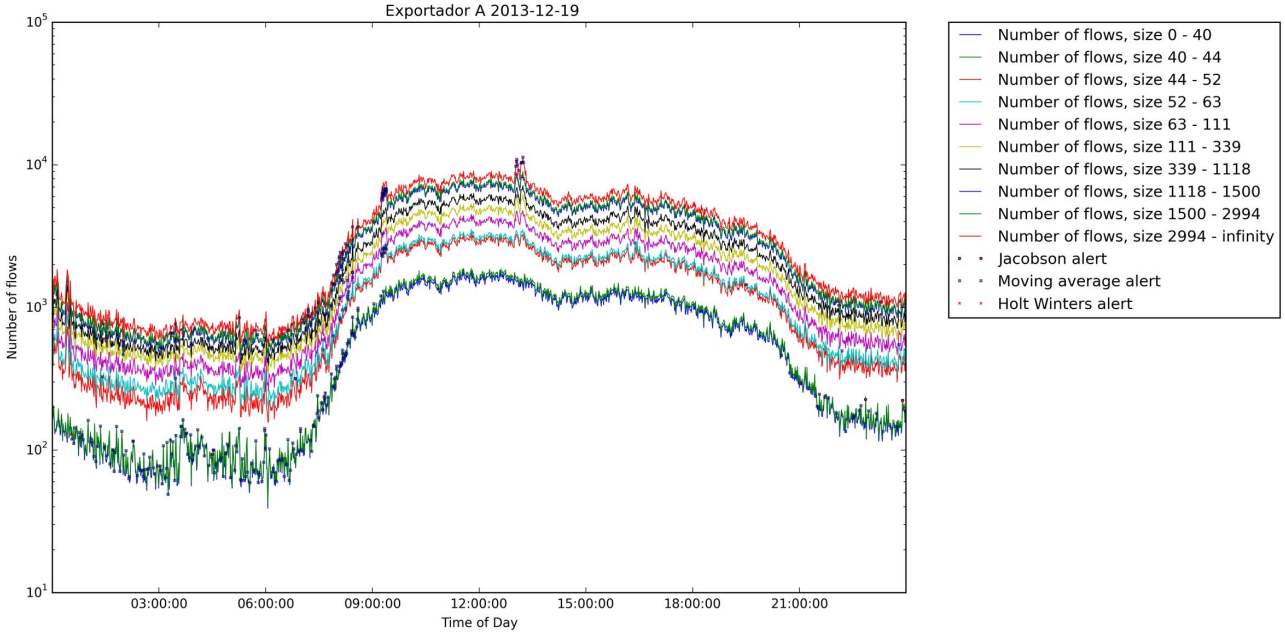


Figura 6.4: Visualización del 19/12 para el exportador A

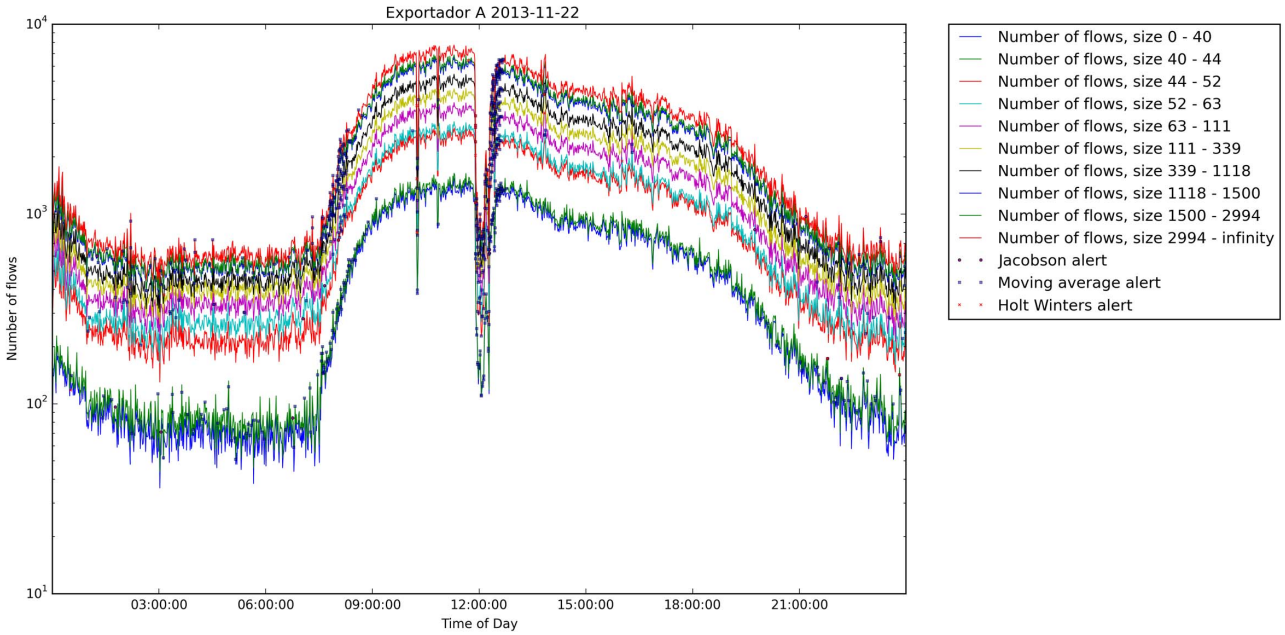


Figura 6.5: Visualización del 22/11 para el exportador A

6.1.2. Exportador B

El exportador B presenta figuras más razonables respecto al número de anomalías detectadas por cada método, aunque de nuevo la media móvil obtiene valores demasiado altos. Las figuras 6.6, 6.7 muestran las anomalías a lo largo del año y mensualmente, respectivamente.

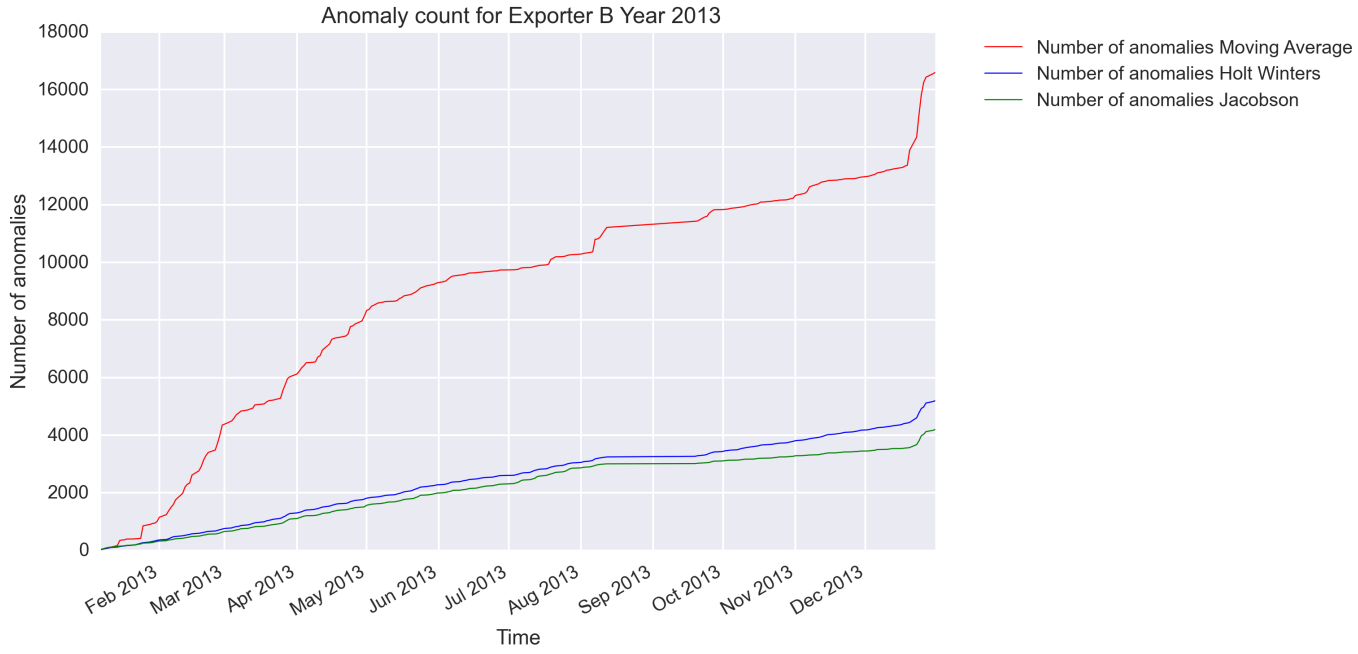


Figura 6.6: Anomalías a lo largo del año para el exportador B

Algunas de las visualizaciones se muestran en las figuras 6.8 y 6.9. Ésta última figura presenta un comportamiento peculiar que sólo se ha visto en este exportador: da la sensación de que se producen ataques periódicamente a lo largo de todo el día, y además se prolongan en los días siguientes. Esta anomalía será estudiada en la sección 6.2.1

