



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

DISEÑO, INTEGRACIÓN Y EVALUACIÓN DE HERRAMIENTAS DE VISUALIZACIÓN DE TRÁFICO PARA EL RECONOCIMIENTO DE ATAQUES COMPUTACIONALES VÍA UNA DARKNET

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS MENCIÓN COMPUTACIÓN

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERA CIVIL EN COMPUTACIÓN

RENATA FACILONGO NOCE

PROFESOR GUÍA:
SR. ALEJANDRO HEVIA ANGULO

MIEMBROS DE LA COMISIÓN:
SR. JOSÉ MIGUEL PIQUER GARDNER
SR. ALEXANDRE BERGEL
SR. CARLOS MARTÍNEZ CAGNAZZO

SANTIAGO DE CHILE
SEPTIEMBRE 2012

Resumen

Una red tipo darknet es una porción de direcciones IPs asignado y ruteado en el cual no existen servicios activos ni servidores. Actualmente no existe un sistema, públicamente disponible, que permita la visualizar de forma gráfica los datos entregados por una red de estas características. El objetivo principal de esta tesis es la creación de un sistema dedicado a la visualización y análisis de datos de una darknet. El desarrollo de éste involucra la integración y evaluación de herramientas existentes además de la creación de un software enfocado en la creación de variados gráficos ad-hoc. Éstos permiten estudiar el comportamiento de la red con el fin de monitorearla para detectar posibles ataques.

La metodología se basó principalmente en la investigación de herramientas, análisis sobre sus resultados y el diseño e implementación de un software dedicado a la creación de gráficos. Para conseguir lo descrito, se realizó un estudio de representaciones gráficas útiles para un analista. Se estudiaron distintos formatos con que la darknet puede entregar la información sobre el tráfico que recibe. Se evaluaron herramientas ya existentes con el fin de determinar si éstas entregan información relevante en términos de seguridad, y se estudió si satisfacen las necesidades planteadas para así utilizarlas o, en caso contrario, modificarlas. Se diseñaron e implementaron nuevas soluciones para contar con información valiosa referente a la toma de decisiones a partir de gráficos. Para esto se estudió cómo manipular los datos fuente para extraerles la información relevante. Para el caso de que éstos crezcan mucho en el tiempo, se realizó un estudio consistente en cómo extender el sistema de gráficos para manejar grandes volúmenes de datos.

Se utilizaron 4 software de monitoreo de tráfico ya desarrollados: Moncube, TCPstat, EtherApe y Moncube, para generar imágenes sobre la actividad en la red. Estos gráficos son producidos diariamente a través de scripts que corren automáticamente cada día. El software diseñado y desarrollado para la creación de nuevas imágenes permite tener una visión completa de la red a través de 10 gráficos diarios diferentes. A través de un sitio web el analista podrá solicitar nuevos gráficos con los datos que él necesite, obteniendo una respuesta en tiempo razonable. En caso de que los datos arrojados por la darknet aumenten considerablemente se estudió el caso de bigdata, realizando un ejemplo gráfico cuyos datos fueron procesados en un ambiente cloud obteniendo respuestas valiosas y en corto tiempo.

Finalmente se obtuvo un sistema completo que, en opinión de los autores, otorga información variada e interesante para un analista en seguridad. El software fue diseñado para ser extensible sin grandes dificultades, paralelizable y aplicable a otras variantes de redes de tipo darknet lo que permitirá monitorearlas y reaccionar ante eventos de seguridad. En un futuro, y a la luz de la experiencia del gráfico procesado en la nube, todos los gráficos creados para este trabajo podrían ser procesados en la nube para conseguir respuestas aún más rápidas que las ya obtenidas y permitir el procesamiento de grandes volúmenes de información.

Agradecimientos

Gracias a mis padres Franco y Brunella, hermanos Loredana, Emanuel y Gianluca por hacer posible y acompañarme en esta carrera que ya finaliza. A Alejandro Hevia, director del CLCERT, gestor del presente tema de tesis y profesor guía; y a Sergio Miranda, Director de Tecnología del CLCERT, por toda la orientación, tiempo y consejos brindados. A mis amigos de la vida, pequeños genios que conocí en la Universidad y compañeros de Atakama Labs por el ánimo, empuje y ayuda entregada. A Ronald por todo su apoyo emocional y técnico en todo momento de esta travesía. Agradezco a los miembros de mi comisión, los profesores José Miguel Piquer, Alexandre Berguel y Carlos Martínez por los comentarios entregados dado que permitieron mejorar aún más mi trabajo.

Tabla de contenido

Resumen.....	2
Agradecimientos	3
1. Introducción	14
1. a. Motivación.....	16
2. Estado del arte y Antecedentes	16
2. a. Ataque de denegación de servicios (DoS):	18
2. b. Gusanos:.....	19
2. c. Escaneo Malicioso:	19
3. Definición del problema	20
3. a. Objetivo general	20
3. b. Objetivos Específicos.....	20
3. c. Metodología	21
3. d. Restricciones y recursos:	22
4. Discusión: Solución Propuesta	22
a. AfterGlow:	22
b. TCPstat:	22
c. EtherApe:	23
d. Moncube:	23
i. Accesos por puerto:.....	24
ii. Accesos por puerto y protocolo:	24
iii. Accesos a puertos en el tiempo:.....	25
iv. Accesos por protocolo:	26
v. Accesos por protocolo y dirección IP externa:	27

vi. Accesos de direcciones IP externas en el tiempo:.....	27
vii. Accesos a direcciones IP de la darknet en el tiempo:.....	28
viii. Accesos por protocolo y dirección IP de la darknet:	29
ix. Gráficos de estado: 10 puertos más accesados por protocolo y los 10 países que realizan más accesos a la darknet:	30
4. a. Utilización de herramientas existentes:	31
4. a.1. AfterGlow:	32
4. a.2. Moncube:	37
4. a.3. TCPstat:.....	41
4. a.4. EtherApe:.....	47
4. a.5. Copiado el material al servidor web y borrado del antiguo:.....	49
4. b. Implementación de herramienta para la creación de gráficos ad-hoc	49
4. b.0. Captura de pcap de ayer:	49
4. b.1. Listas de IPs y puertos:	49
4. b.2. Gráficos por omisión:	61
a) limpia(archivo_entrada, archivo_salida):.....	70
b) limpiatime(archivo_entrada, archivo_salida):	72
c) limpiaIP(archivo_entrada, archivo_salida):.....	73
4. b.3. Alertas	82
5. Despliegue de gráficos en sitio web:.....	86
5.a. Selección de tipo de gráfico:	86
5.b. Selección de fecha:.....	87
5.c. Despliegue de gráficos AfterGlow, Moncube, EtherApe, TCPstat:.....	87
5.d. Despliegue de gráficos pre-calculados (por omisión):	88
6. Creación de gráficos a pedido	90
6. a. Formularios:	90

6. b. Generación de los gráficos:	93
6. c. Envío y despliegue de archivos XMLs:	95
7. Ejemplo realizado en Big Data.....	97
8. Detección de patrones y ataques:.....	105
9. Conclusiones y Aportes	121
10. Datos estadísticos y futuras líneas de investigación	125
10. a. Pruebas y datos estadísticos:	125
10. b. Futuras líneas de investigación:	127
11. Referencias.....	129
11. Anexos.....	134
A. Código fuente del script <i>afterglowMin.sh</i>	134
B. Código fuente del script <i>afterglow.sh</i>	134
C. Código fuente del script <i>moncubeMin.sh</i>	135
D. Código fuente de script <i>moncube.sh</i>	135
E. Código fuente del script <i>TCPstats_protocolMin.sh</i>	136
F. Código fuente del script <i>TCPstats_protocol.sh</i>	136
G. Código fuente del script <i>gnuplot.script</i>	137
H. Código fuente del script <i>TCPstats_SYNACKMin.sh</i>	137
I. Código fuente del script <i>TCPstats_SYNACKMin.sh</i>	138
J. Código fuente del script <i>gnuplotACK.script</i>	138
K. Código fuente del script <i>ether.sh</i>	139
L. Código fuente del script <i>etherMin.sh</i>	139
M. Código del script <i>tiraListas.sh</i>	140
N. Código del script <i>generadorListas.sh</i>	141
O. Código del script <i>graficosFlash.sh</i>	142

P.	Código del método <i>printListaIP</i>	146
Q.	Código del método <i>printListaIP</i>	146
R.	Código del script <i>generador.sh</i>	146
S.	Código del script <i>generadorTime.sh</i>	148
T.	Código del script <i>generadorRing.sh</i>	149
U.	Tabla de llamadas python	150
V.	Código del método <i>limpia</i>	152
W.	Código del método <i>limpiaUDP</i>	153
X.	Código del método <i>limpiaTCP</i>	154
Y.	Código del método <i>limpiatime</i>	155
Z.	Código del método <i>limpiaIP</i>	156
AA.	Código del método <i>limpiaAtaIP</i>	156
BB.	Ejemplos de XMLs por función asociada a tipo de gráfico.....	157
CC.	Código del script <i>tiraAlerta.sh</i>	165
DD.	Código del script <i>generadorAlertas.sh</i>	165
EE.	Código del script <i>buscador.sh</i>	166
FF.	Código del método <i>nadaUDP</i>	167
GG.	Código del método <i>nadaTCP</i>	168
HH.	Código del método <i>listaPuertosUDP</i>	168
II.	Código del método <i>listaPuertosTCP</i>	169
JJ.	Código del método <i>listaAtaIP</i>	169
KK.	Código del método <i>listaAtaIP(map)</i>	169
LL.	Código del script <i>países.sh</i>	170

Índice de Figuras

Figura 1: Funcionamiento del Ataque de Denegación de Servicios.....	19
Figura 2: Accesos por puerto. Maani Charts.....	24
Figura 3: Accesos por puerto y protocolo.....	25
Figura 4: Accesos a puertos en el tiempo.	26
Figura 5: Accesos por protocolo.....	27
Figura 6: Accesos por protocolo y dirección IP externa.....	27
Figura 7: Accesos de direcciones IP externas en el tiempo..	28
Figura 8: Accesos de direcciones IP de la darknet en el tiempo..	29
Figura 9: Accesos por protocolo y dirección IP de la darknet.	29
Figura 10: Gráficos de estado: 10 puertos más accesados según protocolo.....	30
Figura 11: Gráfico de estado: 10 países que realizan mayor cantidad de accesos a la darknet.....	30
Figura 12: Trozo de tráfico de 16:30 a 16:40 de un determinado día.	33
Figura 13: Trozo de tráfico de 16:30 a 16:50 de un determinado día.....	34
Figura 14: Trozo de tráfico de 16:30 a 16:59:59 de un determinado día..	35
Figura 15: Cubo en primera posición.	38
Figura 16: Cubo en segunda posición.....	39
Figura 17: Cubo en tercera posición..	40
Figura 18: Paquetes por protocolo durante una hora.	42
Figura 19: Paquetes por protocolo durante un día.....	43
Figura 20: Paquetes totales durante una hora con tráfico SYN-ACK ficticio.....	43
Figura 21: Paquetes totales durante un día.	44
Figura 22: Captura de pantalla de EtherApe en ejecución.....	47
Figura 23: Formulario de alertas.	82
Figura 24: Ejemplo de lista de alertas.	85

Figura 25: Interfaz para seleccionar gráfico a ver (generados a partir de herramientas existentes).	86
Figura 26: Interfaz para seleccionar gráfico a ver (generados a partir de XMLs).	86
Figura 27: Interfaz para ver gráficos asociados al estado de la darknet.	86
Figura 28: Calendario sitio web (herramientas existentes).	87
Figura 29: Calendario sitio web (gráficos a partir de archivos XML).....	89
Figura 30: Formulario para crear gráficos.....	90
Figura 31: Formulario para crear gráficos (IPs en el tiempo).....	91
Figura 32: Formulario para crear gráficos (Puertos en el tiempo).....	92
Figura 33: Mensaje en caso de no haber datos.	96
Figura 34: Mensaje en caso error. Maani Charts.	96
Figura 35: Interfaz para seleccionar ataque a monitorear.....	105
Figura 36: Imagen relacionada con ataque DoS (timeIP).....	107
Figura 37: Imagen relacionada con ataque DoS (TCPstat).....	107
Figura 38: Imagen relacionada con ataque de DoS (column).....	108
Figura 39: Imagen relacionada con ataque de DoS (columnAta).....	108
Figura 40: Imagen relacionada con ataque de DoS (columnAta).....	109
Figura 41: Imagen relacionada con ataque de DoS a través de servidores DNS (pieProt).....	109
Figura 42: Imagen relacionada con ataque de DoS con protocolo UDP (columnAta).....	110
Figura 43: Imagen relacionada con ataque de escaneo de IPs desde una IP (AfterGlow).....	111
Figura 44: Imagen relacionada con ataque de escaneo de IPs (columnAta).....	112
Figura 45: Imagen relacionada con ataque de escaneo de IPs desde varias IPs (AfterGlow).....	113
Figura 46: Imagen relacionada con ataque de escaneo de IPs desde varias IPs a grupos (AfterGlow).....	114
Figura 47: Imagen relacionada con ataque de escaneo de IPs a un grupo de IPs de la darknet.....	115
Figura 48: Imagen relacionada con ataque de escaneo de IPs a un grupo de IPs de la darknet.....	115
Figura 49: Imagen relacionada con ataque de DoS a otros (TCPstat).....	116

Figura 50: Imagen relacionada con ataque de DoS a otros (column).....	117
Figura 51: Imagen relacionada con el escaneo de puertos (time).....	118
Figura 52: Imagen relacionada con el escaneo de puertos (pie).. ..	119
Figura 53: Imagen relacionada con el escaneo de puertos (rect).....	119
Figura 54: Imagen relacionada con todos los ataques (Moncube).....	120
Figura 55: Peso de archivos de tráfico en el tiempo.....	126

Índice de Tablas

Tabla 1: Ejemplo de tráfico.	52
Tabla 2: Códigos scripts de listaIP.py y listaPuertos.py.....	53
Tabla 3: Códigos de los métodos listaIP() y listaPuertos().	54
Tabla 4: Códigos del método mapGroup.	56
Tabla 5: Ejemplo de llamada a mapGroup.	56
Tabla 6: Definición de listaIP.	56
Tabla 7: Ejemplo de tráfico.	56
Tabla 8: Definición del método mycount.	57
Tabla 9: Ejemplo de llamada a mapGroup.	57
Tabla 10: Ejemplo de tráfico.	57
Tabla 11: Código del método listaPu.....	57
Tabla 12: Ejemplo de llamada a reduce.	58
Tabla 13: Código el método reduce.	58
Tabla 14: Código el método sum.....	59
Tabla 15: Ejemplos de IP.txt y Puertos.txt.	59
Tabla 16: Gráficos, sus categorías y script generador.....	64
Tabla 17: Ejemplo de tráfico.	66
Tabla 18: Ejemplo de tráfico.	67
Tabla 19: Código de para la generación del gráfico “Accesos por puerto”	70
Tabla 20: Ejemplo de tráfico.	71
Tabla 21: Ejemplo de tráfico.	71
Tabla 22: Ejemplo de tráfico.	72
Tabla 23: Ejemplo de tráfico.	73
Tabla 24: Funciones defval y func por método creador de gráfico.....	76

Tabla 25: Función fx por método creador de gráfico.....	78
Tabla 26: Ejemplos de resultados de map y reduce.....	80
Tabla 27: Ejemplo de XML resultante para la creación del gráfico “Accesos por Puerto”.	81
Tabla 28: Códigos de alertaProtocolo.py y alertaPuerto.py.	83
Tabla 29: Métodos defval y func para cada tipo de alerta.....	83
Tabla 30: Ejemplos de alertaProtocolo.txt y alertaPuerto.txt.	84
Tabla 31: Ejemplos de llamadas a la creación de gráficos desde el sitio web.	94
Tabla 32: Código del script lanzar.	94
Tabla 33: Ejemplo de archivo para el día 2012-01-07.....	97
Tabla 34: Ejemplo de archivo para el mes 01.	98
Tabla 35: Código de wordSplitter.py.....	98
Tabla 36: Ejemplo de accesos por día.	98
Tabla 37: Llamada para la descarga de datos.	102
Tabla 38: Detalles de Instancia pequeña.	102
Tabla 39: Detalles de Instancia grande.	103

Índice de Diagramas

Diagrama 1: Diseño de la darknet del CLCERT.	18
Diagrama 2: Representación del funcionamiento del sistema..	32
Diagrama 3: Representación del proceso de creación de listas finales de IPs y puertos..	51
Diagrama 4: Representación del proceso de generación de listas de IPs y puertos.....	60
Diagrama 5: Representación del proceso de creación de XMLs..	65
Diagrama 6 : Representación del proceso de los generadores de gráficos..	68
Diagrama 7: Representación del sistema de alertas.....	85
Diagrama 8: Representación del sistema de creación de gráficos ad-hoc.....	96
Diagrama 9: Sistema de procesamiento de datos de la darknet en la nube..	104

1. Introducción

La masificación del uso de tecnologías computacionales y el alcance global de Internet ha permitido que éstas sean utilizadas con fines no necesariamente benignos. Cuantiosos daños a sistemas computacionales se producen debido a ataques frecuentemente automatizados por parte de criminales o atacantes maliciosos que buscan lucrar mediante éstos o simplemente causar molestias [MI11, SY11].