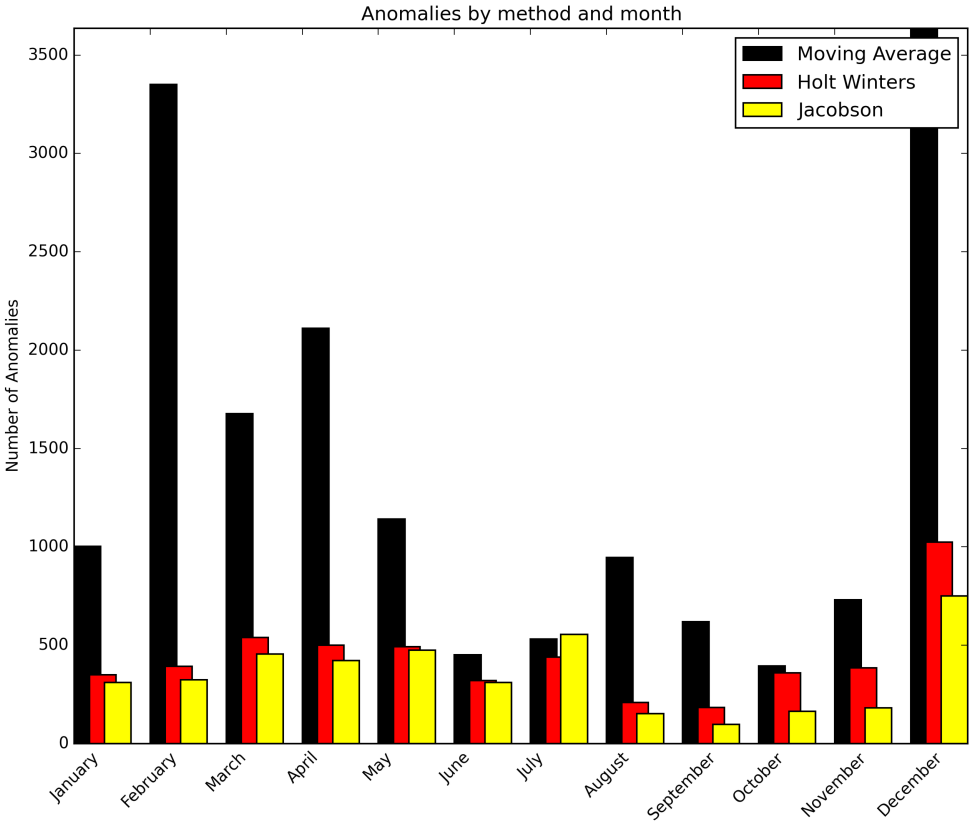


Figura 6.7: Anomalías por mes y método para el exportador B

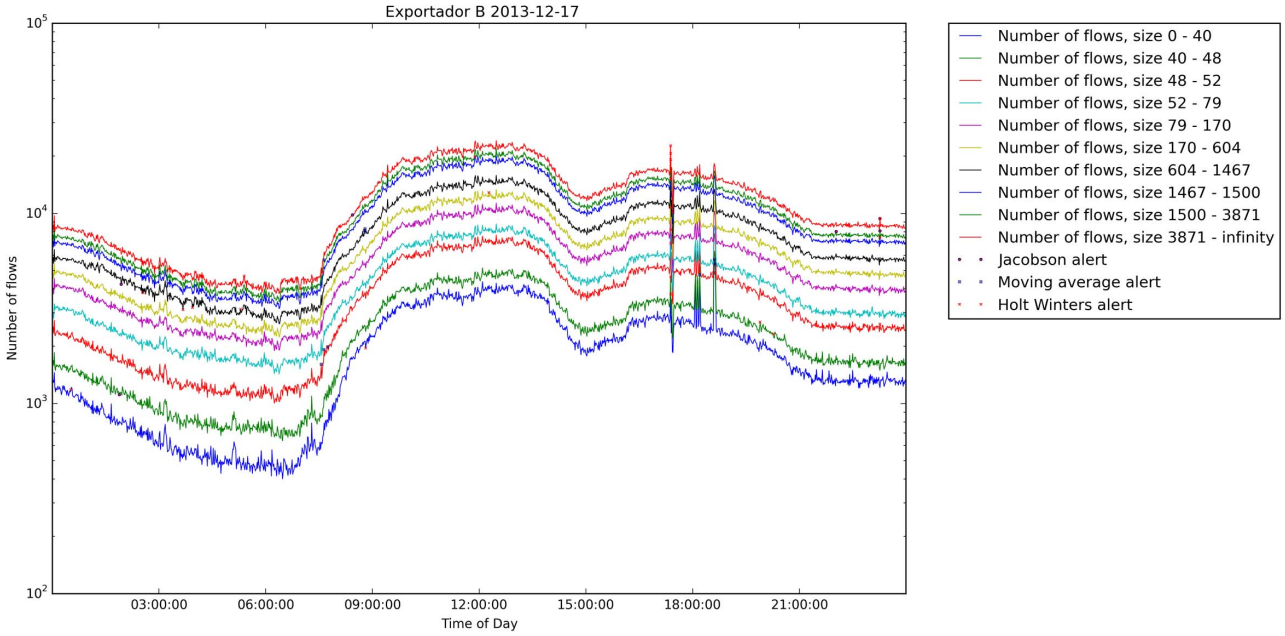


Figura 6.8: Visualización del 17/12 para el exportador B

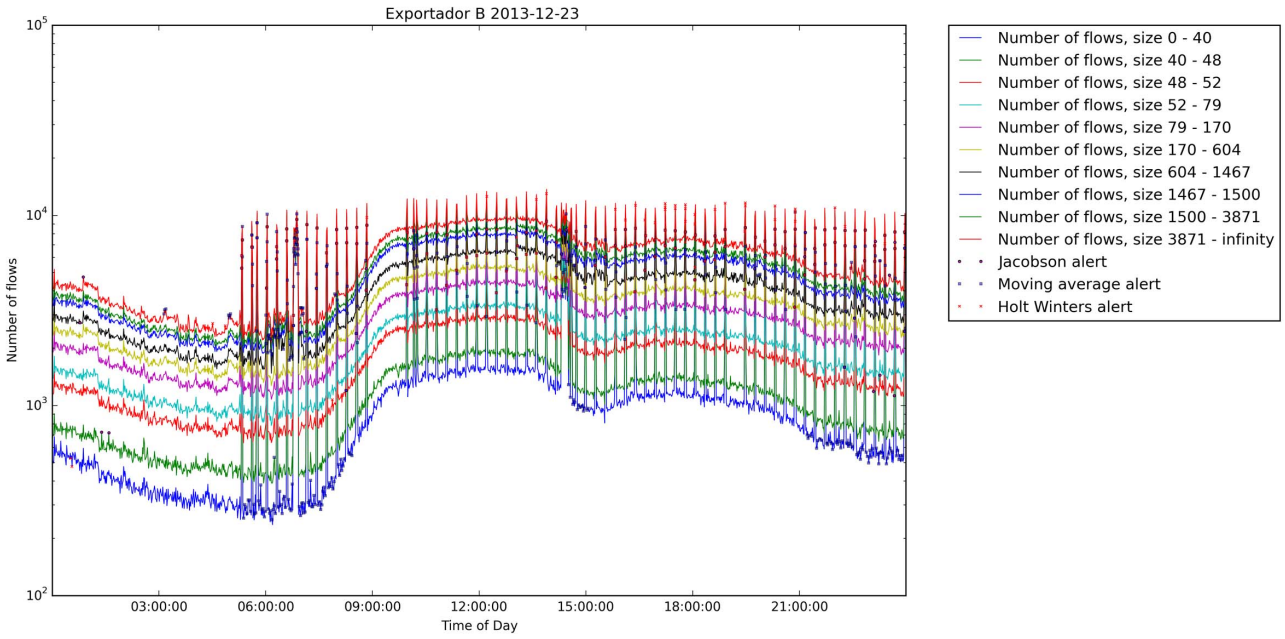


Figura 6.9: Visualización del 23/12 para el exportador B

6.1.3. Exportador C

Las anomalías durante el año y por mes se pueden ver en las figuras 6.10 y 6.11.