La mayoría de los sistemas actuales almacenan registros (o *logs*) de los eventos que experimentan (mensajes recibidos, conexiones, etc.), de manera tal que si ocurre algún error, problema, o cyberataque, éste pueda ser rastreado y algún seguimiento pueda ser realizado. Es así como registros de fuentes tales como dispositivos de red o monitores de transacciones de sistemas financieros pueden ayudar a resolver problemas o a evaluar la red en términos de seguridad. Lamentablemente estos registros en la práctica son complejos de revisar y leer dado su formato (binario y/o dependiente de la aplicación) y extensión (típicamente archivos de texto de varios megabytes de largo). Por lo mismo, es deseable contar con material que permita comunicar de manera más intuitiva y natural - aunque fidedigna - la información capturada en dichos registros [HB11].

“Una imagen vale más que mil palabras”, dice un antiguo proverbio popular capturando la intuición de que una imagen puede comunicar información mucho más eficientemente que una descripción escrita, por ejemplo. En seguridad computacional, tal dicho quizás debiera formularse como “una imagen vale más que mil registros de logs”, según expertos en el tema [RM10]. En este trabajo de tesis se busca utilizar las ventajas de la visualización gráfica de eventos de red para tomar decisiones asociadas a seguridad computacional.

Visualización, en términos de seguridad, es el proceso de generar una imagen basado en registros de logs de un determinado sistema. La justificación básica para este enfoque es que el sistema visual del ser humano está desarrollado en lo que a la detección de patrones visuales respecta y se pueden sacar grandes ventajas de este aspecto [W04]. Por ejemplo, la visualización gráfica de grandes volúmenes de información permite detectar patrones y tendencias fácilmente, disminuyendo el riesgo que ellas sean obviadas al simplemente revisar visualmente registros textuales.

Más específicamente, la representación visual de datos permite comunicar grandes volúmenes de información [RM10]. Frecuentemente el cerebro humano tiene dificultad para procesar grandes volúmenes de información desplegados como texto. En cambio, las imágenes aparecen como alternativas mucho más eficientes para transmitir mucha información a los humanos, dada la facilidad con que nuestro cerebro comprende e internalizada información gráfica [W04].

La visualización de registros (logs) trae múltiples beneficios basados en la capacidad de las personas de procesar imágenes eficientemente. Entre ellos:

- a) La visualización permite crear imágenes que puedan ser interpretadas por humanos para responder dudas particulares de seguridad (por ejemplo, distribución temporal o geográfica de ataques).
- b) Explorar y descubrir: diferentes gráficos provenientes de una misma fuente de datos permiten detectar patrones o tendencias que, a la postre, permiten obtener información desconocida anteriormente.

c) Apoyar decisiones eficientemente: la visualización permite analizar grandes volúmenes de datos rápidamente y tomar decisiones con fundamentos sólidos (como es necesario, por ejemplo, en el campo de la minería de datos).

d) Comunicar información: las imágenes son de gran utilidad al momento de querer transmitir un mensaje. Por ejemplo, la utilización de representaciones gráficas de los problemas de seguridad encontrados por parte de un analista puede significativamente hacer más convincente su argumento de aumentar los recursos del área.

El tema de visualización para seguridad tiene relativamente poca historia, comenzando con los primeros estudios a fines de los 80's [CS89]. En las últimas décadas, el crecimiento explosivo de la cantidad de información existente que necesita ser analizada para detectar problemas de seguridad [MI11, SY11] ha motivado la necesidad de investigar esta área, buscando crear nuevas herramientas que puedan manejar eficientemente dicha información.

En la actualidad, las herramientas clásicas de seguridad tales como firewalls y sistemas de detección de intrusos tienen la capacidad de generar reportes que usan gráficos e imágenes. Estas herramientas son muy específicas y muestran solamente la información asociada al tipo de datos provistos por el firewall, usualmente sin permitir configuración para considerar otras fuentes de datos más genéricas.

En este trabajo de tesis la fuente de datos será el conjunto de registros obtenidos de una red de especiales características denominada darknet. Una darknet es una porción o espacio de direcciones IPs asignado y ruteado en el cual no existen servicios activos ni servidores.

El concepto de darknet fue propuesto por Stefan Savage y su equipo de la Universidad de California, San Diego [BCJPR05, MSBV06]. Se refiere a ella como un espacio de direcciones IP que únicamente recibe paquetes desde el exterior, los cuales son analizados con el fin de detectar, en particular, ataques de denegación de servicios. El término "dark" (oscuro) proviene de la impresión que no existe nada en dicha red. Ningún paquete legítimo debiese ingresar a la darknet, y por consiguiente la gran mayoría de dichos paquetes deben haber sido enviados software malicioso (malware). El malware, en general, realiza búsquedas de dispositivos activos, por lo que enviará paquetes hacia la darknet lo que permitirá su captura y futuro análisis [PYBPP04, BCJMS06]. Una darknet es útil pues permite identificar, entre otros aspectos, las fuentes de ataques maliciosos a una red.

Una darknet incluye al menos un servidor configurado como un captador de paquetes. Este servidor captura todo el tráfico y paquetes que ingresan a la darknet. Luego esta información puede ser utilizada para un análisis en tiempo real o un posterior estudio forense. Cabe destacar que no se ha encontrado ningún proyecto similar al de esta tesis que permita visualizar el comportamiento de una darknet.

Existen otras aproximaciones interesantes al concepto de darknet. Por ejemplo, el proyecto "Network Telescope" de CAIDA [C110, CA210, CA310] y el "Internet Motion Sensor", de la Universidad de Michigan [FJ10]. Todos estos proyectos fueron revisados para este trabajo de tesis y una comparación de estos proyectos aparecerá en la sección 2.

Recientemente el CLCERT (Grupo de Respuesta a Incidentes de Seguridad Computacional de Chile), con el financiamiento del proyecto Amparo/LACNIC [PA11, S11], comenzó un proyecto de implementación de una darknet con el fin de determinar el volumen y características del tráfico malicioso que circula por la red chilena. De esta manera se podrían definir acciones a tomar y así mitigar el impacto que dicho tráfico representa sobre los equipos de la red, y a su vez extender el conocimiento adquirido a una red

más amplia (incluso a nivel nacional). En términos técnicos, el proyecto se basó en la implementación de una darknet propuesta por el Team Cymru [CY10]. La presente tesis de magister está por consiguiente asociada al proyecto de CLCERT, enfocándose en el diseño, desarrollo e integración de herramientas gráficas configurables por el usuario o administrador de la darknet que sean útiles para la visualización del tráfico de red de la darknet. Esto es realizado en base al procesamiento de los archivos binarios que contienen el detalle de los paquetes que arriban a la red. De esta manera se podrá colaborar de alguna manera con la seguridad y toma de decisiones.

1. a. Motivación

Actualmente existen pocos antecedentes confiables sobre el número y características de los ataques provenientes desde el extranjero hacia la red nacional chilena. Más aún, prácticamente no se cuenta con información que permita dimensionar cuántos son los equipos locales comprometidos con alguna clase de malware o que formen parte de *botnets* [CJM07]. El ideal sería poder responder a estas dudas de la manera más clara y precisa posible, pero intuitivamente y con una interfaz amigable, como por ejemplo a través de gráficos, imágenes y videos. Esto permitiría monitorear y analizar los problemas de seguridad de los sistemas computacionales en Chile, y reducir la cantidad de incidentes de seguridad perpetrados desde y hacia éstos.

Con el fin de detectar tráfico malicioso de la manera más confiable posible, se propone desarrollar una herramienta de visualización. Ésta utilizará la información entregada por la darknet (archivos binarios de tráfico diario), la procesará y desplegará gráficamente. De esta manera se contará con estadísticas certeras respecto a los posibles ataques.

En base a los estudios realizados, hoy en día no existe una herramienta públicamente disponible capaz de graficar o desplegar de forma visual datos a partir del tráfico existente en una darknet. Se pueden observar gráficos fijos realizados por entidades investigadoras [PYBPP04, CY10, BCJMS06, G, CY08] pero no son adaptables por un analista externo.

2. Estado del arte y Antecedentes

La topología de Internet está en constante evolución y han apareciendo periódicamente nuevos métodos de ataque en que el software malicioso se propaga y se detecta [G03, CJM, PYBPP04]. Al mismo tiempo, los firewalls y dispositivos diseñados para proteger los hogares y las empresas, están empezando a ser permeables a muchas de las amenazas a las que fueron diseñados para defenderse. En particular, los usuarios móviles actúan como portadores de programas maliciosos, puntos de acceso inalámbricos proporcionan puertas traseras a muchas redes y aplicaciones complejas entregan involuntariamente agujeros abiertos a través de firewalls. El resultado final ha sido una proliferación de la actividad maliciosa sin ser detectados dentro de los perímetros de la red [MSB02, SMS01, MPSW03, MVS01].

Para combatir el aumento de las amenazas dentro de la red se busca utilizar los datos entregados por la darknet para estudiar y/o reducir los ataques provocados por malware. Las darknets tienen múltiples usos: acoger el flujo de los colectores (por ejemplo, para alimentar Honeynets [JR04]), los detectores de backscatter (como los utilizados para detectar ataques de Denegación de Servicios [MVS01]), analizadores de paquetes y sistemas de detección de intrusos (IDS, Intrusion Detection System [MHL02])

[U10]. Lo ventajoso de una darknet es que reduce considerablemente los falsos positivos en comparación a cualquier otro dispositivo o tecnología.

Tal como se mencionó anteriormente, Stefan Savage y su equipo de la Universidad de California, San Diego (UCSD), [BCJPR05, MSBV06] definieron por primera vez a lo que hoy se conoce como “darknet”. Otras organizaciones también han propuesto variantes de darknets. Tal es el caso de CAIDA, con su proyecto llamado “*Network Telescope*” o “telescopio de red” [CA110, CA210, CA310] y el “*Internet Motion Sensor*” [FJ10] de la Universidad de Michigan que se comporta como una darknet distribuida, es decir, como pequeñas darknets repartidas por la red. Cada uno de estos sistemas será detallado a continuación.

Según CAIDA [CA110, CA210, CA310], un telescopio de red es una porción de direcciones IPs donde debería existir poco o nada de tráfico legítimo. El monitoreo de tráfico inesperado que llega a este espacio permite obtener una visión de lo que está ocurriendo en la red. Los servicios “monitoreados” dependen del tipo de requerimiento observado. Por ejemplo, si un adversario intenta conectarse a un sitio web, la darknet re-rutea el requerimiento a un servidor web (posiblemente creado *on the fly*). Este tipo de implementaciones son más genéricas y operativamente más costosas que la darknet simple actualmente implementada en el CLCERT.

Otra variante existente de una darknet simple es el Telescopio Anycast [MO10]. La tecnología anycast es una forma de direccionamiento en la que la información es enrutada al mejor destino posible desde el punto de vista de la topología de la red. El Telescopio resulta ser útil al momento de monitorear la red cuando se encuentra sobrecargada ya que provee a los hosts finales rutas más cortas al telescopio.

El “*Internet Motion Sensor*” (IMS), por otro lado, es una variante desarrollada por la Universidad de Michigan [FJ10]. El IMS es un sistema de vigilancia de amenazas a nivel mundial, cuyo objetivo es medirlas, caracterizarlas y realizarles un seguimiento. Para ello, utiliza una gran colección de sensores distribuidos que vigilan bloques de espacios de direcciones no utilizadas y globalmente enrutables.

El diseño de la darknet implementada en el CLCERT fue el siguiente, todo el tráfico dirigido hacia la red monitoreada es ruteado hacia un servidor de recolección, no se permite que haya tráfico que escape de dicha red. El tráfico capturado que proviene de Internet es dirigido hacia la red que cuenta con 250 direcciones IPs (ipv4, provenientes de una red de clase C en desuso). La conexión entre el servidor y el router se realiza a través de una red privada. Los paquetes (de tipo IP/UDP/ICMP) se recopilan mediante tcpdump [TCPDU10] desde la interfaz de captura (bge0) en el servidor de recolección. Luego se almacena diariamente en archivos binarios (de tipo pcap, package capture) para su posterior análisis. El formato del nombre de los archivos es salida-año-mes-día (por ejemplo salida-2011-01-02). El Diagrama 1 resume lo descrito anteriormente.

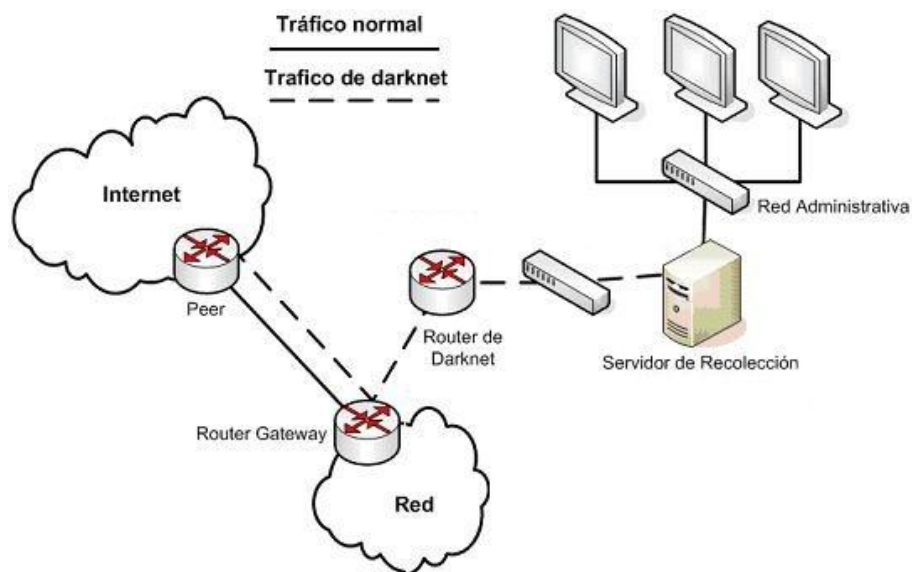


Diagrama 1: Diseño de la darknet del CLCERT. Imagen perteneciente a la documentación del proyecto Amparo. 2010.

La motivación esencial para la creación de una darknet es detectar ataques, como por ejemplo: Denegación de Servicios [MVS01], infección de hosts con gusanos de Internet [MPSW03] y escaneos de red, ataque de Respuesta DNS, ataques a routers, escaneo de puertos mediante el envío de paquetes ICMP, problemas asociados a malas configuraciones [PYBPP04] y ataques a servicios particulares tales como Web, Mail, SSH, FTP (archivos) entre otros. Dentro de los ataques y eventos se pueden detectar a través de una darknet o Telescopio de red [M10] se encuentran:

1. Actividad sospechosa por puertos (diferentes protocolos, TCP, UDP, ICMP, etc.):
2. Tráfico relacionado con servicios específicos (SSH, WEB, DB, etc.).
3. Direcciones IP y dominios en lista negra.
4. Ataques comunes a equipos de la red (fuerza bruta, escaneos, etc.).
5. Patrones generados por malware (escaneos, tráfico excesivo, baja de servicios por parte de gusanos o exploits automatizados, etc.).
6. Botnets dentro y fuera de la red.

Una vez que se tiene toda la información del sistema, ésta es procesada mediante scripts. Debido a la gran cantidad de datos que se reciben diariamente [MSBVS06], éstas se deben procesar en tiempo real. Los principales objetivos del procesamiento son (a) clasificar la información, (b) formatear la información, y (c) analizar la información.

Los principales ataques en los que las mencionadas organizaciones (Team Cymru [CY10], CAIDA [CA110], Universidad de Michigan [FJ10]) se han centrado son los siguientes [CA310]:

2. a. Ataque de denegación de servicios (DoS): Un DoS es un ataque a una red que provoca que un servicio o recurso sea inaccesible a los usuarios. Normalmente provoca la pérdida de la

conectividad de la red por el alto consumo del ancho de banda de la red o sobrecarga de los recursos computacionales del sistema de la víctima. Para dificultar que la víctima bloquee el ataque, el adversario usualmente utiliza una dirección IP fuente falsa y aleatoria (análogo a alterar la dirección del remitente en el correo) en cada paquete que envía a la víctima (Figura 1.a). Dado que el atacado no es capaz de distinguir entre una solicitud proveniente de un atacante o una legítima, la víctima del DoS intenta responder a todas las solicitudes que le llegan (Figura 1.b). Por ende, cuando el atacante falsifica una dirección de origen en el telescopio de red o en la darknet, ésta recibe una respuesta destinada a un computador que no existe (y por lo tanto nunca pudo haber enviado la consulta inicial) (Figura 1.c). Al monitorear estas respuestas no solicitadas, los investigadores pueden identificar a la víctima del ataque de DoS, inferir información sobre el volumen del ataque, ancho de banda de la víctima, su ubicación y los tipos de servicio a los que apunta el atacante [MVS01].



Figura 1: Funcionamiento del Ataque de Denegación de Servicios. Imagen adaptada de [IMCA11]. 2011.

2. b. Gusanos: Es un tipo de malware que tiene la propiedad de duplicarse a sí mismo. Los gusanos utilizan las partes automáticas del sistema operativo (que generalmente son invisibles al usuario), causan problemas en la red (consumiendo ancho de banda) y no alteran los archivos de programas, sino que reside en la memoria. Dado que el espacio de direcciones de la darknet nunca responde (no hay computadores ni servicios asociados a dichas IPs) es inusual que un computador de la darknet sea el blanco de un ataque. Sin embargo un gusano de gran escala que se intente propagar al azar podría ser detectado con una darknet. Otra posibilidad son errores de configuración humanos por parte de administradores de sistema [MPSW03, MSB02, SMS01].

2. c. Escaneo Malicioso: Existen intentos automáticos, semiautomáticos y manuales para localizar computadores vulnerables en Internet. El escaneo difiere de otro tipo de tráfico visible porque no se produce aleatoriamente. Los motivos del atacante al momento de elegir su blanco parecen ser arbitrarios desde la perspectiva de la víctima. Una darknet recibe muchos tipos de escaneos continuamente, incluyendo escaneos basados en “ICMP ping”, el cual puede usarse para consultar por la existencia de un dispositivo en una determinada dirección IP, e incluso explorar secuencialmente una pequeña lista de puertos vulnerables pertenecientes a un cierto rango de direcciones IP. Otros tipos de escaneos incluyen aquellos basados en simples conexiones TCP, y escaneos usando paquetes SYN

seguidos por TCP “reset”. Esta última técnica consiste en activar la flag reset (RST = 1) para interrumpir la conexión TCP luego de recibir el primer paquete de respuesta de la víctima (típicamente un paquete TCP SYN-ACK) y con ello evitar dejar evidencia que se intentó una conexión¹.

3. Definición del problema

3. a. Objetivo general

Tras el estudio realizado, se pudo determinar que no existe sistema que permita la visualización de forma gráfica de los datos arrojados por una red de tipo darknet. Es por esto que esta tesis tiene como objetivo principal la creación un nuevo sistema para la visualización y análisis de datos de una darknet. El desarrollo del sistema involucra la integración y evaluación de las herramientas mencionadas anteriormente. La herramienta debe generar gráficos que permitan visualizar el tráfico capturado por la darknet. Los gráficos deben permitir estudiar el comportamiento de la red con el fin de monitorear ataques DoS, gusanos, spam y compromisos en general.

3. b. Objetivos Específicos

- a. Definición de un formato de obtención de datos desde la darknet para su posterior manejo y análisis.
- b. Comparar las herramientas mencionadas entre ellas e integrarlas a la darknet, para obtener gráficos y videos que representen lo que sucede en la red.
- c. Creación e implementación de algoritmo para filtrar la alta cantidad de datos fuente recibidos de manera eficiente para así permitir un procesamiento suficientemente rápido como para tomar decisiones a tiempo². A partir de ello, se implementará una herramienta que creen los gráficos de acuerdo a parámetros definidos por el usuario a partir de una aplicación web.

¹ Comúnmente este método se utiliza en el caso de que un computador A, por ejemplo, reciba paquetes de un computador B. En caso de que A se reinicie seguirá recibiendo paquetes que están fuera de contexto para él (porque los desconoce), entonces envía un “reseteo TCP” para que el computador B sepa que la conexión no sigue existiendo. Existe la posibilidad de que un tercer equipo malicioso llamado C esté monitoreando la conexión y sea él quien le envíe el paquete con el flag RST y así interrumpa la conexión sin el consentimiento de las partes involucradas (B es engañado ya que piensa que el paquete falsificado es enviado por A). Esta controvertida técnica es utilizada por los proveedores de Internet cuando se quieren romper conexiones “indeseables” (por ejemplo “peer to peer”) [AW05].

² Este trabajo no evaluará cuánto más eficiente es el análisis basado en gráficos versus el basado en texto. Tal análisis quedará como trabajo futuro.

- d. Creación de sitio web donde se desplieguen los gráficos creados por herramientas estudiadas (ver sección Antecedentes) y desde donde se podrán crear nuevos gráficos de acorde a las necesidades del usuario gracias a la herramienta desarrollada a medida (punto (c) anterior). El sitio tendrá filtros y opciones para que el usuario ajuste el gráfico a sus necesidades, además de ayuda e información útil para graficar (lista de IPs y puertos relevantes por ejemplo).
- e. Determinación de un formato (XML) para compartir la información de tráfico con otros CERTs. Por último se contará con un sistema de alertas donde se le notificará al analista cuándo se sobrepasen índices de tráfico definidos por él.
- f. Estudiar y sugerir mecanismos para manejar el problema de big data [MSBVS06] para la aplicación. Realizar un gráfico ejemplo manejando grandes volúmenes de tráfico.

3. c. Metodología

La metodología se basó principalmente en la investigación y la evaluación crítica de herramientas y sus resultados. Además de esto se investigó sobre métodos y procedimientos para la creación de nuevas herramientas. A continuación se detallan los pasos seguidos:

- a. Inicialmente se realizó un estudio de representaciones gráficas que podrían ser útiles para un analista.
- b. A continuación se estudió los distintos formatos con que la darknet puede entregar la información sobre el tráfico que recibe.
- c. Luego, se evaluó herramientas que ya se encuentran desarrolladas con el fin de determinar si entregan información relevante y de interés en términos de seguridad, y si son compatibles con la información fuente.
- d. Posteriormente se evaluó si satisfacen las necesidades planteadas para así utilizarlas. En caso de que no cumplieran en su totalidad con lo requerido se analizó la opción de modificarlas.
- e. Se diseñaron nuevas soluciones para contar con información valiosa referente a la toma de decisiones a partir de gráficos. Para esto se estudió cómo manipular o filtrar eficientemente los datos fuente para extraerles la información relevante.
- f. Finalmente, se estudió cómo extender el sistema de gráficos para manejar grandes volúmenes de datos. Las soluciones incluyeron la utilización del algoritmo de Map/Reduce en *Cloud Computing* [DG04, JDV09]

En el proceso de evaluación de estas herramientas, es importante tener siempre presente qué tipos de ataques se desea monitorear y considerar las experiencias de otras organizaciones que han trabajado en este tema. Una herramienta existente es descartada si no otorga la información necesaria para identificar ataques o comportamiento malicioso proveniente de alguna dirección IP.

3. d. Restricciones y recursos:

Se cuenta con 2 equipos otorgados por el CLERT: un computador de escritorio (ubicado en el laboratorio del CLCERT) y un servidor web, además del equipo portátil de la tesista.

El computador de escritorio es el encargado de la creación de todos los gráficos y videos existentes en este trabajo de tesis. Lamentablemente no tiene grandes capacidades de procesamiento (Procesador Intel Pentium Dual Core, 1 Gb de memoria RAM, 3.00GHz de CPU, 70 Gb de disco duro, Sistema Operativo Ubuntu 10.04.4 de 64 bits), dada esta restricción los códigos y scripts empleados para esta tesis tuvieron que ser optimizados lo máximo posible para poder llevar a cabo todas las tareas en este computador.

Los frecuentes cortes de luz en la Universidad provocaron ciertos inconvenientes dado que el computador del laboratorio debe permanecer siempre prendido, su apagado repentino provocaba que no le llegaran datos desde la darknet y que no pudiera crear todos los gráficos que estaban programados. Los cortes de luz no afectaban la recopilación de los datos dado que el servidor de recolección está conectado a una UPS (Uninterruptible Power Supply). El computador fue programado para reiniciarse, en caso de un corte de luz, automáticamente. Una vez prendido, un servidor virtual X instalado en el computador (Real VNC) [XSERV11] y las tareas programadas como "root", se encargan de comenzar la creación de los gráficos a pesar de que no se haya iniciado sesión y que la pantalla se encuentre apagada. Finalmente este computador formó parte de la UPS para evitar futuros inconvenientes.

El servidor web se encarga únicamente del despliegue de los datos en la página web. Para ello, el envío y recepción de datos entre el computador del laboratorio y el servidor web son periódicos.

4. Discusión: Solución Propuesta

Resulta importante determinar qué gráficos se realizarán y de qué manera. Hay que destacar que los gráficos deben ser auto explicativos para que el usuario pueda comprenderlos fácilmente y la toma de decisiones no se vea retrasada.

Siguiendo las sugerencias del libro Applied Security Visualization [RM10], se determinó que el uso de las siguientes herramientas existentes podría ser de gran utilidad para cumplir las metas deseadas:

a. AfterGlow: [AF10] software que realiza una malla o grafo con todas las conexiones existentes en un cierto período de tiempo. Es considerada una herramienta open source útil dada la información que entrega y su fácil configuración [RM10]. Varios estudios que han utilizado este software para conocer el comportamiento de determinados malwares [V09, L06, JB05].

b. TCPstat: [TS10] herramienta que permite la generación de gráficos en dos dimensiones. Permite graficar cuántos paquetes por segundo llegan a la red durante un determinado tiempo. Esta herramienta es una de las más usadas en monitoreo de redes y ha sido utilizada en diversos estudios dada su simplicidad y eficiencia [TDT03, BD03, CB07].

c. EtherApe: [EA10] permite generar un video tomando como entrada un archivo de tráfico. Se crea una simulación *en tiempo real* de las conexiones existentes entre el exterior y la darknet. Este software permite tener una visión temporal clara sobre lo que ocurre y se destaca como una de las mejores existentes [RM10]. La simulación mediante video resulta atractiva pues incorpora el factor tiempo en el grafo de conectividad y por ello varios investigadores han comenzado a usar esta herramienta [MG08, TPP09].

d. Moncube: [MON10] sistema capaz de generar gráficos pero simulando el tráfico dentro de un cubo que se puede mover, rotar, acercar y alejar entre otras propiedades. Herramientas similares (como Shoki [S11] e InetVis [I11]) se describen en la literatura destacando su manera de integrar variada información desplegándola de forma sencilla [RM10]. No se han encontrado publicaciones ni investigaciones que utilicen este software.

Los cuatro sistemas mencionados anteriormente crearán gráficos y videos configurables, pero por el administrador del sistema (tesista en este caso), pero cabe destacar que, tal como fue mencionado con anterioridad es deseable la existencia de gráficos creados por los usuarios. Es por esto que se creó un software capaz de parsear y filtrar los datos de la manera seleccionada por el usuario para luego desplegar gráficos acordes a sus necesidades.

Para el despliegue de los gráficos se utilizará el software Maani Charts [MA11] que recibe como entrada un archivo de formato XML con cierta estructura definida (dependiente de cada tipo de gráfico a crear) y a través de Flash [F12] es capaz de desplegar en una página web los gráficos que se especifiquen. Este software existe en versión gratuita y pagada, la única diferencia entre ambos es que la segunda (utilizada en este proyecto) permite interacción por parte del usuario con el gráfico (aplicar filtros, zoom, rotaciones etc).

Existen otras herramientas que se podrían haber utilizado para esta tarea, tales como High Charts³, Google Charts⁴, rrdtool⁵, graphite⁶ etc., pero se optó por Maani Charts dada su sencillez, interactividad y la posibilidad de reutilizar los XMLs utilizados para compartir información con otros centros de seguridad.

La creación del archivo XML que recibirá Maani Chart como entrada será creado a partir de un Parser, que se encargará de crearlo a partir del procesamiento de datos del tráfico en binario que entrega la darknet cada día.

Maani Charts permite la creación de variados tipos de gráficos (tanto en forma de desplegar la información como en diseño) gracias a su extensa documentación y ejemplos provistos en su sitio web [MAPI12]. Es por esto que se hizo un estudio de qué gráficos podrían ser útiles para un analista de seguridad, teniendo siempre presente que la simpleza de éstos es fundamental para su rápida comprensión.

³ Disponible en <http://www.highcharts.com/>.

⁴ Disponible en <https://developers.google.com/chart/>.

⁵ Disponible en <http://oss.oetiker.ch/rrdtool/prog/index.en.html>.

⁶ Disponible en <http://graphite.wikidot.com/>.

A continuación se detallan los gráficos creados para este trabajo de tesis y sus principales características:

i. Accesos por puerto: A través de estos gráficos se puede saber la cantidad de accesos a cada uno de los puertos escogidos. El despliegue se realiza en un gráfico de torta que indica el porcentaje de los accesos a cada puerto escogido de la darknet.

Es importante destacar la relación entre este gráfico y el ataque de escaneo de puertos (mencionado en la sección 2). Si el analista observa, en este gráfico, que los puertos principales son accedidos en porcentajes similares, puede sospechar que se realizó dicho ataque en la red. Además podrá ser capaz de detectar otro tipo de malware como gusanos o troyanos (dedicados a puertos).

En la detección de un escaneo de puertos es importante conocer los porcentajes de accesos a éstos. Se utilizó un gráfico de torta para desplegar la información, pues ofrece visualmente una forma rápida e intuitiva de comparar las partes de un entero.

Además se pueden identificar ciertos patrones que mediante otros medios de visualización, una tabla por ejemplo, podrían ser obviados: existencia de 2 puertos altamente accedidos y el resto de ellos con un bajo porcentaje de accesos. En este caso se destacarán visualmente los grandes porcentajes, y los pequeños se visualizarán unos junto a otros en grupos (evitando ruido). Esto es mucho más rápido e intuitivo de identificar que buscar grandes números en una tabla y comparar con el resto de los valores.

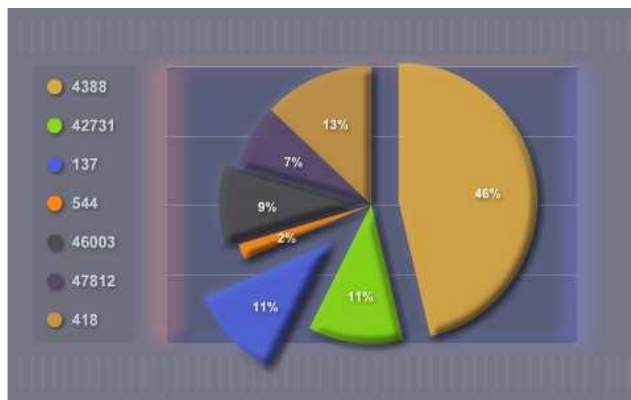


Figura 2: Accesos por puerto. Maani Charts. 2012.

ii. Accesos por puerto y protocolo: A partir de estos gráficos se conocerán las conexiones existentes a los puertos escogidos permitiendo una comparación entre los distintos protocolos. Primero se podrá apreciar un gráfico polar que graficará la cantidad de accesos a cada puerto y su protocolo. Al ser todas las curvas graficadas a la vez se permite una fácil comparación. Además se desplegará un gráfico de burbujas que entrega mucha información a la vez: protocolo, puerto, accesos y la dirección IP desde donde se realizó la conexión (al deslizar el puntero sobre las burbujas se desplegará el detalle de la información). Finalmente se mostrará un gráfico de barras que indica los puertos escogidos, la cantidad de accesos (desplazando el mouse sobre las barras) y el protocolo mediante el cual se hizo la conexión.

Mediante la comparación entre las curvas, burbujas y barras se puede tomar decisiones al saber qué puertos están siendo más vulnerables y por quienes son accedidos.

De los tres gráficos mencionados el principal es el de barras. Esto es porque gracias a él también se pueden identificar ataques de escaneos de puertos. A diferencia del caso anterior, en este gráfico se observan las cantidades absolutas de accesos a los puertos principales, junto al protocolo utilizado. Si las barras coinciden en su largo, probablemente se esté frente al ataque mencionado. El hecho de que además se despliegue el protocolo ayuda a que se puedan detectar gusanos, por ejemplo.

El gráfico de barras muestra de manera clara cantidades absolutas y los colores asociados a los protocolos permiten tener una rápida visión de lo que está sucediendo. El gráfico polar entrega la misma información pero mostrada de otra forma, esto es útil al momento querer comparar las curvas generadas para cada uno de los puertos, además la identificación de patrones resulta particularmente interesante llamando, en este caso, más la atención los protocolos que los puertos mismos. El gráfico de burbujas entrega los datos de manera clara y resumida en caso de que la cantidad de datos sea pequeña (si los círculos tienen un radio grande podrían provocarse solapamientos entre las burbujas y crear confusión).

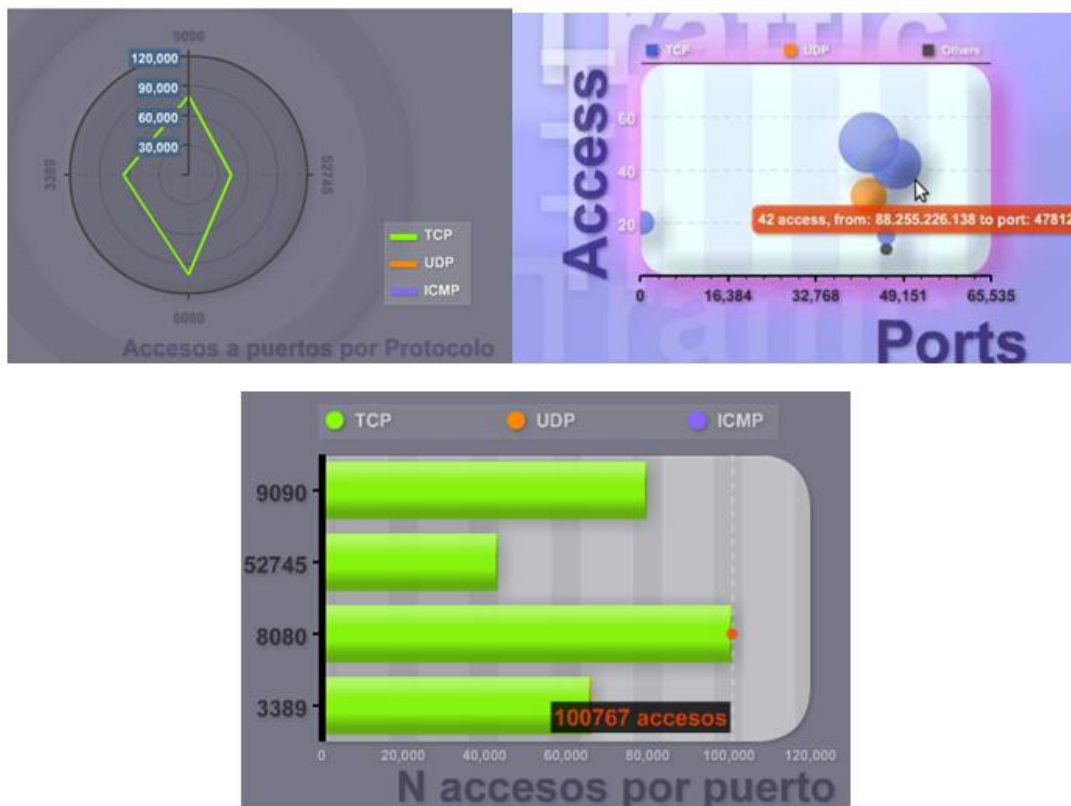


Figura 3: Accesos por puerto y protocolo. Maani Charts. 2012.

iii. Accesos a puertos en el tiempo: Con este gráfico se conoce la cantidad de conexiones existentes a los puertos escogidos y cómo éstas han ido evolucionando en el tiempo. Los principales puertos accedidos en el día serán desplegados en una lista junto a la cantidad de accesos diarios. Además se trazará una recta para analizar su tendencia a medida que pasan los minutos (al deslizar el puntero sobre

los puntos se desplegará el detalle de la información). Cabe destacar que el despliegue de los accesos es por minuto y no acumulativo en el tiempo.