Tanto para el exportador C como para el B las gráficas se han hecho más suaves, en comparación con el exportador A.

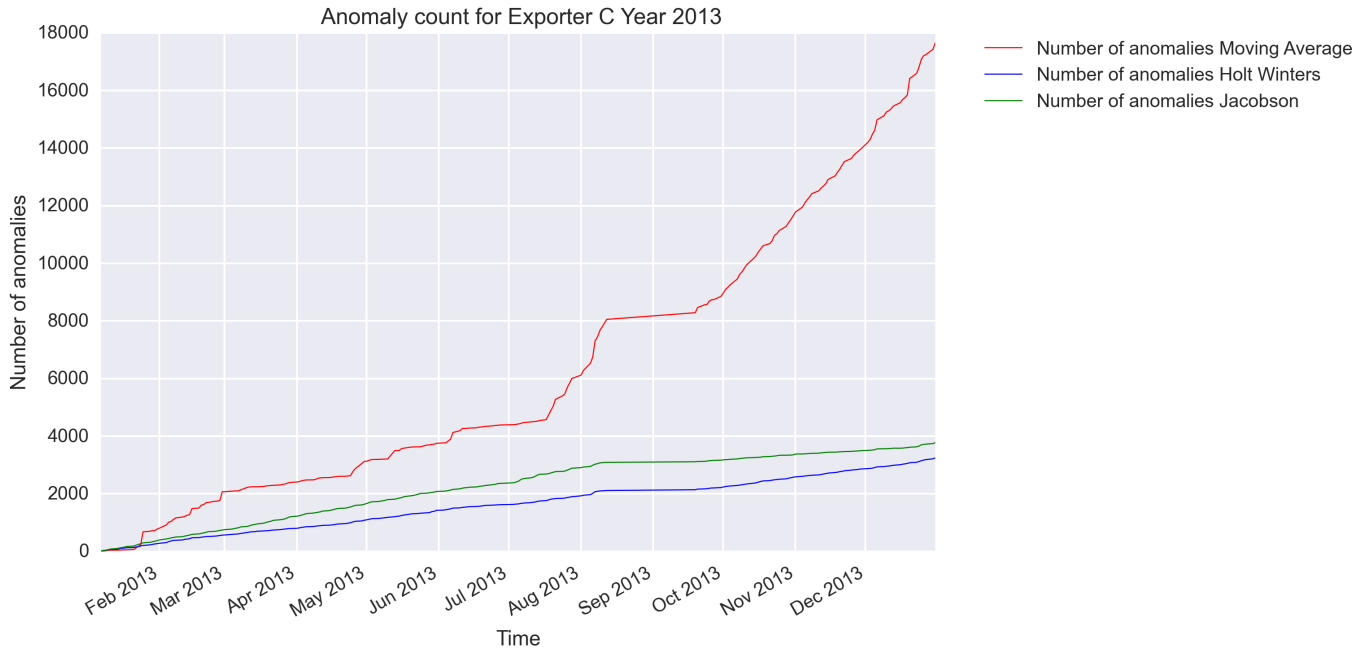


Figura 6.10: Anomalías a lo largo del año para el exportador C

En las figuras 6.12 y 6.13 se pueden ver dos de las visualizaciones de este exportador. En esta última figura se puede apreciar la diferencia entre una parada temporal del exportador y un vaciado (*flush*) de la caché de flujos. La primera es una parada temporal, mientras que la segunda es un vaciado debido a que se alcanza el tamaño máximo. En ambos casos veremos las bajadas al nivel de todos los deciles. De lo contrario sería algo que valdría la pena investigar en profundidad.

También se observa que en ambas figuras el método de la media móvil da muchas anomalías para el último intervalo de tamaños de flujo. Esto es así por la alta variabilidad de la cola pesada que estudiamos con anterioridad.