Con esta información se consigue conocer cómo se van atacando los puertos desde el exterior y así analizar en qué horarios se producen la mayor cantidad de escaneos de puertos. Si un puerto, por ejemplo, es accedido constantemente a lo largo del día se podría sospechar que un ente malicioso está intentando dar de baja a un servicio que corra en ese puerto específico.

Este gráfico de puntos detalla de manera sencilla el comportamiento temporal de los accesos a puertos. Los colores permiten diferenciar y comparar fácilmente la información entre un puerto y otro, permitiendo hacer análisis y tomar decisiones.



Figura 4: Accesos a puertos en el tiempo. Maani Charts. 2012.

iv. Accesos por protocolo: Mediante el análisis de estos gráficos se puede tener una idea de los protocolos más utilizados por los posibles atacantes. Uno de los gráficos despliega mediante una torta los protocolos utilizados en porcentaje (de los puertos principales). El otro, consiste en un gráfico de anillos que muestra la comparación de tráfico existente para la última semana a partir del día escogido y el protocolo utilizado. De esta manera se puede tener una visión más a largo plazo de las conexiones que llegan a la red y posiblemente inferir algún tipo de comportamiento anómalo, como por ejemplo, ataque a servicios en particular.

El gráfico de torta está vinculado a la detección de ataques de denegación de servicios. Si el analista visualiza por ejemplo una alza en la cantidad de accesos mediante el protocolo UDP, se podría sospechar de un ataque dado que los servidores DNS responden con el protocolo UDP, pero ¿por qué a la darknet un DNS le respondería si ella no ha enviado ninguna solicitud? sólo en caso de que un atacante esté consultado a los DNS falseando su dirección de origen por una perteneciente a la darknet.

Nuevamente se está en presencia del uso de gráficos de torta por su intuitiva manera de representar porcentajes (de accesos por protocolo en este caso). Afortunadamente, el gráfico de anillos da información temporal relevante y debido a que cada día tiene un área asociada, se pueden identificar patrones rápidamente, como por ejemplo, qué día tuvo más accesos TCP durante la semana, y cómo es ese valor comparado con los accesos mediante UDP.

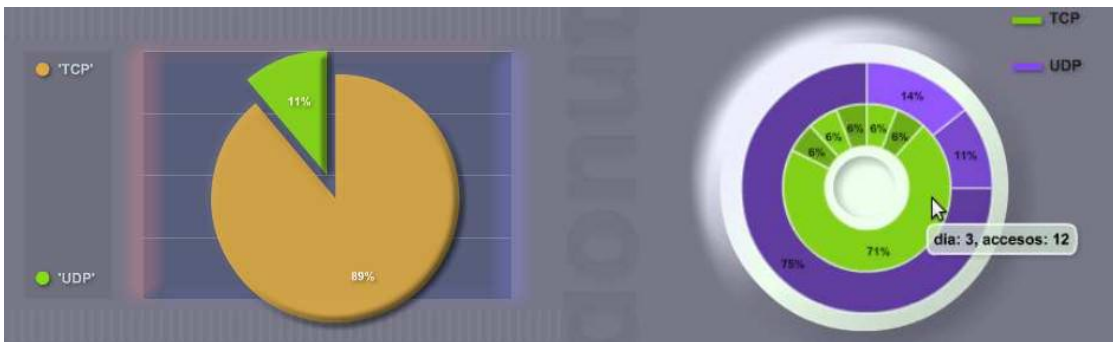


Figura 5: Accesos por protocolo. Maani Charts. 2012.

v. Accesos por protocolo y dirección IP externa: Estos gráficos dan a conocer la cantidad de solicitudes o paquetes que llegan a la darknet, mediante cierto protocolo y desde cierta dirección IP. De esta manera se tiene noción de quienes son los que intentan acceder a la red, desde donde vienen, qué ISP tienen y toda la información que pueda ser deducida a partir de la dirección IP. Deslizando el puntero por el gráfico se tendrán detalles sobre las conexiones. Además se permiten rotaciones de los gráficos para tener una vista más cómoda.

Con esta información se podría notificar a las ISP cuales de sus direcciones IP pudieran haber sido comprometidas.

Ambos gráficos muestran la misma información, la única diferencia entre ellos es la dimensión utilizada. A pesar de ello, el gráfico en 2D es más intuitivo ya que tiene menos ruido (no despliega ceros) y por lo mismo su lectura es más simple. Entrega información útil para estudiar si existe un ataque de Denegación de Servicios a la red (alguna IP atacante enviando muchos paquetes a la red).

Nuevamente las barras son ideales para desplegar cantidades absolutas y el color de los puertos acompaña para la detección de patrones visuales existentes: “la gran mayoría de accesos se realizaron desde IPs con protocolo TCP, siendo el pick 300 accesos provenientes de la dirección 172.17.55.133”.

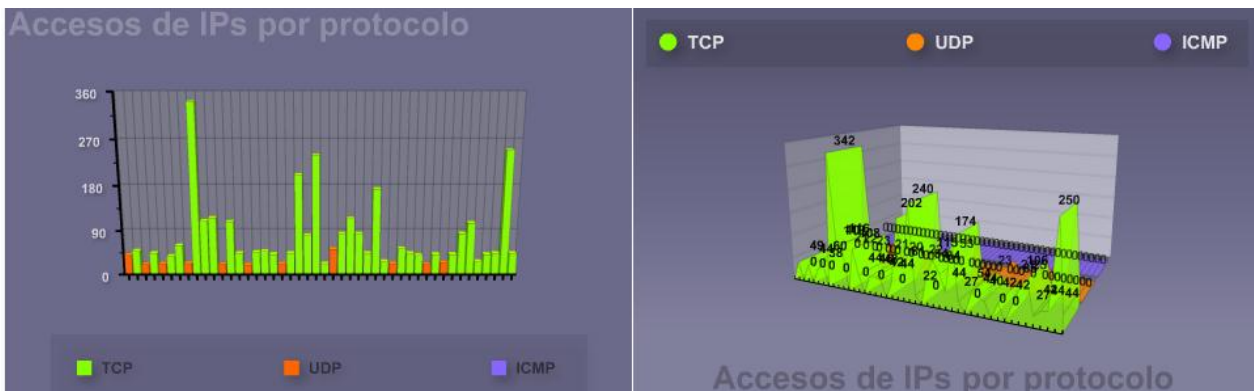


Figura 6: Accesos por protocolo y dirección IP externa. Maani Charts. 2012.

vi. Accesos de direcciones IP externas en el tiempo: Con este gráfico se puede conocer la cantidad de conexiones existentes desde las direcciones IP escogidas y cómo éstas han ido evolucionando en el tiempo. Al usuario se le presenta una lista con las direcciones IP que han contactado a la darknet en

cierta fecha, la ISP de cada dirección, la cantidad de accesos y el país de proveniencia. Se trazará una recta para analizar su tendencia a medida que pasan los minutos (al deslizar el puntero sobre los puntos se desplegará el detalle de la información).

Con estos datos se consigue conocer el comportamiento particular de algunas direcciones IP y así analizar en qué horarios se producen la mayor cantidad de accesos o posibles ataques.

Este gráfico otorga información para identificar un ataque de Denegación de Servicios, al igual que con el gráfico anterior, pero ahora se suma como dato la hora de los accesos. Cabe destacar la importancia de la información entregada en la tabla, con ella se pueden analizar diferentes aspectos, como por ejemplo: ¿qué país tiene más IPs comprometidas o accede más frecuentemente a la red? ¿A qué ISP pertenece la IP con mayor cantidad de accesos? etc.



Figura 7: Accesos de direcciones IP externas en el tiempo. Maani Charts. 2012.

vii. Accesos a direcciones IP de la darknet en el tiempo: Con este gráfico se puede conocer la cantidad de conexiones existentes a las direcciones IP de la darknet y cómo éstas han ido evolucionando en el tiempo. Este gráfico es igual al del punto anterior (vi) exceptuando que las IPs graficadas son las de la darknet. En este caso no se despliega una tabla con las IPs porque se sabe que todas ellas pertenecen a la darknet y se conoce su forma.

Este gráfico ayuda a identificar un ataque de Denegación de Servicios (una dirección IP de la darknet es accedida repetidas veces) y un escaneo de IPs (se apreciará la forma en que las direcciones IP de la darknet más accedidas fueron atacadas en el tiempo).

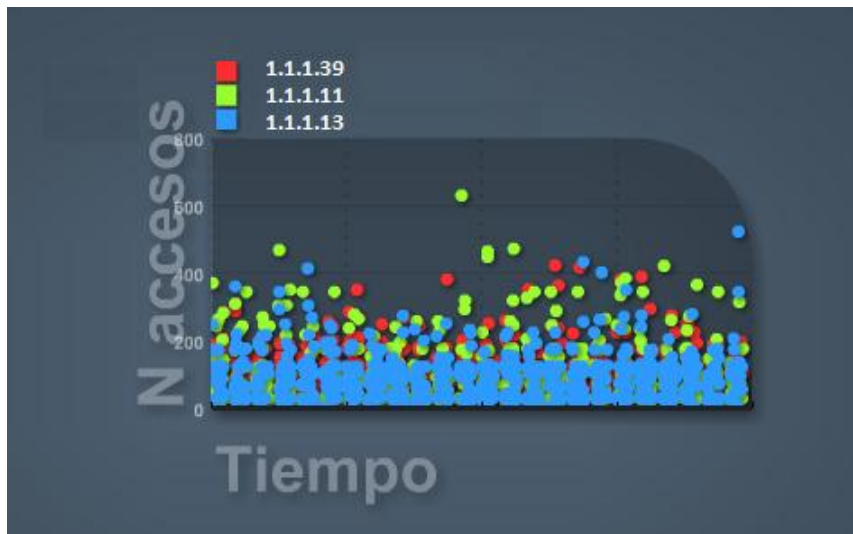


Figura 8: Accesos de direcciones IP de la darknet en el tiempo. Maani Charts. 2012.

viii. Accesos por protocolo y dirección IP de la darknet: Estos gráficos dan a conocer la cantidad de solicitudes o paquetes que llegan a la darknet, indicando la dirección IP atacada. Se puede notar que estos gráficos son iguales a los el punto v, exceptuando el hecho de que la IP desplegada es la de destino.

Entrega información útil para estudiar si existe un ataque de Denegación de Servicios a la red (alguna IP de la darknet recibiendo muchos paquetes). Además permite saber si se está realizando un escaneo de direcciones IP a la darknet, ya que se apreciará cuáles están siendo accedidas y en qué cantidades.

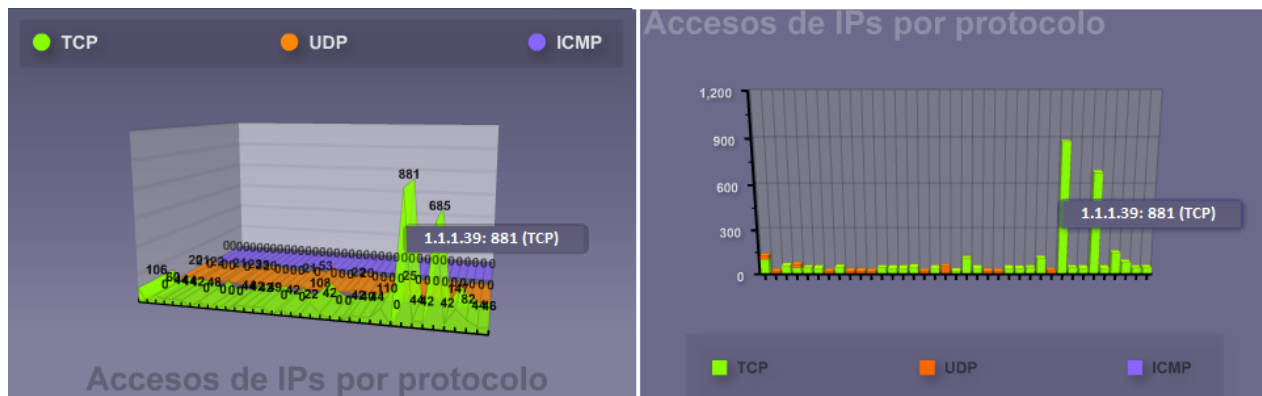


Figura 9: Accesos por protocolo y dirección IP de la darknet. Maani Charts. 2012.

La creación de cada uno de estos gráficos implica un gran procesamiento por parte del computador encargado de realizarlos. Cada vez que el usuario desee crear y visualizar un gráficos ad-hoc en la página web se enviará una solicitud al computador a cargo lo cual pudiera sobrecargarlo. Es por esto que se tomó la decisión de crear gráficos “por defecto” cada noche, para que cuando el usuario ingrese al sitio siempre pueda ver gráficos interesantes pre-calculados, y si éstos no cumplen con sus expectativas podrá dar la orden de crear uno con las parámetros que estime convenientes.

Aparte de los gráficos recientemente explicados, se crearon tres más con la intención de entregar información sencilla sobre el estado de la darknet diariamente:

ix. Gráficos de estado: 10 puertos más accesados por protocolo y los 10 países que realizan más accesos a la darknet: En la Figura 10 se aprecian los puertos de la darknet más accesados según protocolo y su cantidad de accesos diarios; mientras que en la Figura 11 se pueden ver los diez países que realizaron mayor cantidad de accesos durante el día a la darknet y cuántos éstos fueron.

A diferencia de los otros gráficos anteriormente explicados, éstos no serán editables por el usuario, sino que cada día entregarán información fija del estado de la darknet durante esa jornada.

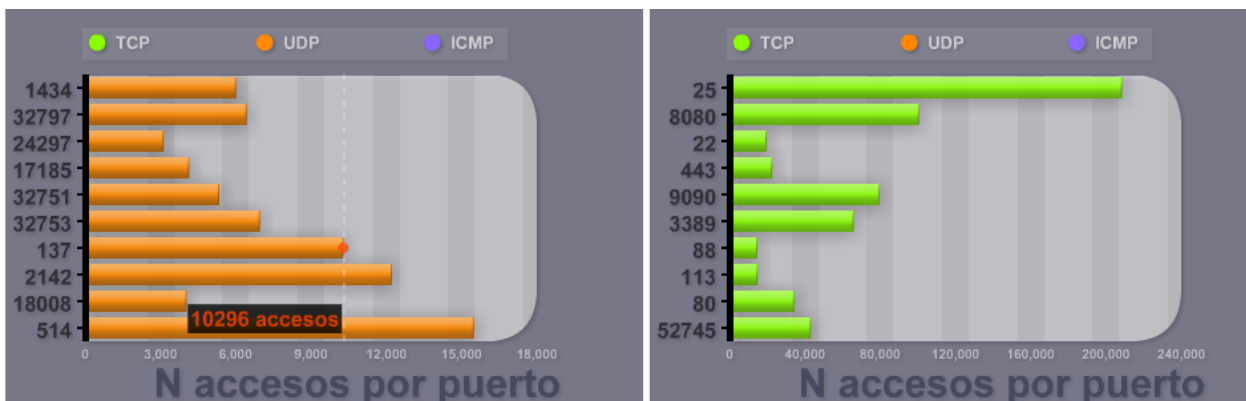


Figura 10: Gráficos de estado: 10 puertos más accesados según protocolo. Maani Charts. 2012.



Figura 11: Gráfico de estado: 10 países que realizan mayor cantidad de accesos a la darknet. Maani Charts. 2012.

El sitio web cuenta con un sistema de alertas, donde el usuario podrá determinar qué índices estima riesgosos de un conjunto predefinido (cantidad de accesos por determinado protocolo, o a un determinado puerto) y luego revisar un registro que indica si alguno de los límites definidos fue sobrepasado.

Para comodidad del usuario, el sitio web tiene un calendario en donde está indicado si para cierto día existen gráficos o videos disponibles (dependiendo de lo seleccionado). De esta manera, el usuario no perderá tiempo buscando gráficos en días en que el sistema no estaba disponible.

Finalmente el sitio web tiene las listas de puertos atacados y direcciones IPs para cada día, justo con información relevante respecto a ellas (número de accesos diarios, país de procedencia, proveedor de Internet asociado). Estas listas ayudarán al usuario a determinar qué puertos y/o direcciones IP desea visualizar en un determinado período de tiempo.

4. a. Utilización de herramientas existentes:

Tal como se mencionó anteriormente se utilizó distintos software de monitoreo de redes para la creación de gráficos diarios. Las herramientas seleccionadas fueron: Afterglow, TCPstat, Moncube y EtherApe. Cada uno de estos programas genera gráficos automáticamente a partir de un archivo que contiene la descripción de los paquetes que tienen como destino la Darknet.

El archivo de tipo pcap con el tráfico diario recibido en la Darknet es almacenado en un servidor, y es descargado en forma automática y segura (utilización de crontab y scp) al computador donde se generarán los gráficos. De esta manera, siempre se cuenta con el archivo de tráfico del día anterior alojado en el computador para generar gráficos diarios.