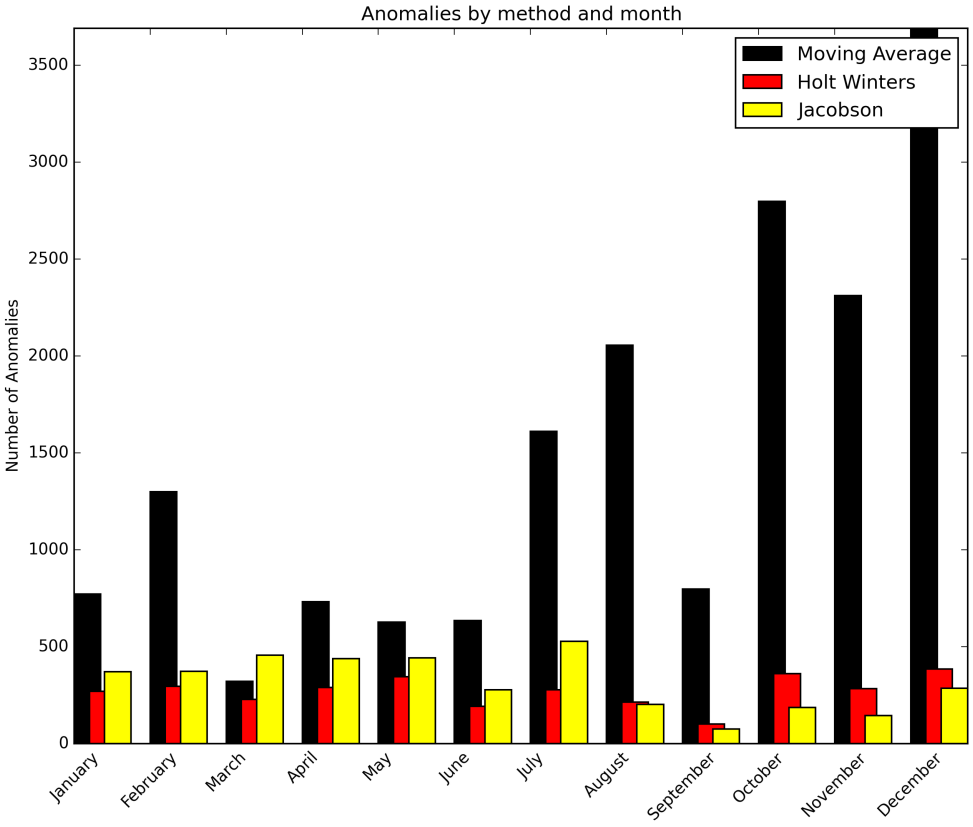


Figura 6.11: Anomalías por mes y método para el exportador C

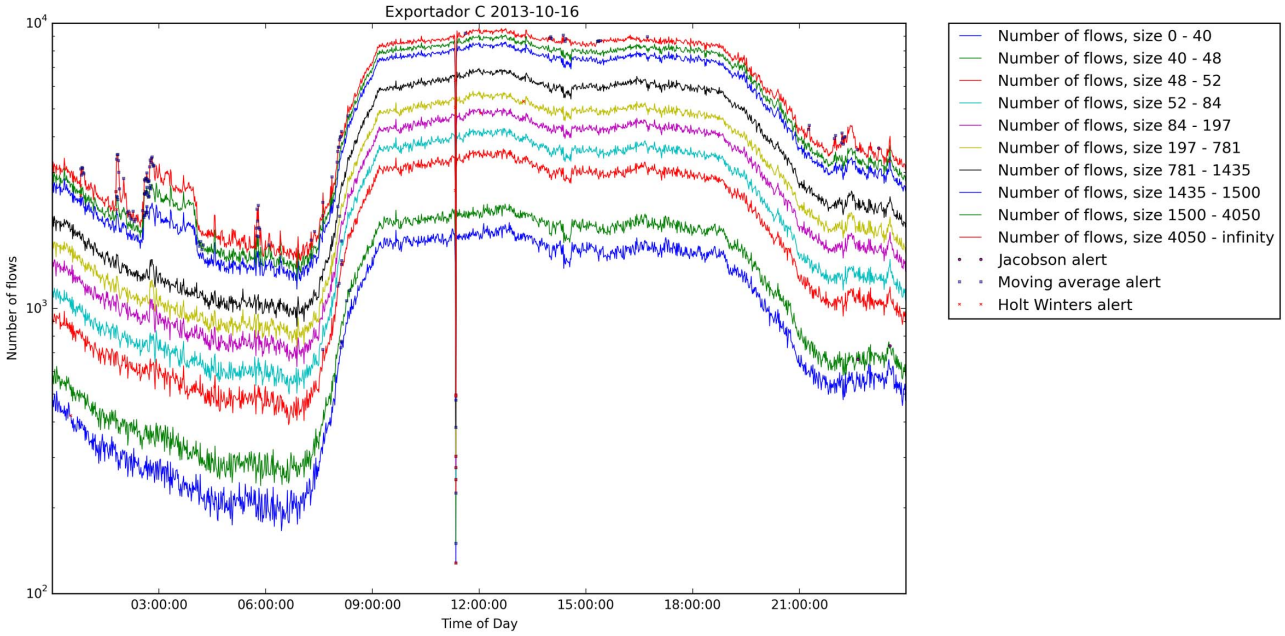


Figura 6.12: Visualización del 16/10 para el exportador C

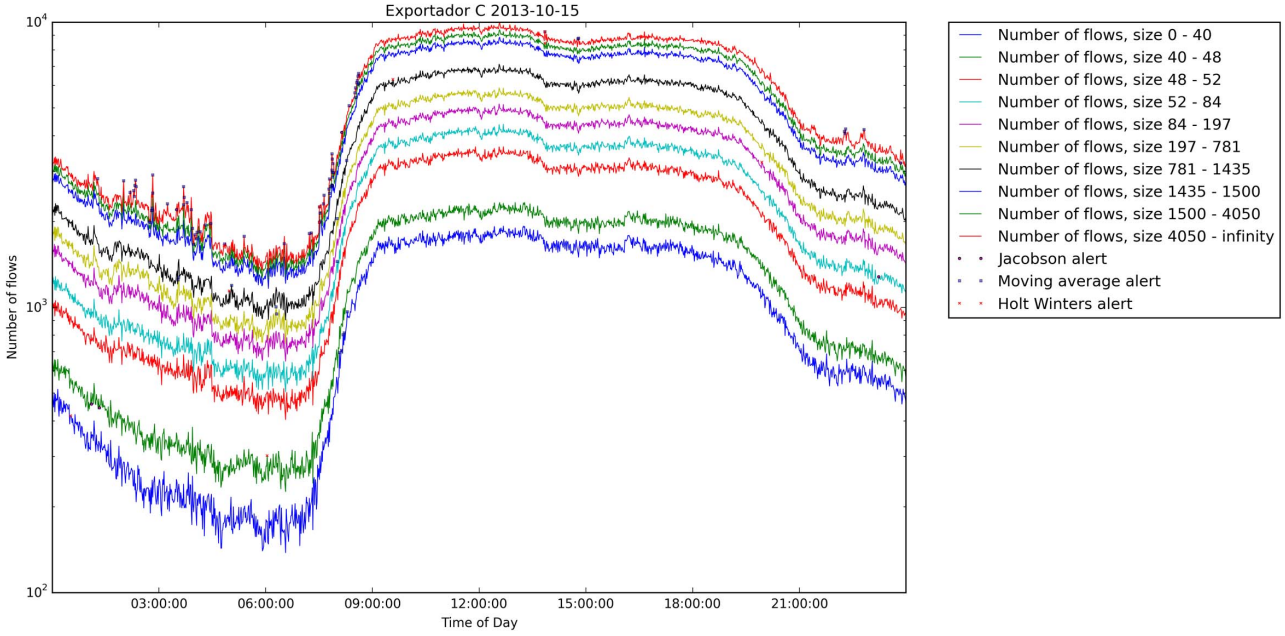


Figura 6.13: Visualización del 15/10 para el exportador C

6.1.4. Exportador D

En las figuras 6.14 y 6.15 se muestran las anomalías a lo largo del año y las anomalías por método y mes, respectivamente.

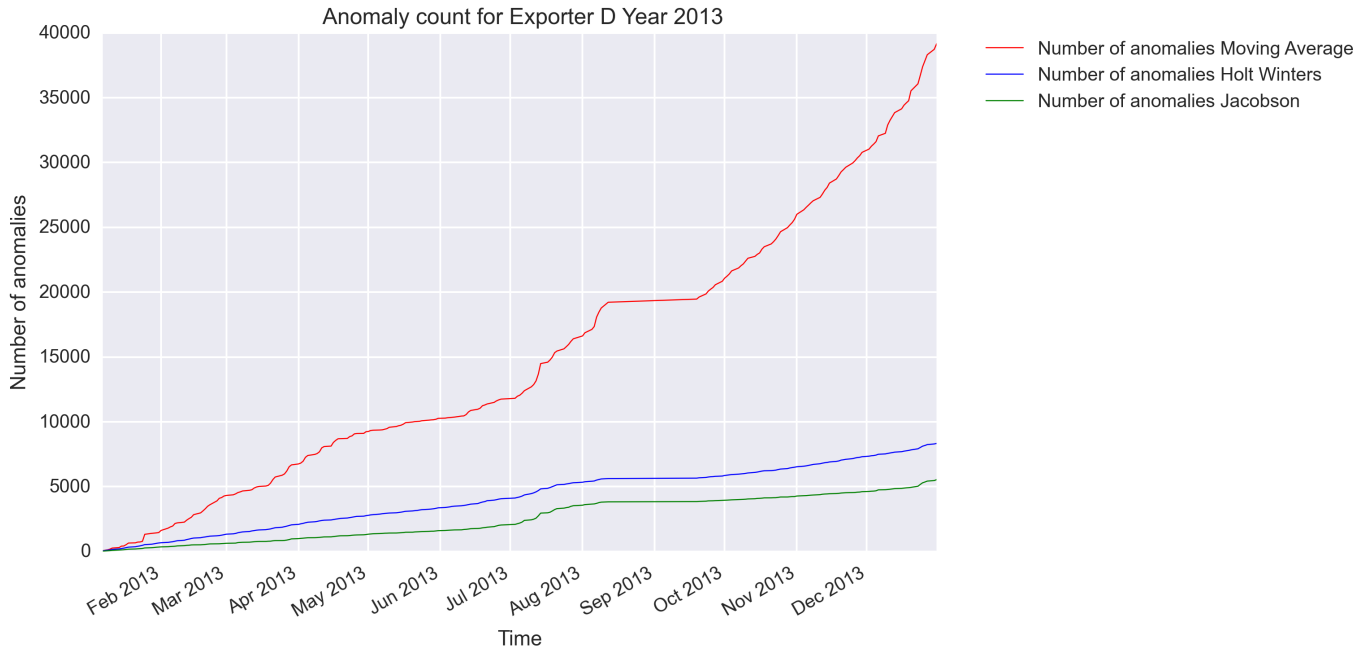


Figura 6.14: Anomalías a lo largo del año para el exportador D

Las gráficas elegidas para el exportador D son las figuras 6.16 y 6.17.
Como observación general este exportador ha presentado varias anomalías interesantes a lo largo del año en el intervalo de tamaño más bajo. Por ello estudiaremos la anomalía de la figura 6.17 en la sección 6.2.2.

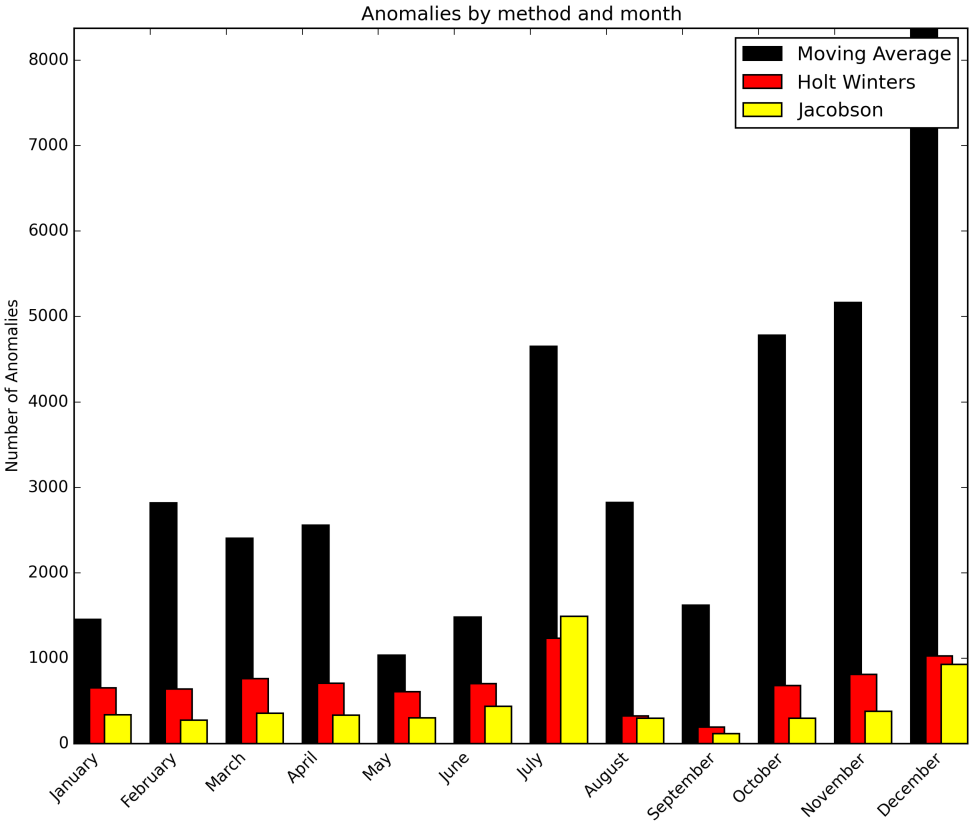


Figura 6.15: Anomalías por mes y método para el exportador D

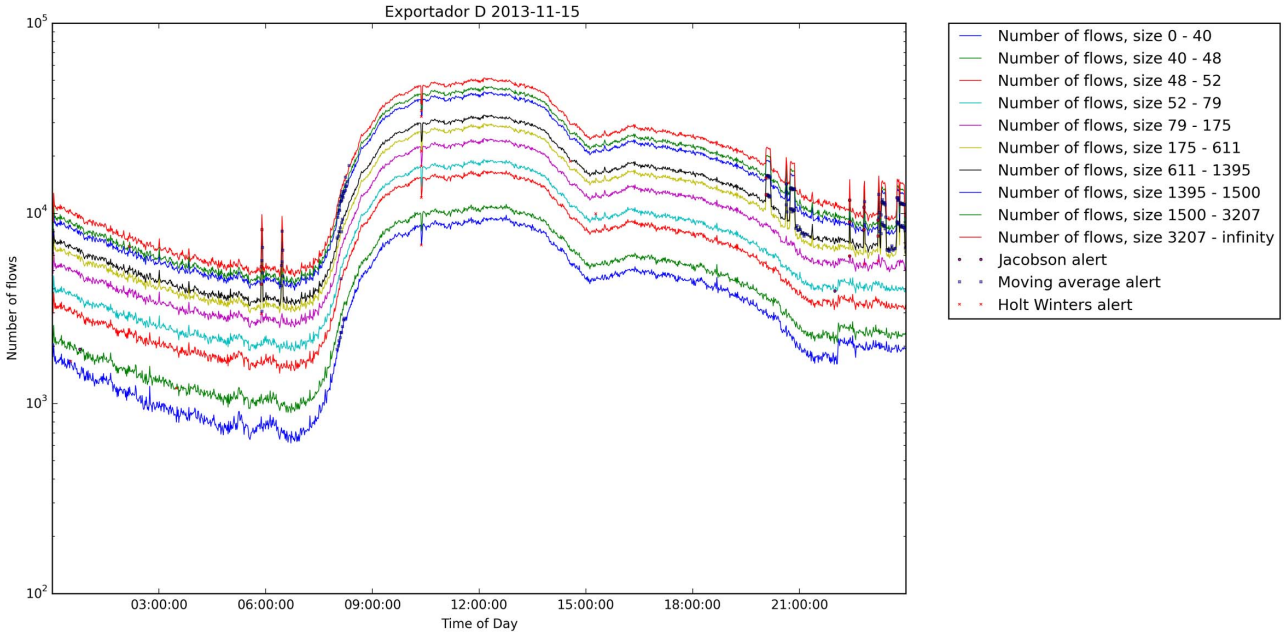


Figura 6.16: Visualización del 15/11 para el exportador D

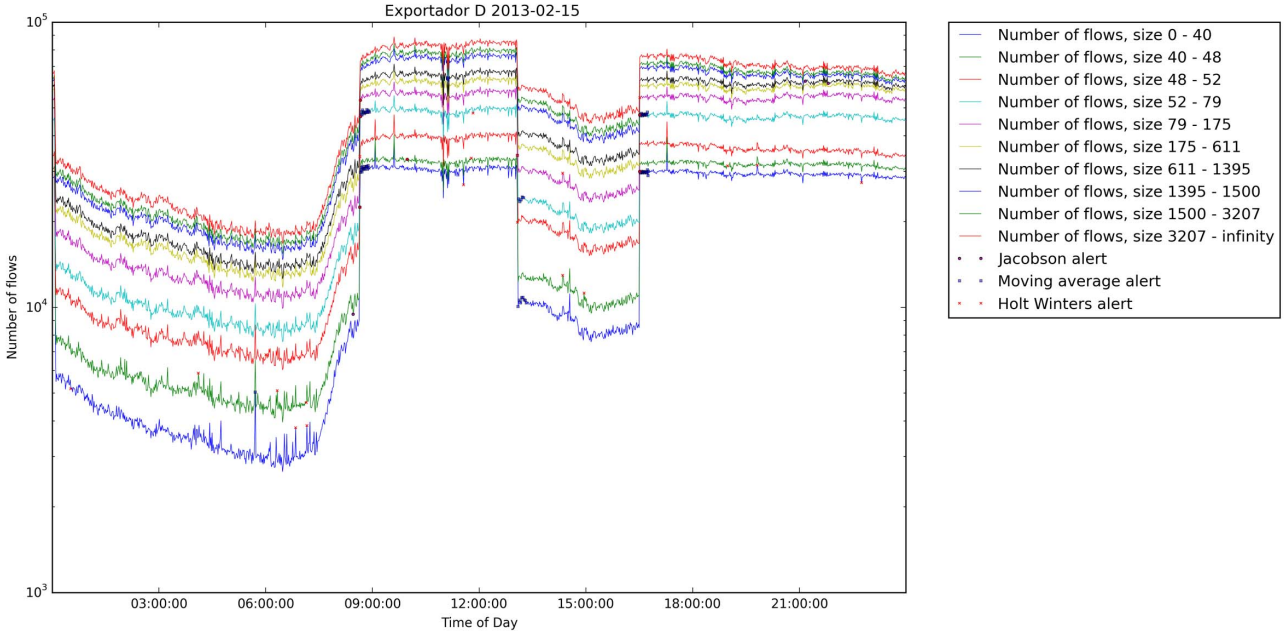


Figura 6.17: Visualización del 15/02 para el exportador D

6.1.5. Exportador E

En las figuras 6.18 y 6.19 podemos ver la evolución del número de anomalías a lo largo del año, así como el número de anomalías por tipo y mes respectivamente.

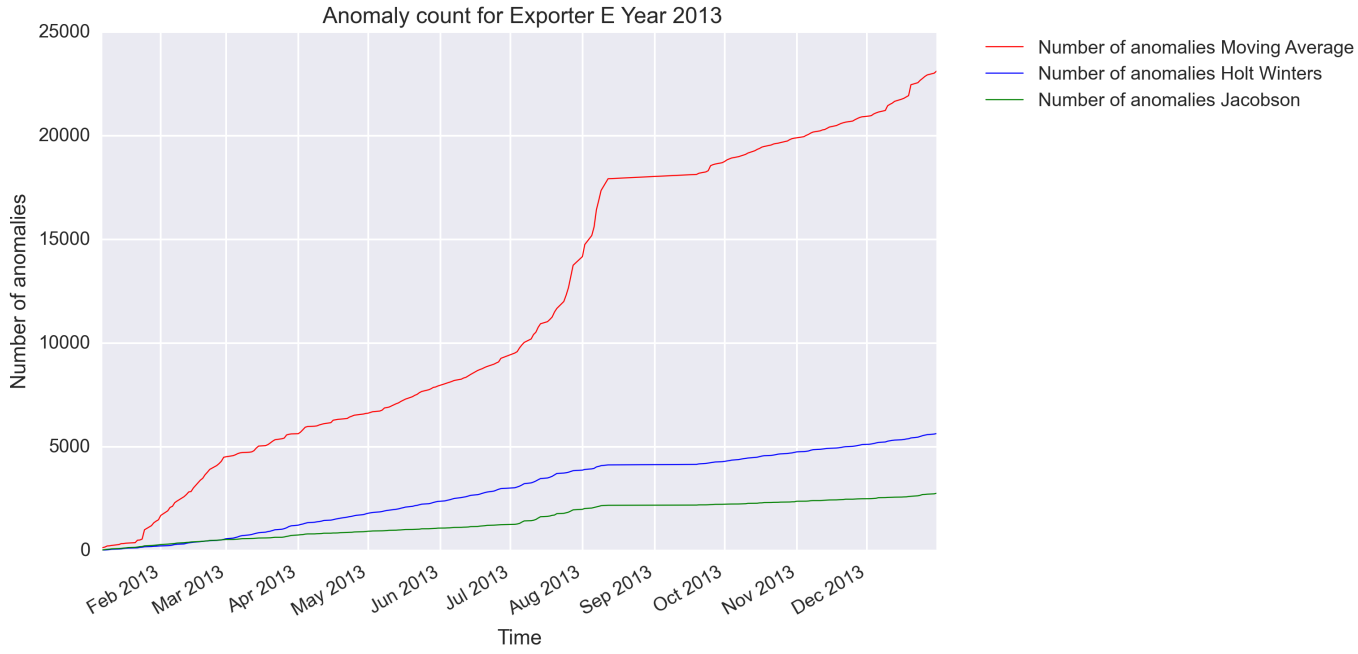


Figura 6.18: Anomalías a lo largo del año para el exportador E

Dos visualizaciones para este nodo se muestran en las figuras 6.20 y 6.21. En la primera se observa una anomalía que se mantiene durante casi una hora y la estudiaremos en la sección 6.2.3. En la segunda se puede observar una anomalía puntual cerca de las 06 : 00, que sólo salta en el segundo intervalo de tamaño, de manera que es fácilmente identificable visualmente.