Además, existe una tarea programada que, cada 5 minutos, descarga el archivo de tráfico del día actual lo cual permitirá tener una visión en línea de lo que ocurre en la Darknet, mediante la creación de videos cada 15 minutos.

Las imágenes (creadas con Afterglow, Moncube y TCPstat) y videos (creados con EtherApe) se envían a un servidor web para luego desplegarlos en Internet.

El Diagrama 2 muestra lo que sucede entre los servidores mencionados:

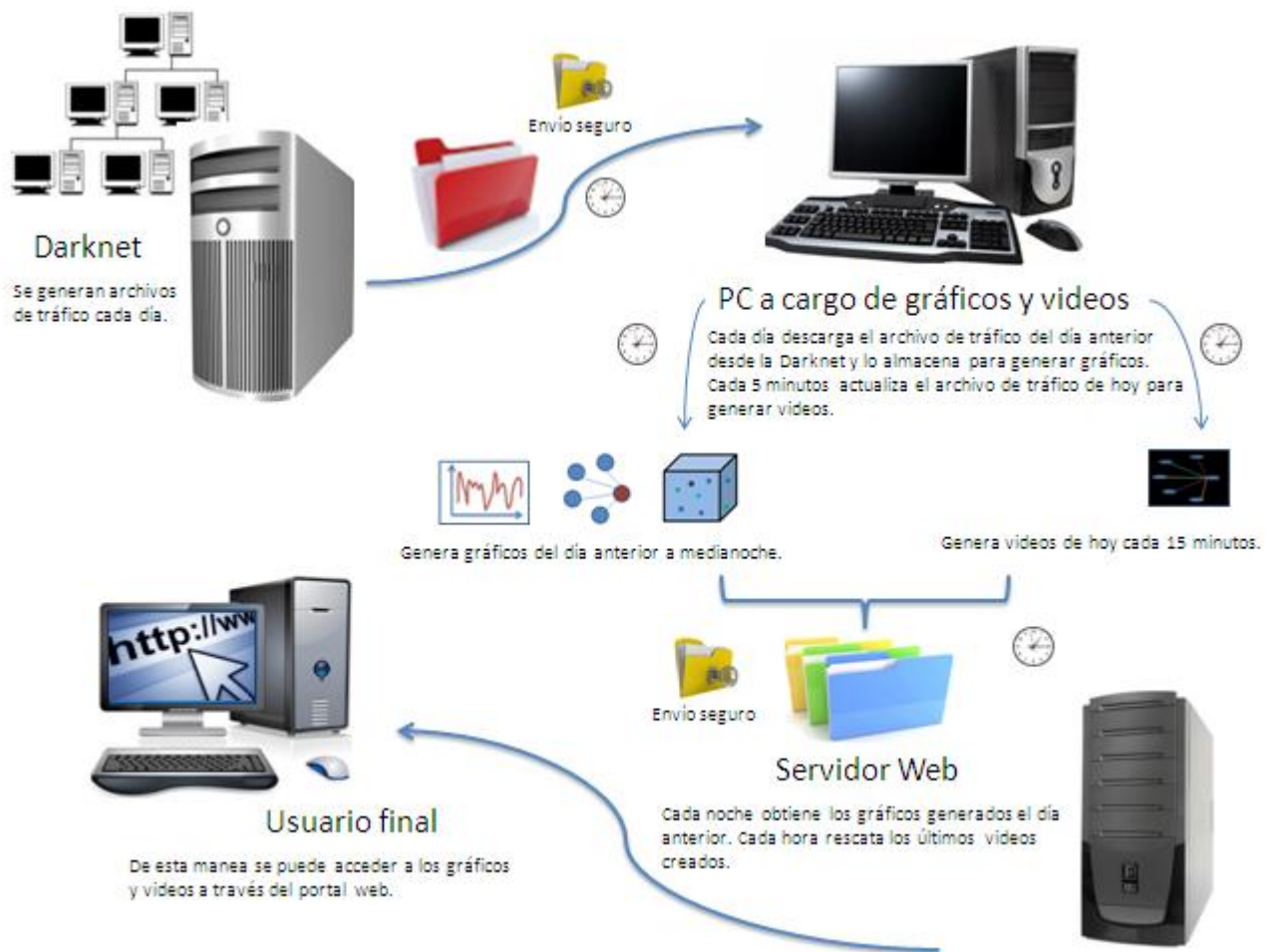


Diagrama 2: Representación del funcionamiento del sistema. Imagen original. 2011.

Cabe destacar que todas las herramientas utilizadas generan distintos gráficos a partir del archivo de tráfico del día anterior, a excepción de EtherApe cuya labor es generar un video simulando el tráfico existente en los últimos 10 minutos.

A continuación se explicarán cada una de las herramientas utilizadas y su funcionamiento:

4. a.1. AfterGlow:

AfterGlow [AF10] es un software que realiza una malla con todas las conexiones existentes en un cierto período de tiempo. Para este proyecto de tesis se generan varias mallas por día. Durante cada media hora del día, se generan 3 gráficos: uno a los 10 minutos, otro a los 20 minutos y otro al final de la media hora. De esta manera se puede apreciar cómo va variando el estado de las conexiones existentes en la red.

Éstas son algunas de las imágenes obtenidas con AfterGlow:

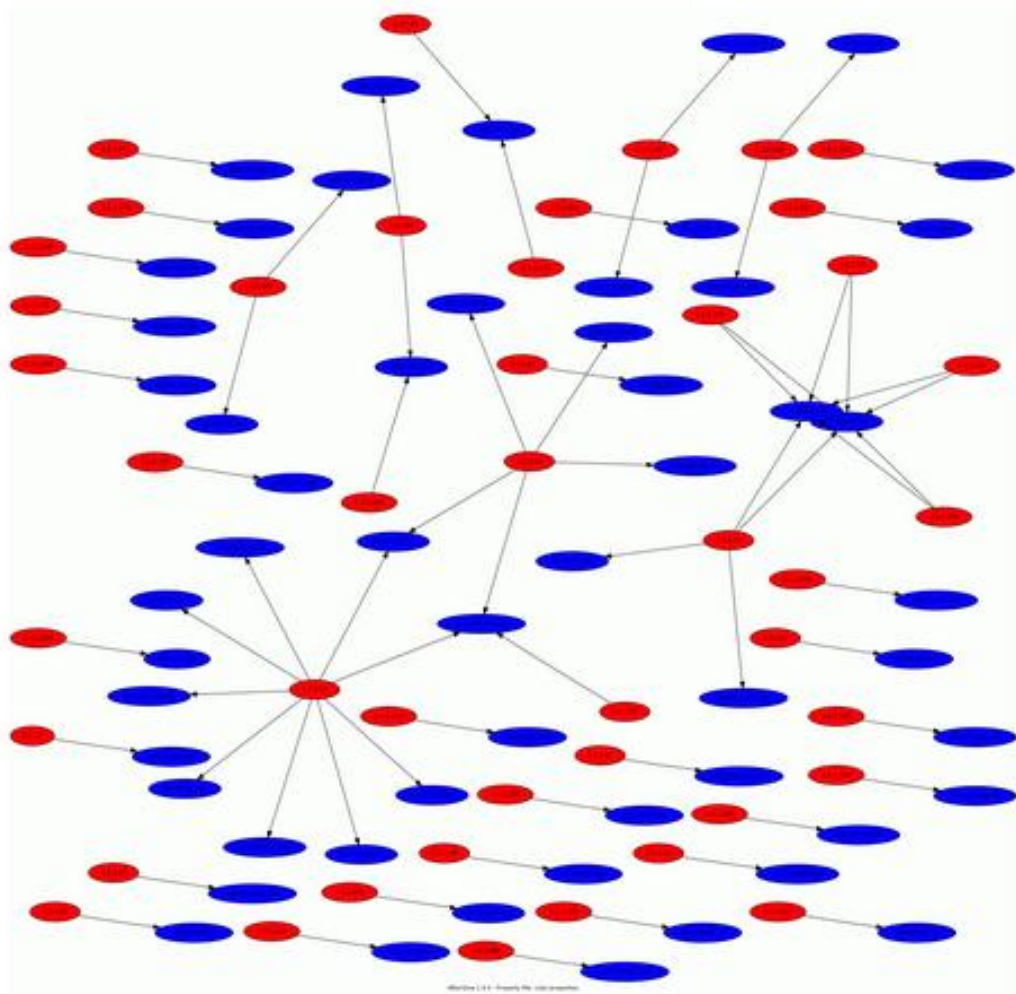


Figura 12: Trozo de tráfico de 16:30 a 16:40 de un determinado día. Afterglow. 2012.

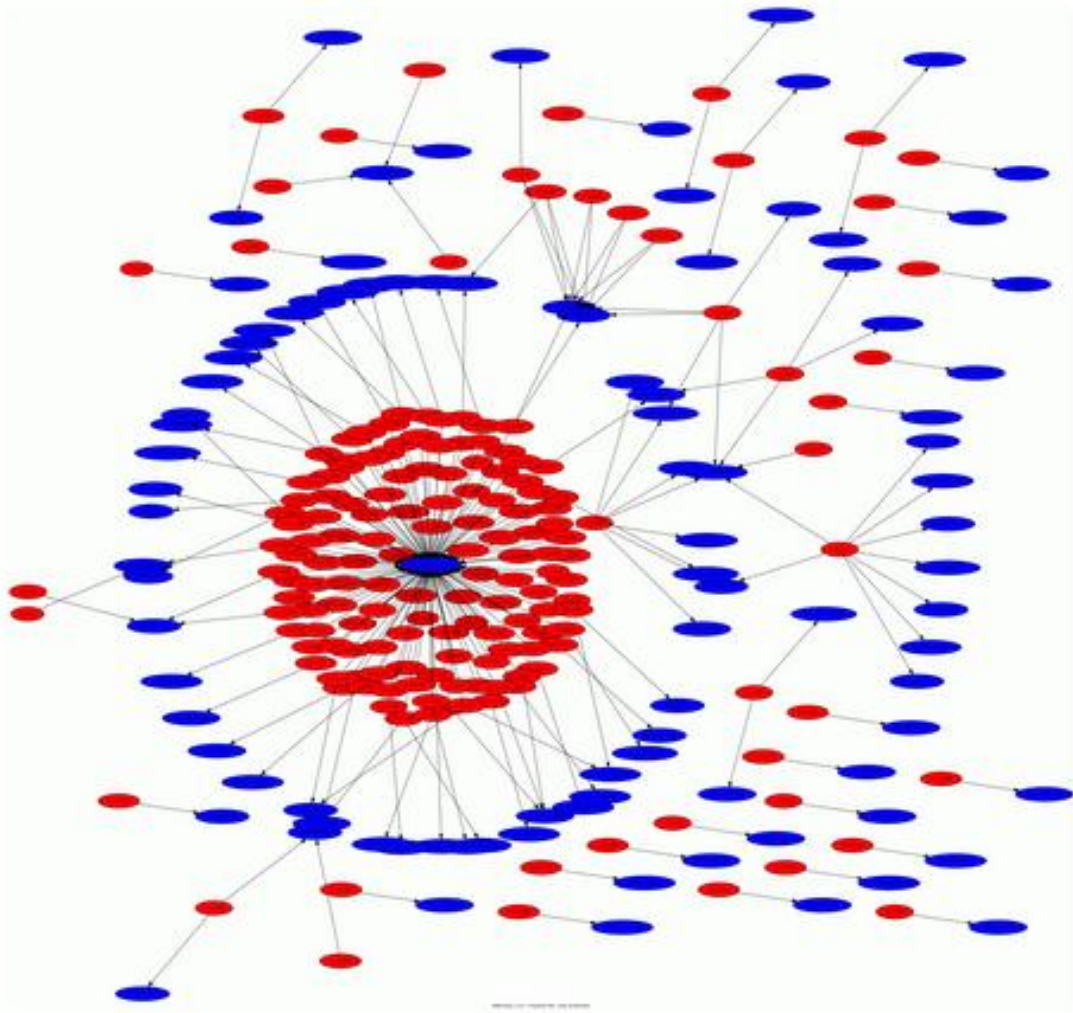


Figura 13: Trozo de tráfico de 16:30 a 16:50 de un determinado día. Afterglow. 2012.

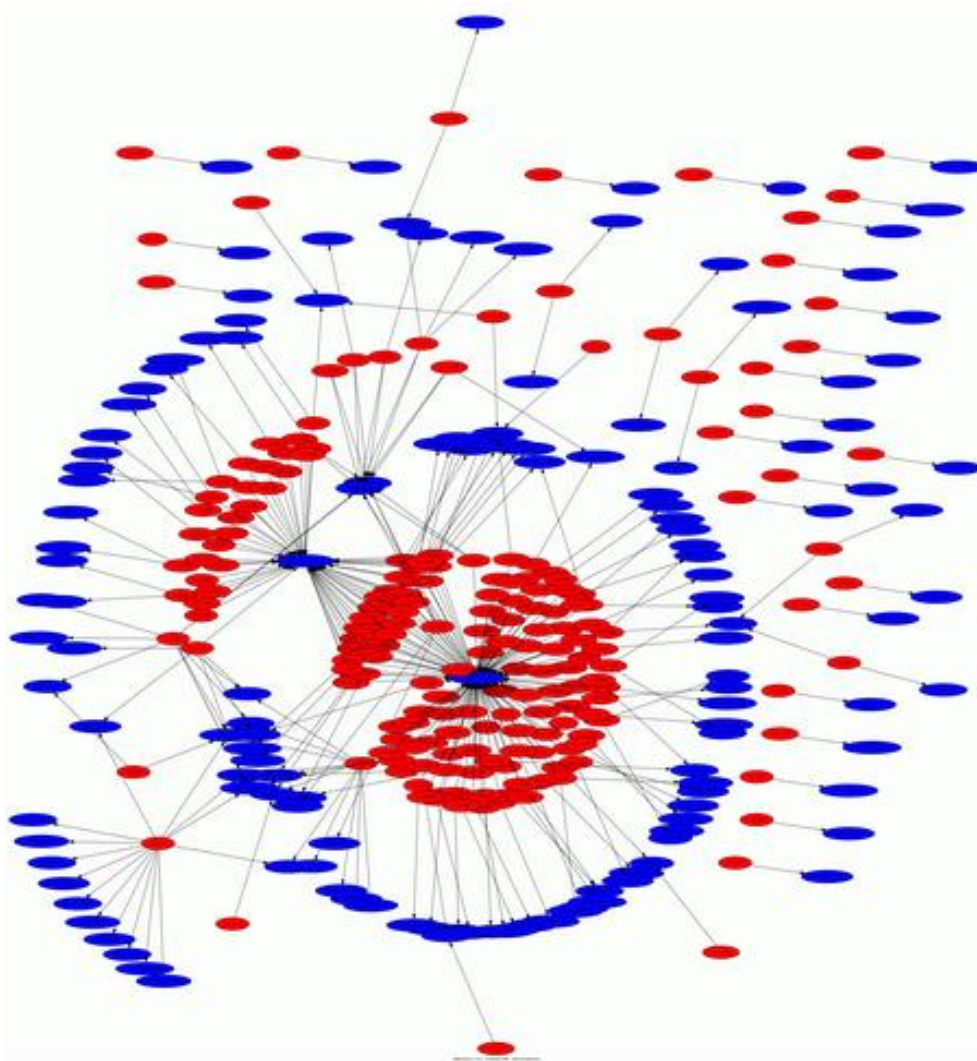


Figura 14: Trozo de tráfico de 16:30 a 16:59:59 de un determinado día. Afterglow. 2012.

Tal como se puede apreciar en la Figura 12, Figura 13 y Figura 14, los nodos de color rojo son las IP de destino (en este caso, las direcciones IP de la Darknet) mientras que los nodos azules son las direcciones IP de origen, probablemente comprometidas.

Esta herramienta tiene relación con uno de los malwares mencionados anteriormente: gusanos, específicamente aquellos que hacen escaneo de direcciones IP. Si existe el caso en que un atacante está revisando varias direcciones IP pertenecientes a la darknet ello se visualizará de forma intuitiva con AfterGlow, formándose patrones que llamarán la atención del analista (se verá un nodo de la malla representando a una dirección IP atacante que tiene lazos con otros nodos, correspondientes a varias IPs

de la darknet). Otros gusanos o malware se pueden identificar visualizando este tipo de diagramas, como es el caso de los ataques detectados en una Honeynet.⁷

Se implementó un script, llamado *afterglowMin.sh*⁸, cuya función es calcular la fecha de ayer y de definir los cortes de tiempo necesarios para graficar, es decir, por cada media hora existente en el día se envía la orden de generar tres archivos pcap. A modo de ejemplo, si se llama “hh” a una determinada hora del día, se solicitará la generación de un archivo que contiene el tráfico existente desde las hh:00 hasta las hh:10, otro desde las hh:00 hasta las hh:20 y otro desde las hh:00 hasta las hh:30 (ocurre lo mismo para la siguiente media hora, se generan archivos pcap desde las hh:30 hasta las hh:40, hasta las hh:50 y hasta las hh:59:59). Finalmente *afterglowMin.sh*⁹ hará las llamadas necesarias a otro script (*afterglow.sh*), quien será el encargado de graficar utilizando el software AfterGlow.

El script *afterglowMin.sh* funciona como sigue:

1. Se calcula la fecha del día de ayer.
2. Se determina el nombre y ubicación del archivo de tráfico al que se le deben hacer los cortes. Además se le hace una copia para trabajar con él.
3. Para cada hora del día se necesitan 6 archivos de tráficos: 3 por cada media hora. Por lo tanto, para cada uno de ellos se llama al script encargado de graficar (*afterglow.sh*, explicado más adelante) con el archivo de tráfico diario, y las horas en las que se quiere hacer el corte.
4. Finalmente se elimina la copia del archivo de tráfico cortado.