6.2. Anomalías en profundidad

En esta sección se hará un breve análisis de tres anomalías presentadas en las figuras 6.9, 6.16 y 6.20. Para analizarlas se ha utilizado diversos filtrados con AWK. A fin de analizar en profundidad el fichero, y puesto que todo lo que se necesita son una serie de filtros sobre el fichero original) resulta más eficiente hacerlo con AWK.

6.2.1. Anomalía 1

Las anomalías de la figura 6.9 son interesantes al ser periódicas en el tiempo. Un análisis del intervalo de tamaño 0-40 bytes nos indica que los picos se deben a tráfico UDP entre dos hosts fijos, que usan puertos no conocidos y en cada ocasión usan uno nuevo. El tamaño de cada flujo es de 29 bytes. En un intervalo de 2 horas y media observamos casi 13000 puertos de destino alcanzados, lo cual nos indica que se han dado escaneos de puerto.

El patrón para cada pico observado es el siguiente: la misma dirección de IP de origen inicia muchos flujos UDP desde puertos distintos a una misma dirección de destino, contra el mismo

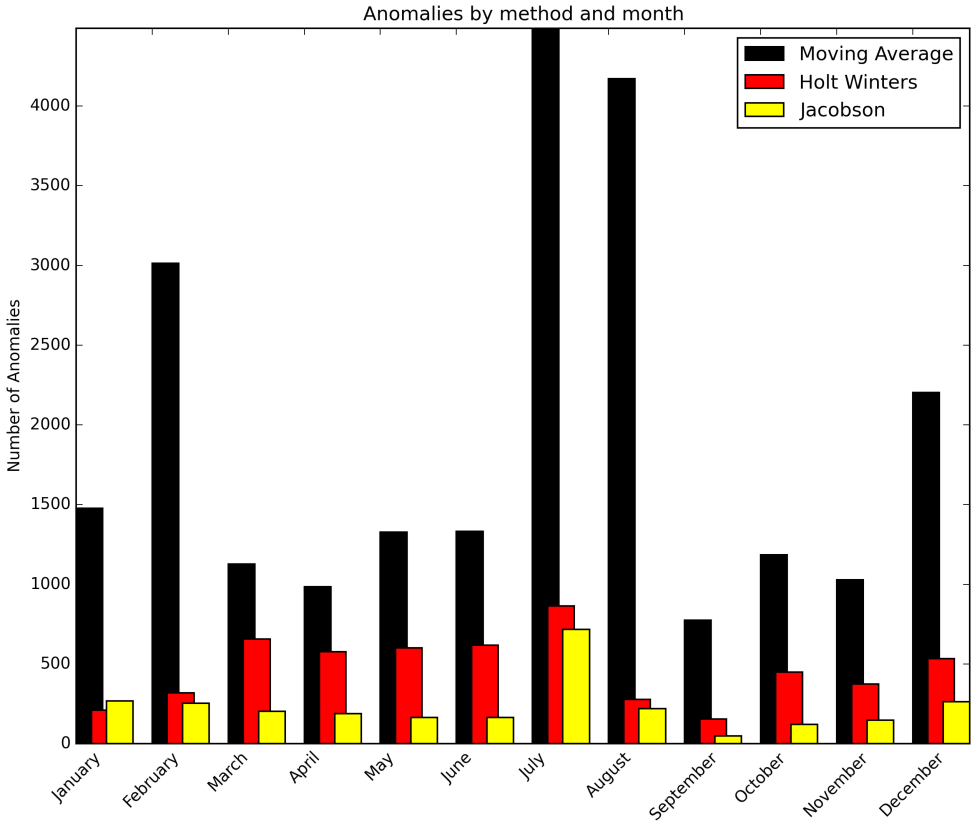


Figura 6.19: Anomalías por mes y método para el exportador E

puerto. Cada pico varía el puerto del destino contra el que acaban todos los flujos.

La dirección de destino resulta ser una ISP de Bucharest, Rumanía, y la dirección de origen interna a RedIRIS. Se plantean tres hipótesis posibles en este caso: bien se trata de un escaneo de puertos, bien se trata de algúna comunicación bot-maestro, o bien es una comunicación de un keylogger de un byte cada vez, lo cual sería raro. Un escaneo de puertos no puede ser, puesto que el puerto de destino es siempre el mismo para cada pico. En este caso,las únicas hipótesis que quedan son que se trate de una comunicación entre un bot y su maestro, o de un keylogger. La hipótesis del keylogger es improbable puesto que el envío de información no se hace byte a byte por UDP, por tanto debe de tratarse de una comunicación con un botmaster. Eso explicaría la periodicidad de las anomalías en la figura 6.9.

6.2.2. Anomalía 2

La anomalía de la figura 6.16 es quizá la más peculiar de las que estamos estudiando, por el largo período de tiempo que abarca. En concreto estudiaremos el aumento de tráfico en el primer intervalo de tamaño observado a partir de las 22:00. Un análisis minucioso muestra que se trata de tráfico UDP entre dos direcciones IP de origen y destino fijas. El puerto de origen es siempre el mismo, 60141, mientras que el de destino va variando entre números de puertos no conocidos. El tamaño de los paquetes es siempre 29 bytes. La IP de origen pertenece a una de las subredes de IP's de origen de las que se hablarán en la siguiente anomalía de estudio, interna a RedIRIS.

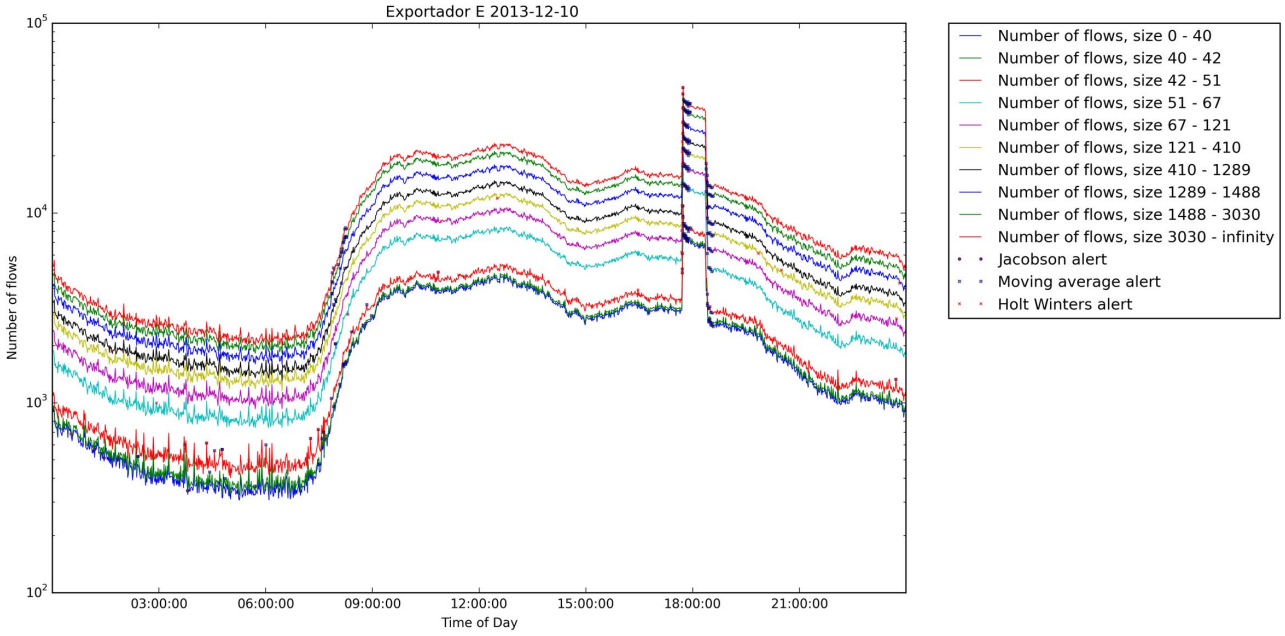


Figura 6.20: Visualización del 10/12 para el exportador E

Al tratarse de datagramas de 29 bytes, son UDP con un byte de carga. Al observar que los puertos de destino varían siempre entre números no conocidos, nos decantamos por que sea un escaneo de puertos masivo, vertical. Este resultado coincide con el analizado en [26], en el que sí que tenían acceso al contenido de los paquetes. La IP de destino no cambia, curiosamente, durante la duración de la anomalía.
Por tanto concluimos que lo que estamos observando es un escaneo de puertos desde una dirección de RedIRIS hacia una red de una universidad griega, según la información que proporciona el *whois* de RIPE.