Por su parte, el script *afterglow.sh* realiza los siguientes pasos:

1. Se calcula la fecha del día de ayer (este dato será utilizado al final para ordenar las imágenes en carpetas diarias).
2. Se ingresa al directorio y elimina todo posible gráfico existente que tenga el mismo nombre del que se generará, para evitar copias.
3. Se realiza un llamado al software Tcpslice [TCPS10] quien es el encargado de generar los pcap de los tiempos solicitados a partir del archivo de tráfico diario, el de ayer. Los parámetros que recibe son \$2: archivo de ayer, \$3: hora inicio del corte, \$1: hora final del corte. Estos parámetros son definidos tal como se explicó anteriormente por *afterglowMin.sh*. Luego genera el archivo *tcpslice_\$1_\$2_\$3.pcap* que se utilizará como dato de entrada para graficar.

⁷ Red de características similares a las de una darknet, a excepción de que genera respuestas tras las solicitudes que recibe. Ejemplos de malware disponibles en: <http://cipherdyne.org/psad/honeynet/scan34/> y <http://cipherdyne.org/psad/honeynet/scan30/>.

⁸ Código disponible en apéndice A.

⁹ Código disponible en apéndice B.

4. Para evitar la publicación de las IPs de la darknet se modifican mediante una máscara utilizando el software tcprewrite [TCPRW10], el cual se encarga de dejar solo el último octeto de la IP sin alterar.
5. Se ejecutan los programas AfterGlow (afterglow.pl) y Tshark [TSHA10] especificando que se desean observar las IP de origen y de destino. Esto genera un archivo .dot con la información a ser graficada.
6. Luego el software Neato [NEAFT10] es el encargado de generar el gráfico como una imagen .gif a partir del archivo .dot.
7. Se crean una serie de directorios para ir almacenando los gráficos por día de manera ordenada. A continuación, se cambia el formato de la imagen de GIF a PNG utilizando convert [CONV10] y se borra el archivo con extensión GIF.

Mediante una tarea programada, Afterglow corre todos los días a las 00:05 de la madrugada, de manera que el archivo que contiene el tráfico del día de ayer ya se encuentre almacenado y disponible para su uso.

4. a.2. Moncube:

Moncube [MON10] es otro sistema capaz de generar gráficos pero simulando el tráfico dentro de un cubo que se puede mover, rotar, acercar y alejar entre otras propiedades. Para este proyecto se toman fotos a este cubo en distintas posiciones para visualizar de mejor manera lo que ocurre mediante imágenes.

Al igual que el software anterior, se toma el archivo de tráfico del día de ayer para generar las imágenes. Con este sistema se generan 3 imágenes diarias, todas son fotos del cubo en distintas posiciones.

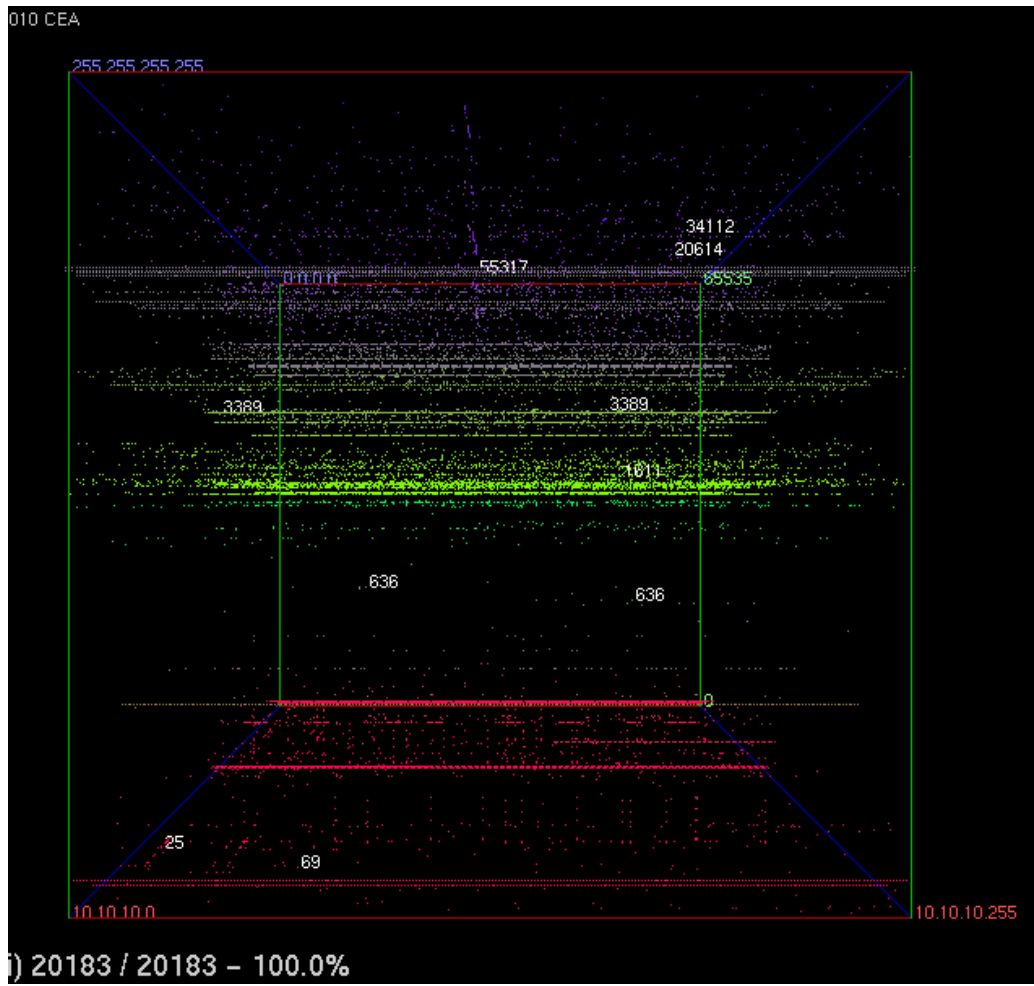


Figura 15: Cubo en primera posición. Moncube. 2012.

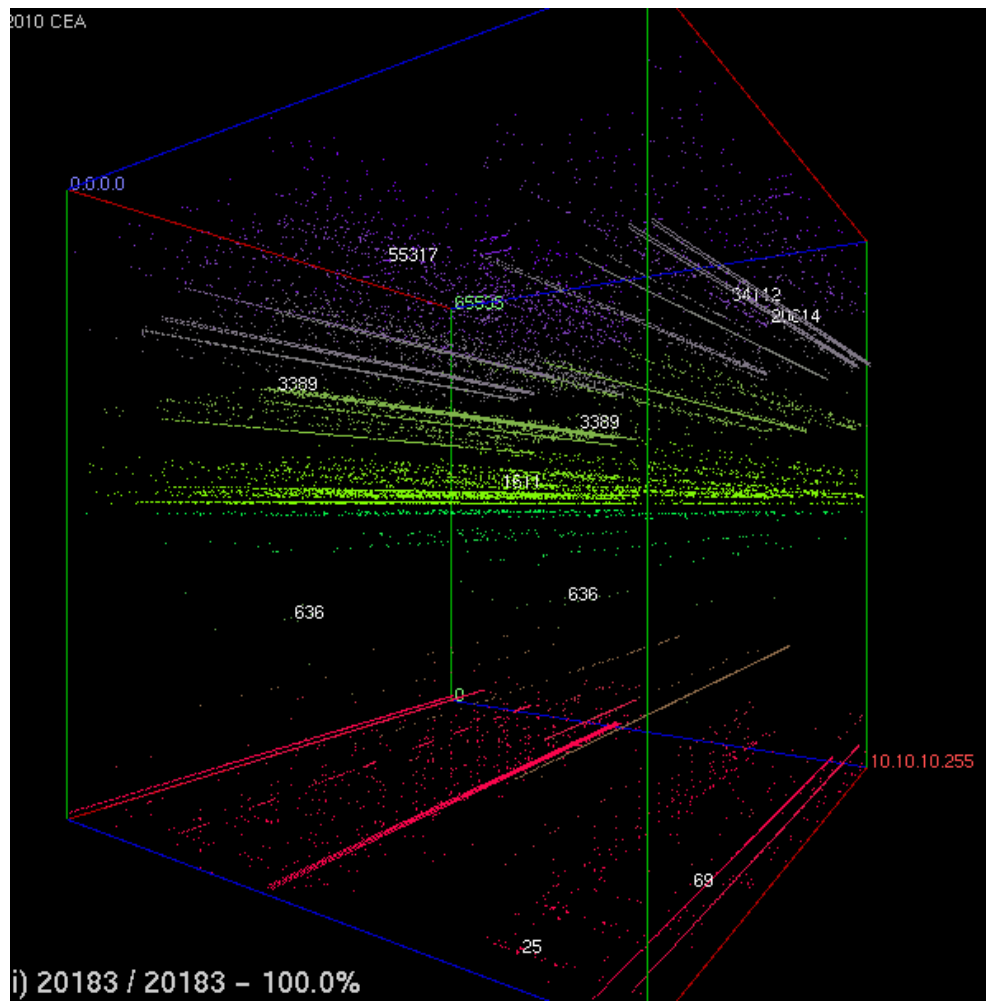


Figura 16: Cubo en segunda posición. Moncube. 2012.

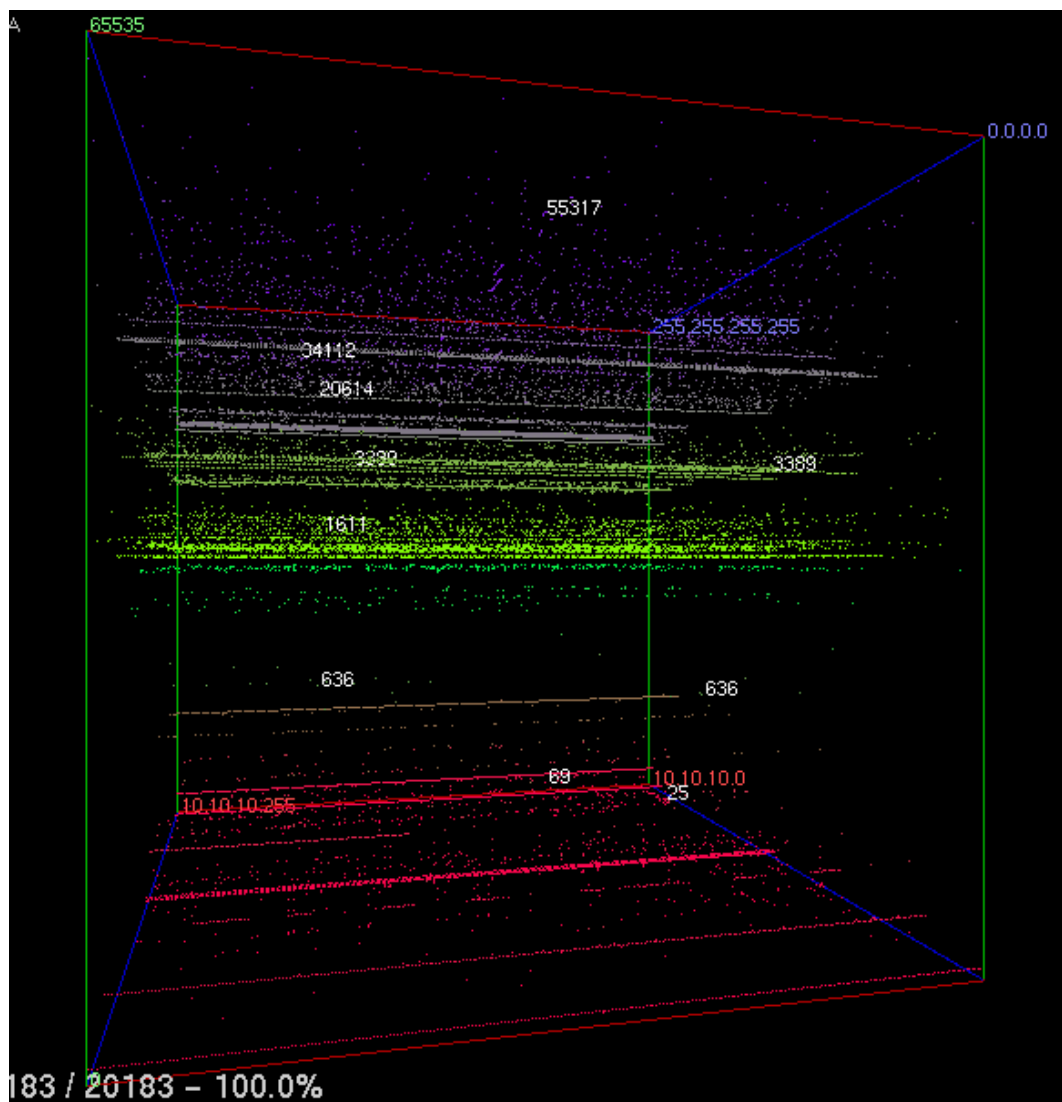


Figura 17: Cubo en tercera posición. Moncube. 2012.

El eje rojo son las IPs de destino (darknet, simbolizadas por las IP de la forma 10.10.10.X), el eje azul equivale a las IPs de origen, y el eje verde corresponde a los puertos de destino.

Este software se relaciona con los ataques de escaneo de puertos e IPs (gusanos) y denegación de servicios dado que se formarán patrones en los distintos ejes según sea el caso (líneas paralelas al eje rojo en caso de un escaneo de IPs, al eje verde en caso de escaneo de puertos, y al eje azul en caso de una denegación de servicios). Cabe destacar su importancia dado que ayuda a detectar variados tipos de ataques.

Para ejecutar la aplicación deben configurarse diversos parámetros tales como: el archivo de entrada a leer, el formato del archivo, el rango de IPs de destino (IPs de la Darknet), colores etc. Existe un script,

llamado *moncubeMin.sh*¹⁰ encargado de calcular el la fecha del día de ayer y de solicitar a otro script (*moncube.sh*¹¹) la generación de las imágenes.

El script *moncube.sh*, encargado de hacer la mayor parte del trabajo, ejecución de Moncube y captura de imágenes, realiza los siguientes pasos:

1. Se calcula la fecha del día de ayer para utilizarlo al final para ordenar las imágenes en carpetas diarias.
2. A continuación se implementa la función encargada de capturar imágenes del cubo mediante la tecla “w”, definida por Moncube para sacar fotos. También se usa la tecla “r” para iniciar y terminar la rotación del cubo.
3. Se ingresa al directorio donde está el ejecutable de Moncube y se borran archivos con nombres que podrían crear copias.
4. Se cambia la IP de la Darknet para mantenerla en secreto (IP de destino) y se define como 10.10.10.0 generado un nuevo archivo. Para esto se usa la herramienta *tcprewrite* a partir del archivo de tráfico de ayer, dado como parámetro \$1.
5. A partir del archivo generado en el paso anterior, se genera uno nuevo que contiene el formato que Moncube necesita para poder graficar. Esto se hace mediante *tcpdump*.
6. Se llama al ejecutable “*parsecube*” pero también llamando a la función creada en el paso 2 para que corra al mismo tiempo y capture las imágenes.
7. Se crean los directorios necesarios para tener las imágenes ordenadas.

Moncube se corre a las 00:01 para que el archivo del tráfico de ayer se encuentre disponible. Este es el primer software en correr, y trabaja solo ya que utiliza más recursos que otros de los utilizados.

Lo interesante de este software es que a través de las imágenes se pueden observar ciertos patrones que pueden indicar ciertos ataques o comportamientos peligrosos. Los desarrolladores de Moncube hicieron una lista de imágenes y sus posibles significados para contribuir en el estudio de la seguridad, esta información se encuentra en la referencia especificada anteriormente [MON10] bajo el link llamado “Scan patterns”.

4. a.3. TCPstat:

TCPstat [TS10] es una herramienta que permite la generación de gráficos en dos dimensiones. Permite graficar cuántos paquetes por segundo llegan a la red durante un determinado tiempo. A partir de esto se generan dos tipos de gráficos, uno donde se indican los protocolos de los paquetes, y otro donde se grafican dos curvas: una que representa a los paquetes con cualquier flag activada y otra con solamente

¹⁰ Código disponible en apéndice C.

¹¹ Código disponible en apéndice D.

los paquetes cuya flag SYN-ACK está activada, en caso de que existan (antes de entrar a la darknet los paquetes SYN-ACK son filtrados por el Servicio de Tecnologías de la Información), ya que la llegada de este tipo de paquetes es una señal de un ataque de denegación de servicios. Esto es porque un atacante puede enviar una solicitud (envío de paquete SYN) a un determinado equipo poniendo como dirección de origen alguna IP perteneciente a la darknet. De esta manera el equipo vulnerado le responde a la darknet con paquetes SYN-ACK. Dado que la Darknet nunca intentó comunicarse con alguien, existe la sospecha de que existen computadores siendo vulnerados mediante un ataque de denegación de servicios (envío múltiple de paquetes esperando respuesta para que se le acaben los recursos al atacado).

Estos son algunos de los gráficos que se pueden obtener:

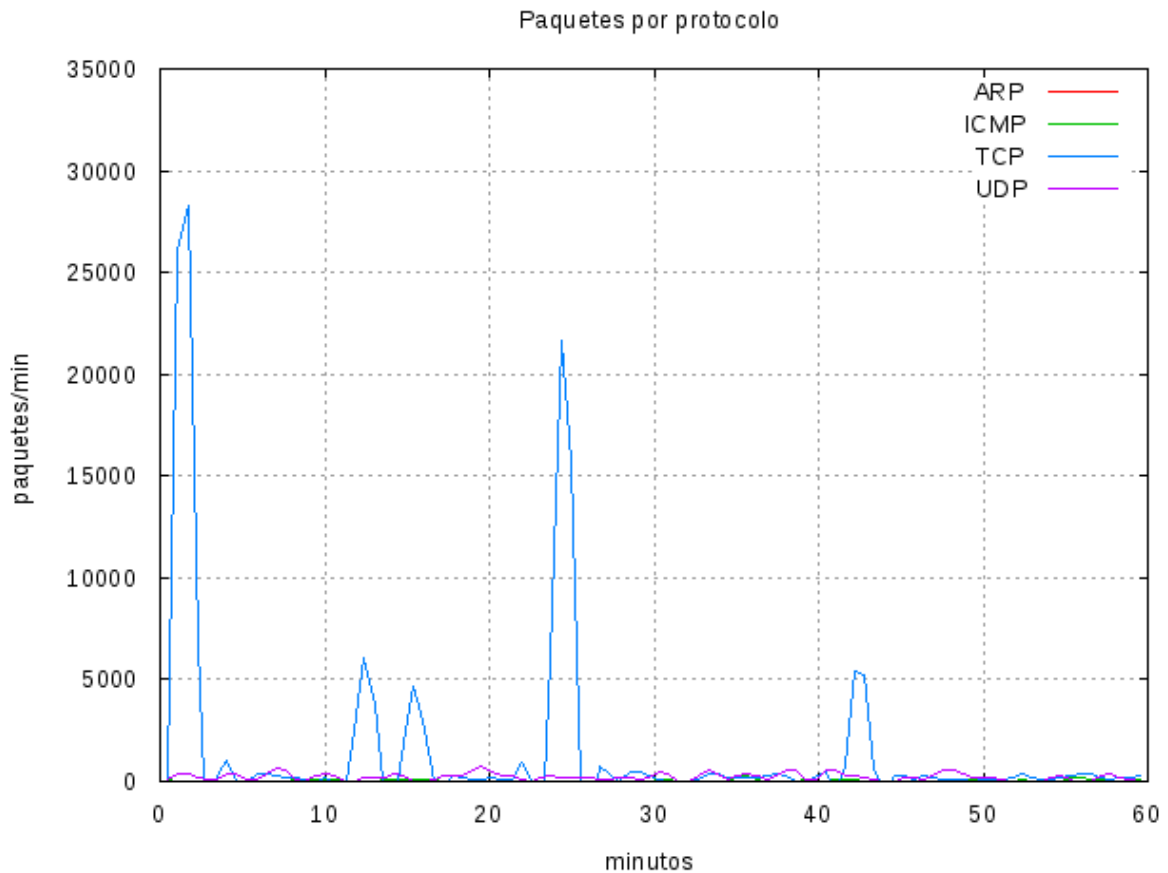


Figura 18: Paquetes por protocolo durante una hora. TCPstat. 2012.

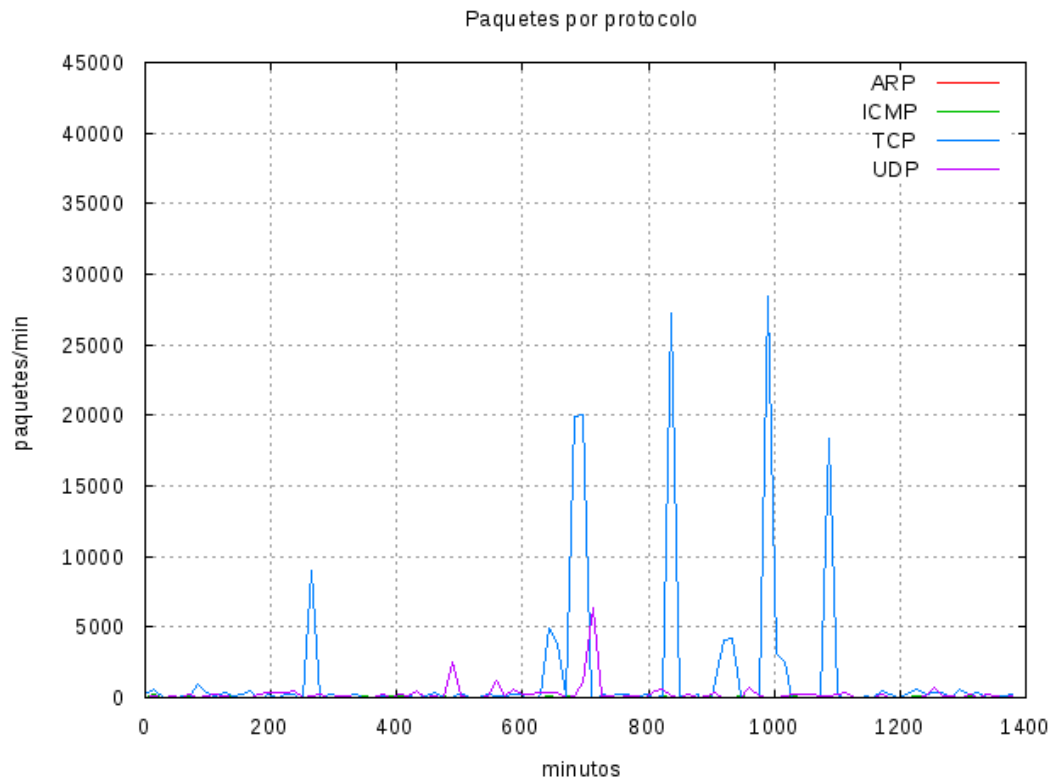


Figura 19: Paquetes por protocolo durante un día. TCPstat. 2012.

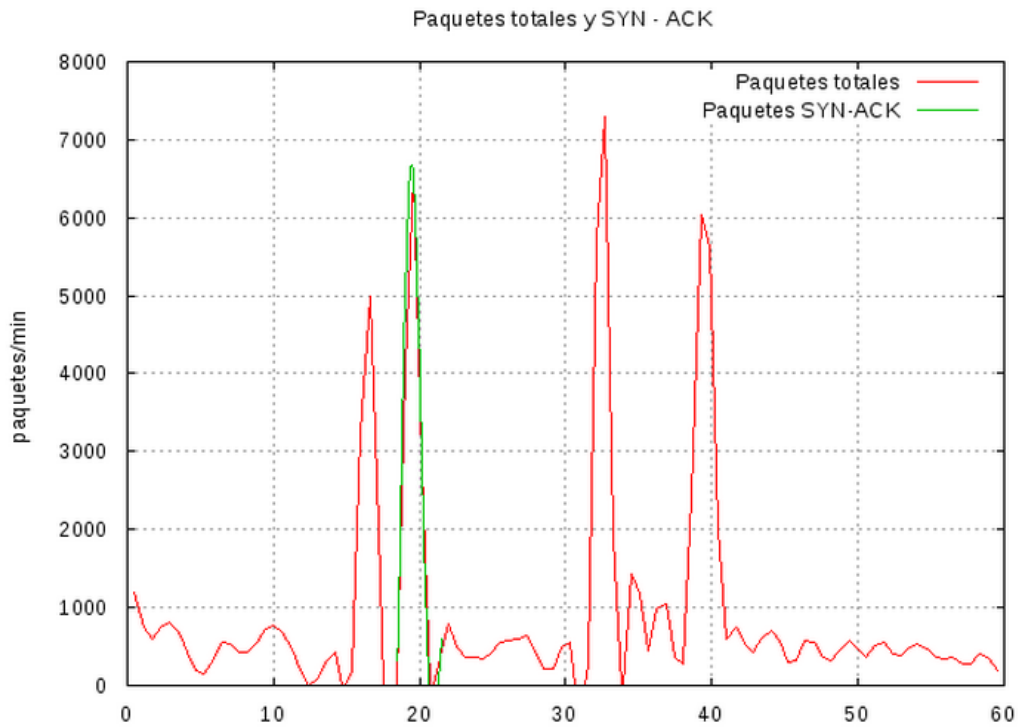


Figura 20: Paquetes totales durante una hora con tráfico SYN-ACK ficticio. TCPstat. 2012.

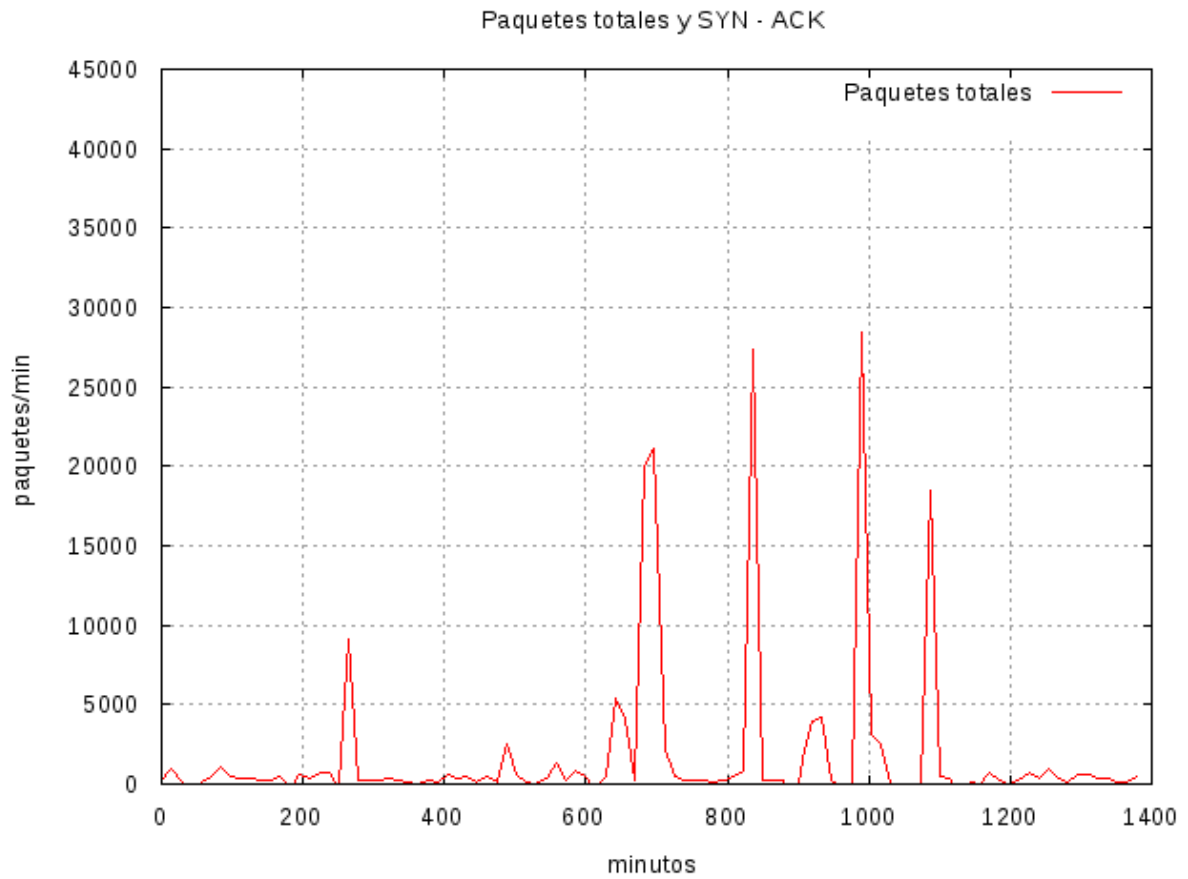


Figura 21: Paquetes totales durante un día. TCPstat. 2012.

Con los dos primeros gráficos (protocolo de paquetes) se podrían detectar ataques de Denegación de Servicios. Esto es porque la posible respuesta dirigida a la darknet desde los servidores DNS utilizando el protocolo UDP puede ser visualizado en este diagrama, tal como ocurre en el gráfico “Accesos por protocolo” mencionado con anterioridad.

Tal como se explicó anteriormente estos últimos gráficos (SYN-ACK) están fuertemente relacionados con la detección de una denegación de servicios. Pero además esta herramienta se vincula al reconocimiento de escaneo de IPs, porque si se observa este malware en a otro gráfico (AfterGlow o IPs en el tiempo) gracias a TCPstats se puede determinar si el ataque va dirigido a la darknet, o son otras máquinas comprometidas las que se están comunicando con ella. En este último caso habría existencia de paquetes SYN-ACK mientras que en el primero no.

Para la generación del gráfico de paquetes según protocolo se realizó un script (*TCPstats_protocolMin.sh*¹²) que se encarga de cortar el archivo de tráfico del día de ayer en pequeños archivos que contienen el tráfico de una hora del día cada uno. Luego se llama a otro script

¹² Código disponible en apéndice E.

(*TCPstats_protocol.sh*¹³) encargado de graficar (tanto lo que ocurre cada hora como lo que ocurre en el día en su totalidad).

El procedimiento que realiza el script *TCPstats_protocolMin.sh*, encargado de los cortes del archivo de tráfico, es el siguiente:

1. Se calcula la fecha de ayer.
2. Se utiliza el nombre del archivo de tráfico y se copia en el directorio de tcpstat para trabajarlo.
3. Para cada hora del día se crea un archivo de tráfico que contiene solamente los paquetes recibidos por la darknet esa hora.
4. Luego se llama al siguiente script (*TCPstats_protocol.sh*) que se encargará de graficar utilizando como entrada el tráfico de la hora asignada.
5. Finalmente se le entrega al script encargado de graficar (*TCPstats_protocol.sh*) una copia de todo el tráfico del día anterior para generar un gráfico diario.

El script encargado de la generación de gráficos utilizando la herramienta TCPstat (*TCPstats_protocol.sh*) realiza el siguiente proceso:

1. Se calcula la fecha del día de ayer.
2. Se ingresa al directorio adecuado y se eliminan archivos que contengan nombres que podrían generar copias.
3. Se llama a tcpstat con el archivo de tráfico adecuado (ya sea de una hora o diario) y se filtran los paquetes por protocolo, sacando un promedio de paquetes vistos en los últimos 60 segundos para cada protocolo. Este paso genera archivos .data separados por protocolo.
4. Luego se llama a *gnuplot.script*¹⁴, que se explicará a continuación, para generar el gráfico final a partir de los archivos .data creados en el paso anterior. Se generan imágenes .png.
5. Se guardan los archivos usados y creados por día en sus respectivas carpetas.

El script *gnuplot.script*, mencionado anteriormente, es el encargado de graficar, definir escalas y poner etiquetas al gráfico. Este script define el título del gráfico, el nombre del eje X e Y, entre otros aspectos.

Para generar gráficos a partir de paquetes que tengan la flag "SYN ACK" prendida se utilizan scripts, similares a los anteriores. El análogo a *TCPstats_protocolMin.sh*, explicado anteriormente, se llama *TCPstats_SYNACKMin.sh*¹⁵ y el similar a *TCPstats_protocol.sh* es *TCPstats_SYNACK.sh*¹⁶.

¹³ Código disponible en apéndice F.

¹⁴ Código disponible en apéndice G.

¹⁵ Código disponible en apéndice H.

Los pasos que realiza el script *TCPstats_SYNACKMin.sh* son los siguientes:

1. Se calcula la fecha de ayer
2. Conociendo el nombre del archivo de tráfico, se copia en el directorio de *tcpstat* para trabajarlo.
3. Para cada hora del día se crean un archivo de tráfico que contiene solamente los paquetes de esa hora.
4. Luego se llama al siguiente script (*TCPstats_SYNACK.sh*) que se encargará de graficar utilizando como entrada el tráfico de la hora asignada.
5. Se le entrega al script encargado de graficar (*TCPstats_SYNACK.sh*) una copia de todo el tráfico del día anterior para generar un gráfico diario.

El script *TCPstats_SYNACK.sh* es el encargado de la generación de gráficos y actúa de la siguiente manera:

1. Se calcula la fecha del día de ayer
2. Se ingresa al directorio adecuado y se eliminan archivos que contengan nombres que podrían generar copias.
3. Para este caso se grafican 2 curvas, una que hace referencia a todos los paquetes y otra que tiene solo aquellos paquetes con la flag "SYN-ACK" prendida. Para esto último se aplica un filtro de *tcpdump* que revisa la flag en cuestión. Se generan archivos *.data* tal como en el caso anterior.
4. Luego se llama a *gnuplotACK.script*¹⁷, que se explicará a continuación, para generar el gráfico final a partir de los archivos *.data* creados en el paso anterior. Se generan imágenes *.png*.
5. Se guardan los archivos usados y creados por día en sus respectivas carpetas.

Finalmente el script *gnuplotSYNACK.script*, que se encarga de graficar y poner etiquetas al gráfico. Es similar a *gnuplot.script* solo que ahora se grafican todos los paquetes en una curva, y los SYN-ACK en otra (en caso de que existan, si no hay la curva no aparecerá.)

Tanto los scripts encargados de generar gráficos por protocolo de paquetes en tiempo, como los encargados de graficar paquetes SYN-ACK en el tiempo, corren a las 00:02 utilizando la información del tráfico del día anterior.

¹⁶ Código disponible en apéndice I.

¹⁷ Código disponible en apéndice J.

4. a.4. EtherApe:

Este software [EA10] permite generar un video tomando como entrada un archivo pcap de tráfico. EtherApe crea una simulación “en tiempo real” de las conexiones existentes entre el exterior y la Darknet.

Se lanza EtherApe para que realice la simulación mediante un video, por otro lado se filma el sector de la pantalla donde el software está corriendo. De esta manera se logra capturar un video de la simulación. La Figura 22 muestra una captura de pantalla de la ejecución de EtherApe.