6.2.3. Anomalía 3

El análisis del tráfico observado en la figura 6.20 entre las 18 y las 19 para el intervalo de 40-42 bytes de tamaño de flujo muestra que se han accedido 33728 puertos distintos en una hora, lo cual indica que ha habido escaneos de puertos. Un análisis de las direcciones IP muestra los orígenes de dichos escaneos, pero no se atribuyen a una única dirección sino a un pequeño grupo de direcciones. Los escaneos se muestran incompletos, pero son tanto longitudinales como transversales.

Por otro lado observamos un dato curioso: en una hora, se producen 103821 conexiones por el puerto 80 (HTTP), seguidas por un 56060 por el puerto 443 (HTTPS). Para poner en comparación estas cifras, la anomalía del exportador B estudiada en 5.1.2.1 presenta, en un intervalo de dos horas y con anomalías, tan sólo $\sim$ 60000 conexiones HTTP y 21000 HTTPS. Claramente sospechamos de un ataque de denegación de servicio o de intento de vulneración de algún exploit vía puerto 80.

Una observación más detallada de las direcciones IP origen que causan la mayor parte del tráfico HTTP muestra que son hosts de varias subredes españolas, tanto de origen como de

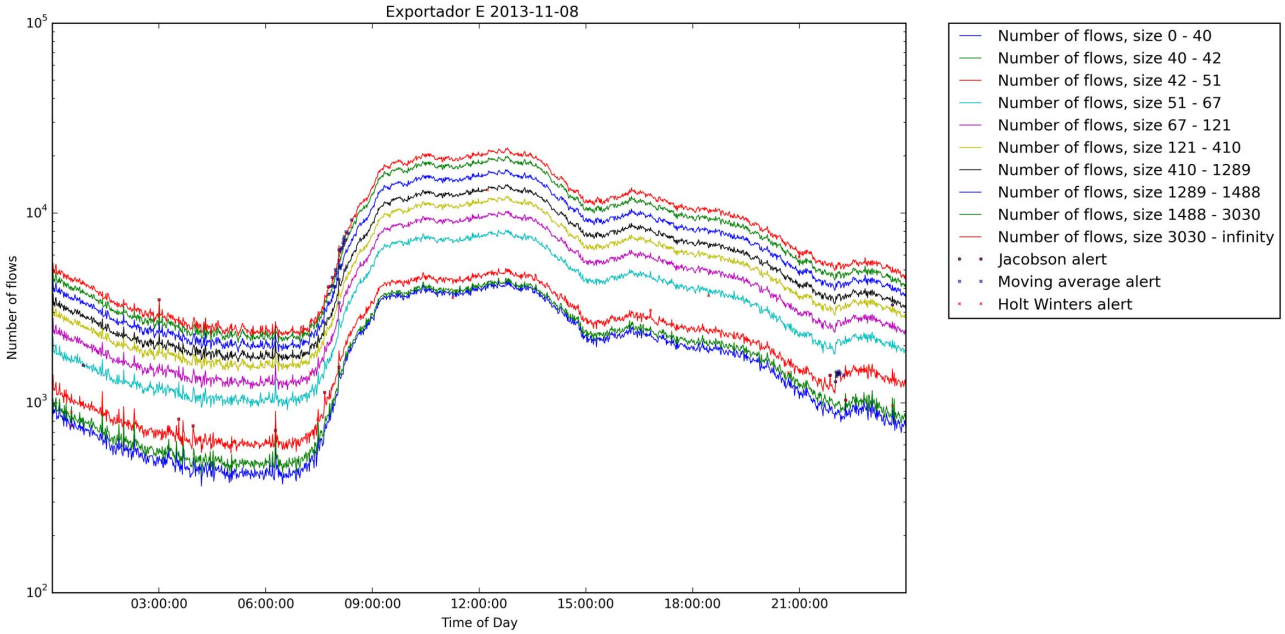


Figura 6.21: Visualización del 08/11 para el exportador E

destino. Varios grupos de direcciones IP de una mismas subredes generan mucho tráfico en el mismo período de tiempo. Los hosts de destino son muchas veces los mismos que los que causan el mayor tráfico en origen.
Por tanto, a falta de investigar el *payload* de los paquetes, de los que no disponemos al estar haciendo análisis de flujos, nos lleva a pensar en un DDoS o en propagación de algún gusano.
Al final no resulta ser lo que parece, pues investigando las direcciones que originan la mayoría del tráfico (son las mismas para los diferentes tamaños de flujo) nos damos cuenta de que se están comunicando entre sí varias universidades de España, y entre ellos, centros de supercomputación. Al ver la anomalía a todos los tamaños de flujo posibles, lo más probable es que se trate de un intercambio de datos entre universidades españolas y otras entidades conectadas con RedIRIS, y no realmente de un ataque. En este caso sería útil averiguar qué ocurrió realmente este día, si hubo algún evento científico, etc...

6.3. Conclusiones

En este capítulo se ha presentado un resumen de los resultados de ejecutar el programa frente a los datos de tráfico en 5 nodos de RedIRIS durante el año 2013. Se ha mostrado, en primer lugar, conclusiones generalizadas de los resultados, para acto seguido pormenorizar en los resultados de cada nodo. De cada uno de ellos se ha proporcionado un conteo de anomalías por mes y método, así como una evolución en el número total de anomalías detectadas a lo largo del año. Estos análisis nos han permitido descartar la media móvil como método válido de detección.

Por último, se han presentado tres casos de estudio particulares de anomalías detectadas, en los que hemos investigado las causas de las mismas a partir de las observación de la visualización del programa desarrollado. Así, utilizando AWK y diversos recursos adicionales hemos reconstruido los ataques o bien postulado varias hipótesis en los casos en los que la falta de mayor detalle en el conjunto de datos original nos impida saber las causas reales de la anomalía.

7

Conclusiones y trabajo futuro

En este capítulo se presentan las conclusiones del trabajo desarrollado a lo largo de este trabajo de fin de grado. Se presenta asimismo una breve sección de posibles mejoras y sugerencias para trabajo futuro

7.1. Conclusiones

El desarrollo de una herramienta como la propuesta en este trabajo es de utilidad por varias razones.

En primer lugar, permite obtener un mejor conocimiento de la red que se esté monitorizando. Los deciles del tamaño de los flujos en bytes dependen de la red observada, e incluso pueden variar en la misma red con el tiempo. A medida que pasa el tiempo el volumen en bytes de tráfico aumenta, lo que a priori se traduce en un tamaño medio de flujo mayor, que puede alterar los deciles de la distribución. Sin embargo esto puede corregirse recalibrando los deciles cada cierto período de tiempo, por ejemplo, seis meses. El estudio de la red también ayuda, por ejemplo, a comprobar la proporción entre *mice flows* y *elephant flows*. En este sentido se evita la arbitrariedad observada en los intervalos de tamaño estáticos de la herramienta *Time Series Solver*

Además, permite una rápida identificación de anomalías en el volumen de tráfico, medido en número de flujos en un instante dado. Al dividir por deciles de tamaño de flujo, es fácil para un administrador de red o un investigador forense identificar visualmente cuales pueden ser las causas de una anomalía en función de si se produce en un intervalo determinado, de si se produce en varios intervalos a la vez, etc...

El estudio de la red en función de, en el fondo, su función de distribución apunta en varias direcciones interesantes. Para empezar no es necesario conocerla, sino tan sólo obtener una muestra, en este caso los deciles, y trabajar a partir de ella. Este concepto es extrapolable a diferentes puntos de vista. Por ejemplo, podríamos haber obtenido los deciles de la duración en segundos de los flujos a lo largo de un día. De esta manera la detección de anomalías sería igual

salvo que lo que estaríamos observando serían hechos del tipo "entre las 12:00 y las 12:05 se observa un número excesivo de flujos de duración 0 ms", lo cual podría estar indicándonos un escaneo de puerto. Este enfoque, además, puede ser estudiado y explotado con usando herramientas más potentes que las utilizadas en este trabajo, como por ejemplo las presentadas en [25].

Los métodos de detección empleados son, de nuevo, mejorados respecto a la herramienta *Time Series Solver*, de manera que se utilizan mejores intervalos de confianza para la detección, que tienen en cuenta la desviación típica y la media de las observaciones anteriores. Aunque el método de la media móvil es descartable a la hora de analizar anomalías, los otros dos métodos resultan eficaces y se comportan mejor a la hora de efectuar la detección.

Aunque la visualización no permite un análisis en profundidad de las anomalías observadas, utilizando herramientas adicionales se pueden investigar las causas de una anomalía. Al no disponerse del tráfico a nivel de paquetes el análisis de la herramienta no puede ser más exhaustivo, pero el uso de herramientas adicionales como scripts de AWK permite obtener la información suficiente para entender las causas del comportamiento anómalo. En el peor de los casos se pueden establecer varias hipótesis, a falta de información adicional.

Por último, este trabajo ha servido de base para un artículo presentado al CNSM 2015 bajo el título "*Dictyogram: A statistical approach for the definition and visualization of network flow categories*"(David Muelas, Miguel Gordo, José L. García Dorado, Jorge E. López de Vergara), en el que se han estudiado diferentes métodos para aproximar los deciles de los exportadores y el efecto que ello produce en las observaciones: calculando la observación más profunda, la mediana, y la media (en el caso de este trabajo, los deciles han sido obtenidos con la media de todos los valores obtenidos).

7.2. Mejoras y Trabajo Futuro

En primer lugar la herramienta, aunque está pensada para su utilización como técnica forense, podría implementarse para su ejecución en tiempo real, siempre y cuando se disponga de suficientes datos para obtener los deciles de la distribución en el nodo pertinente. Una opción sería implementarlo como plugin de *softflowd*. De esta manera, se necesitaría una muestra relevante previa para poder obtener los deciles de la red bajo estudio. La herramienta iría actuando con un restraso preconfigurado con *softflowd* para enviar los archivos pcap al colector.

La herramienta presenta visualizaciones día a día, pero es sencillo modificarlo a fin de que se puedan dibujar datos semanales, por ejemplo. En este caso, la utilización del método de Holt Winters de tercer orden sería adecuado para efectuar la detección, como se presentó en la sección 3.2.4.1. En general, el programa se puede modificar para cualquier ventana de tiempo deseada, siempre y cuando los parámetros de los métodos se modifiquen de manera oportuna, especialmente los relativos al estimador de la desviación típica y de la media muestral.

El programa no proporciona interfaz gráfica de cara a un usuario final, pero una mejora podría consistir en su elaboración, a fin de que permitiera modificar los valores de los parámetros de los métodos de detección, así como selección de ficheros, visualización interactiva, etc. Por ejemplo, esta visualización interactiva podría consistir en que en cuanto el usuario seleccionara una anomalía, el programa automáticamente filtrara y presentara por pantalla los flujos activos en ese momento, permitiendo al usuario un filtrado manual para entender la anomalía en el menor tiempo posible.

El estudio de la función de densidad del tráfico también es un punto interesante de investigación adicional. En cierto sentido tomar deciles discretiza esta distribución, lo cual inevitablemente conlleva una cierta pérdida de información. El estudio de la misma con técnicas de análisis funcional puede aportar nuevos puntos de vista.

Por último, los métodos de detección presentados pueden ser sometidos a una verificación, comparando sus aciertos contra datos de tráfico completos versus un muestreo de los mismos. Puesto que RedIRIS no proporciona los datos completos (ni etiquetados) en este caso no ha sido posible realizarlo, y consideramos que también es un punto interesante verificar la fiabilidad de la detección bajo muestreo.

Bibliografía

[1] International Telecomunications Union. ICT eye statistics. *http://www.itu.int/net4/itu-d/icteye/*, Feb 2015.

[2] R. Sadre A. Sperotto, G. Vliek and A. Pras. ICT eye statistics. *Proc. of the 15th Open European Summer School and IFfP TC6.6 Workshop (EUNICE '09)*, pages 208–216, 2009.

[3] Michael Collins. Network security through data analysis. *Ed. O' Reilly*, pages 30–33, 2014.

[4] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks, vol. 31*, page 2435âĂŞ2463, 1999.

[5] B. Claise. Specification of the ip flow information export (ipfix) protocol for the exchange of ip traffic flow information. *RFC 5101 (Proposed Standard)*, page Available at http://www.ietf.org/rfc/rfc5101.txt, Jul 2008.

[6] IETF. Packet sampling (psamp) working group. *http://www.ietf.org/html.charters/psamp-charter.html*, Jul 2008.

[7] CERT. Computer emergency response team. *www.cert.org*, 2015.

[8] M. Roesch. Snort, intrusion detection system. *http://www.snort.org*, Jul. 2008.

[9] G. Schaffrath and B. Stiller. Conceptual integration of flow-based and packet-based network intrusion detection. *Proc. of 2nd International Conference on Autonomous Infrastructure, Management and Security (AIMS '08)*, pages 190–194, 2008.

[10] F. Valeur C. Kruegel and G. Vigna. Intrusion detection and correlation: Challenges and solutions. *Springer-Verlag Telos*, 2004.

[11] V. Igure and R. Williams. Taxonomies of attacks and vulnerabilities in computer systems. *Communications Surveys and Tutorials, IEEE, vol 10 no. 1*, pages 6–19, 2008.

[12] S. Hansman and R. Hunt. A taxonomy of network and computer attacks. *Computers and Security, vol 24, no.1*, pages 31–43, Feb 2005.

[13] Z. Li Y. Gao and Y. Chen. Towards a high-speed router based anomaly intrusion detection system. *http://conferences.sigcomm.org/sigcomm/2005/poster-121.pdf*, Aug. 2005.

[14] Y. Gao Z. Li and Y. Chen. A DOS resilient flow-level intrusion detection approach for high-speed networks. *Proc. of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS)*, page 39, 2006.

[15] Q. Zhao J. Xu and A. Kumar. Detection of super sources and destinations in high-speed networks: Algorithms, analysis and evaluation. *IEEE Journal on Selected Areas in Communications, vol. 24 no. 10*, pages 1840–1852, Oct 2006.

[16] M.S. Kim H.J. King S.H. Chung J.Hong. A flow-based method for abnormal network traffic detection. *Proc. of IEEE/IFIP Network Operations and Management Symposium (NOMS'04)*, page 599âĂŞ612, Apr. 2004.

[17] Z. Zhu G. Lu Y. Chen Z. Fu P. Roberts and K. Han. Botnet research survey. *32nd Annual IEEE International Computer Software and Applications*, pages 967–972, Aug 2008.

[18] D. H. A. Karasaridis B. Rexroad. Wide-scale botnet detection and characterization. *Proc.of the first conference on First Workshop on Hot Topics in Understanding Botnets (HotBots 07)*, pages 1–8, 2007.

[19] G. Gu R. Perdisci J. Zhang and W. Lee. Botminer: Clustering analysis of network traffic por protocol and structure independent botnet detection. *Proc. of 17th USENIX Security Symposium (USENIX Security '08)*, pages 139–154, June 2008.

[20] Jan Rejchrt, Tomas Jirsik, and Jan Vykopal. Time series solver. `http://www.muni.cz/ics/services/csirt/tools/tss`, 2013. Masarykova univerzita.

[21] Jan Rejchrt, Tomas Jirsik, and Jan Vykopal. Time series solver. `https://labs.ripe.net/Members/jan_rejchrt/network-anomaly-detection-2013-survey-evaluation`, 2013. Masarykova univerzita.

[22] Open Software. Softflowd. *http://www.mindrot.org/projects/softflowd/*, 2014.

[23] V. Jacobson. Congestion avoidance and control. *ACM SIGCOMM-88*, August 1988.

[24] C. C. Holt. Forecasting trends and seasonals by exponentially weighted averages. *Pittsburgh ONR memorandum no. 52*, 1957.

[25] D. Muelas, J. E. López de Vergara Méndez, and J.R. Berrendero. Functional data analysis: A step forward in network management. *Universidad Autónoma de Madrid*, 2014.

[26] Ignasi Paredes Oliva. Addressing practical challenges for anomaly detection in backbone networks. *Universidad Politécnica de Cataluña BarcelonaTech*, pages 11–12, June 2013.