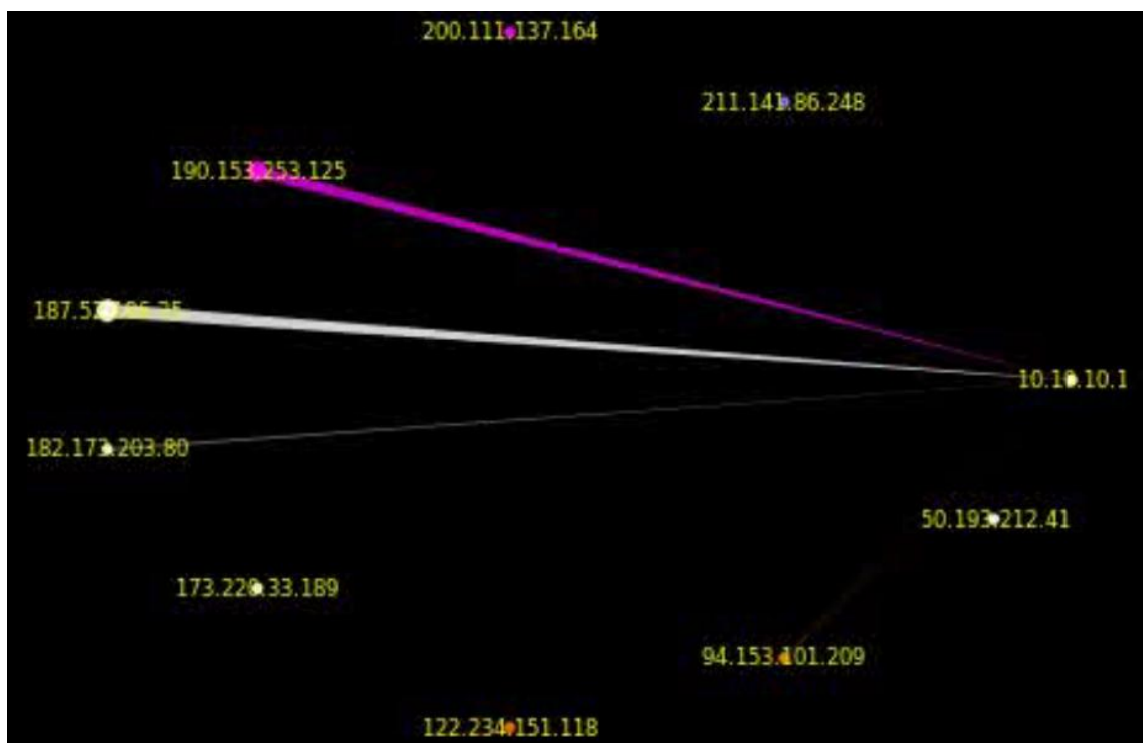


Figura 22: Captura de pantalla de EtherApe en ejecución. 2012.

Este software corre cada 15 minutos y crea un video de los últimos 10 minutos transcurridos, con un desfase de 5 minutos (tiempo dado para la correcta creación del video). Existe un script, llamado *ether.sh*¹⁸, encargado de crear cortes en el archivo de tráfico del día actual, y generar pequeños pcap con la información de lo ocurrido en los 10 minutos mencionados.

Los pasos del script *ether.sh* se detallan a continuación:

1. Cálculo de la fecha actual.
2. Se copia el archivo de tráfico en el directorio de EtherApe.

¹⁸ Código disponible en apéndice K.

3. Se define la hora de inicio y de fin de cada video. La de inicio será la de hace 15 minutos atrás, y la de fin será la de hace 5 minutos atrás. De esta manera se obtendrán simulaciones de 10 minutos de duración. Recordar que este script se lanza cada 15 minutos.
4. Luego se realizan los cortes para tener pequeños archivos de tráfico que contienen lo ocurrido en los 10 minutos de tráfico definidos.
5. Se mueve el archivo generado al directorio de EtherApe.
6. Se llama al siguiente script encargado de lanzar EtherApe, llamado *ethMin.sh*¹⁹ con los parámetros necesarios.

El script *ethMin.sh* es el que hace correr EtherApe y filmar el video, a continuación se presenta su procedimiento:

1. Cálculo del día actual.
2. Se desea filmar la ventana de Etherape, entonces en caso de que éste se cierre, no se desea que el script siga capturando video al fondo del Escritorio. Es por esto que se crea una variable que determina si el proceso “etherape” sigue vivo, luego este valor es utilizado como condición en la toma de video.
3. A continuación se implementa la función encargada de filmar cuando EtherApe está corriendo. Acá se calcula la posición de la ventana de EtherApe para filmar en las coordenadas correctas (utilización del comando “sed” [SED10]).
4. Se mueve el archivo de tráfico al directorio necesario, se entra en él y se eliminan archivos que podrían crear duplicidades.
5. Se crea un nuevo archivo dándole a todos los paquetes la misma dirección de destino de la darknet, simulada como 10.10.10.10.1 en este caso. Esto se hace a partir del archivo de entrada que contiene tráfico de los últimos 10 minutos (con un desfase de 5 minutos).
6. Etherape corre justo antes del método encargado de la captura de video para asegurarse de que el video está visible.
7. Se crean los distintos directorios para organizar las imágenes, videos y archivos de tráfico.

Este sistema corre todos los días tal como los demás, pero cada 15 minutos para así obtener una visión “on line” de lo que está ocurriendo en la Darknet.

Para el lanzamiento automático de todas estas tareas se utiliza “Configured Scheduled Tasks” de Gnome, que funciona como “cron” pero con la posibilidad de ejecutar comandos que necesiten interfaz gráfica, tal como es el caso de Etherape y Moncube.

¹⁹ Código disponible en apéndice L.

4. a.5. Copiado el material al servidor web y borrado del antiguo:

Cada día se copian las imágenes y los videos desde el computador generador de gráficos al servidor web para el despliegue de los gráficos en el sitio utilizando el comando “scp”. Esto se realiza mediante unos scripts existentes en el servidor web que corren unos minutos después de que los gráficos ya se hayan creado en el computador del laboratorio. Para el caso de los videos, el copiado se realiza cada hora.

Además, existen scripts encargados de borrar videos y gráficos antiguos del computador del laboratorio, siempre y cuando estén en el servidor web. De esta manera se mantiene el equipo generador de gráficos limpio y sin redundancias con el servidor web. Así se evita que existan alertas tales como que el disco se encuentra lleno, lo que impide la ejecución de las tareas programadas y por lo tanto la creación de gráficos.

4. b. Implementación de herramienta para la creación de gráficos ad-hoc

Cada noche se crean 10 archivos XML, cada uno de ellos corresponde a uno de los gráficos que luego Maani Charts desplegará mediante Flash en la página web. A continuación se describirá el proceso asociado a esta creación por partes.

4. b.0. Captura de pcap de ayer:

Cada día a las 00:00 la darknet envía un archivo con el tráfico del día anterior. Éste queda almacenado en el Desktop del computador del laboratorio, dentro de una carpeta llamada capturas. El nombre de este archivo será “salida-” concatenado con a fecha del día asociado en formato YYYY-MM-DD.

4. b.1. Listas de IPs y puertos:

La primera tarea es detectar las direcciones IPs que enviaron paquetes a la darknet. Una vez que el archivo de tráfico del día anterior ya fue recibido por el computador del laboratorio corre este script llamado *tiraListas.sh*²⁰ cuyo procedimiento es el siguiente:

1. Primero se calcula la fecha del día anterior y se le concatena “salida-” para utilizar el archivo recién recibido.
2. A continuación se llama a otro script, llamado *generadorListas.sh*²¹ (se explicará más adelante) cuyo resultado serán 2 archivos, uno contendrá todas las IPs (como columna) que accedieron a la darknet, y el otro una columna con todos los puertos visitados. En estas listas, al lado de cada IP o puerto (en una segunda columna) aparecerá la cantidad de veces que determinada IP envió un paquete o el número de veces que cierto puerto fue accedido, según sea el caso. Los datos de las

²⁰ Código disponible en apéndice M.

²¹ Código disponible en apéndice N.

listas quedan en orden descendente dependiendo de la cantidad de accesos. Estas listas quedarán disponibles en el directorio de capturas.

3. Las listas resultantes del script mencionado se cambian de directorio a una carpeta llamada "listas" ubicada también en el Desktop del computador del laboratorio. Además se les cambia el nombre concatenándole el nombre del archivo de tráfico recibido recientemente desde la darknet, para saber de qué día son las IPs y puertos accedidos.
4. Luego se guarda una copia de la lista de IPs con la cantidad respectiva de paquetes enviados a la darknet (será utilizada en otro momento).
5. Es de gran interés tener información sobre las IPs que acceden a la darknet, ya que de cierto modo, es la manera de "conocer" las máquinas comprometidas (cabe recordar que cualquier IP que accede a la darknet es sospechosa, dado que ella no responderá, ¿por qué alguien con intenciones honestas habría de querer contactarla?). Es por esto que se utilizará *whois*, servicio prestado por el Team Cymru [WHO11], para tener información a partir de las direcciones IP. Este programa recibe como entrada el archivo que contiene la lista de IPs (junto con su cantidad de accesos) con las líneas de inicio y fin estipulados. El resultado será un nuevo archivo con 4 columnas: dirección IP, cantidad de paquetes enviados a la darknet, país de origen e ISP.
6. En el manual de uso de este servicio se indica que, en caso de querer preguntar por más de una dirección IP, se debe crear un archivo con cierto formato que contenga la lista de IPs de interés y ejecutar *whois* con él como parámetro. Se procede a crear este archivo con el formato determinado: Primero el archivo con la lista de IPs debe tener la palabra "begin" en la primera línea y "end" en la última. Además en caso de querer conocer el país de origen de la lista de direcciones IP a enviar, en la segunda línea debe ir la palabra "countrycode".
7. A continuación se utiliza *whois* que entrega como resultado un archivo llamado IPdatossalida-2012-01-02.txt (por ejemplo) contenedor de las cuatro columnas con información.
8. Ahora esta lista se depura para que luego pueda ser desplegable en la página web. Para esto se elimina la primera línea que contiene qué significa cada columna, se borran los resultados vacíos que el servicio *whois* no pudo descifrar y se elimina la última línea que, en algunos, casos viene cortada. En este punto se realiza una copia de la lista y se guarda como *paises\${archivo}.txt* (dependiendo del archivo procesado), su uso será explicado más adelante.
9. Luego, sobre la lista original (no la copia guardada en *paises\${archivo}.txt*) se seleccionan las 1000 primeras líneas (para que el archivo no sean demasiado largo). Entonces de esta manera se tendrá una "tabla" con los datos de las 1000 IPs (cómo máximo) que más han accedido a la darknet, en orden descendente en cuanto a accesos diarios.
10. Se realiza un procedimiento similar para la lista de puertos (y sus accesos). Se eliminan las listas de puertos de tipo "None" (cuando el protocolo de la conexión es ICMP no se conoce el puerto accedido), se rescatan las 1000 primeras líneas (como máximo) y se le cambia el nombre a Puertosdatossalida-2012-01-02.txt, por ejemplo.
11. Se eliminan archivos que ya no se utilizan.

12. Dado que ya se tienen las listas con las IPs y puertos más accedidos, se está en condiciones para llamar al creador de gráficos (de XMLs) llamado *graficosFlash.sh*²², dado que éste las necesitan como parámetro. El único gráfico que no se crea en este script es el de los 10 países que acceden más a la darknet (será creado en el punto siguiente por *países.sh*), esto se debe a que se realiza de forma diferente al resto de los gráficos, dado que solamente procesa 10 líneas de información (10 países con sus accesos).
13. Se ingresa a un directorio específico y se llama al script *países.sh*²³ para la creación del gráfico de estado de los 10 países que realizan más accesos a la darknet. Este script utiliza la lista almacenada en el archivo *países\${archivo}.txt*, mencionado anteriormente, que contiene una copia del resultado entregado por el servicio whois.

El proceso escrito recientemente se puede apreciar en el Diagrama 3:

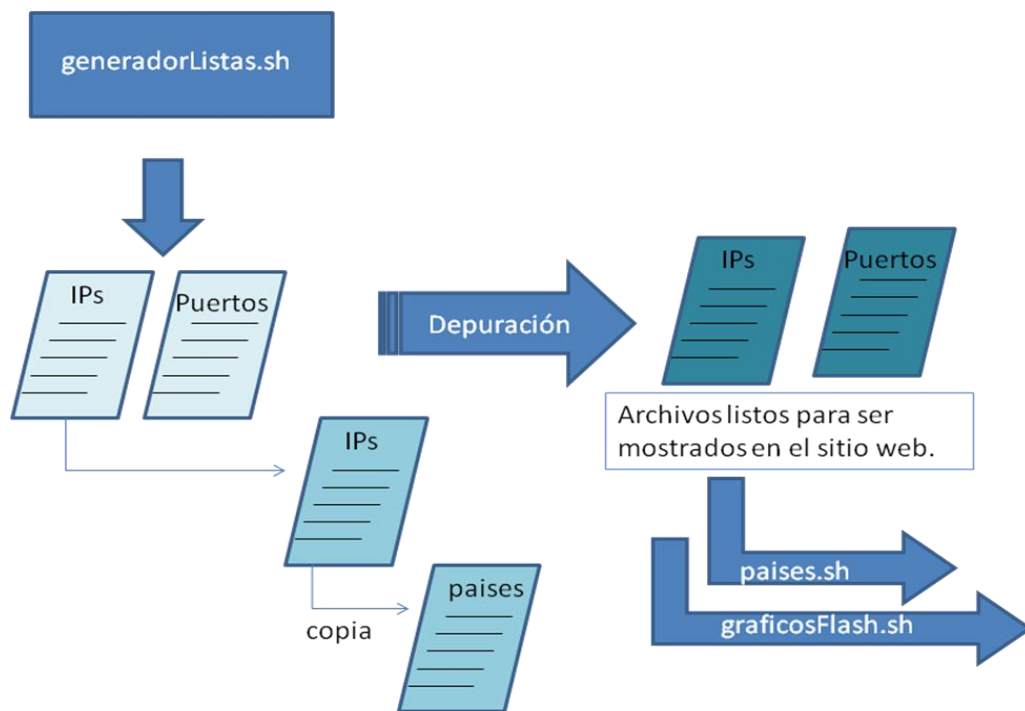


Diagrama 3: Representación del proceso de creación de listas finales de IPs y puertos. 2012.

Antes de ver el detalle de cómo se crean los gráficos (*graficosFlash.sh* y *países.sh*) primero de explicará sobre el generador de listas que se mencionó con anterioridad. El script *tiraListas.sh* hizo una llamada a *generadorListas.sh* con los parámetros `00h00m00s 23h59m59s $archivo` y "all". A continuación se detallará el comportamiento de dicho script:

1. Se llama al programa TcpSlice [TCPS10] para que realice un corte del tráfico diario. Para esto se le entregan como parámetros la hora de inicio, la de fin y el archivo de tráfico que se desea

²² Código disponible en apéndice O.

²³ Código disponible en apéndice LL.

cortar. Cabe recordar que el primer parámetro de *generadorListas.sh* fue 00h00m00s, el segundo 23h59m59s y el tercero \$archivo (siendo éste el archivo de tráfico del día anterior). Por lo tanto en este caso no se está haciendo un verdadero corte, si no que se está tomando el día completo desde las 00:00:00 hrs hasta las 23:59:59. Esto se hace por seguridad, dado que el archivo de tráfico proveniente de la darknet se crea y se envía a las 00:00 (del día presente), pero ocurren ocasiones en que esto se realiza con algunos segundos de retraso, entonces se tiene tráfico desde las 00:00 del día anterior hasta las 00:00:xx del día siguiente en un mismo archivo, por lo tanto TcpSlice ayuda a limpiar esos segundos extras.

2. TcpSlice entrega como resultado un archivo binario contenedor del tráfico existente en el horario seleccionado.
3. A continuación se llama a Tcpcmdump [TCPDU10] para leer el tráfico que está en binario y poder manejarlo como archivo de texto. En este caso se llama con los parámetros -n (no convierte las direcciones) y -r para leer el archivo de salida generado por TcpSlice.
4. A continuación se le hacen algunos cambios al archivo de manera de que solamente queden columnas con datos.
5. El archivo resultante será del tipo (IPs de la darknet reemplazadas por 1.1.1.x):

```
00 00 49 643174 IP 218 86 103 28 29005 > 1 1 1 121 62214: UDP, length 30
00 00 54 301936 IP 190 22 101 23 1870 > 1 1 1 34 25: tcp 0
00 00 52 967747 IP 58 9 108 89 > 1 1 1 239: ICMP echo request, id 8, seq
38204, length 8
```

Tabla 1: Ejemplo de tráfico.

6. A continuación se almacenan los valores recibidos como parámetro y se hace un shift de los 3 primeros.
7. En este momento se llama a un programa en python encargado del procesamiento de datos. El parámetro que recibe es el archivo resultante a la llamada de Tcpcmdump. Además se tiene un directorio con logs para analizar en caso de errores. Esta llamada a python generará la lista de IP con su respectiva cantidad de paquetes enviados a la darknet.
8. Ahora, para generar la lista de puertos con sus respectivos accesos, antes de llamar al código en python se eliminan algunas líneas del archivo resultante de la llamada a Tcpcmdump. Éstas son las que contengan la palabra "ICMP" dado que los paquetes que llegan a la darknet con este protocolo no indican el puerto accedido.

A continuación se explicarán los códigos python utilizados en la creación de las listas.

listaIP.py	listaPuertos.py
<pre>from Main import * nada(sys.argv[1], "fileLimpio_listaIP.txt") listaIP()</pre>	<pre>from Main import * nada(sys.argv[1], "fileLimpio_listaPuertos.txt") listaPuertos()</pre>

Tabla 2: Códigos scripts de listaIP.py y listaPuertos.py.

Siempre que se crea una lista (o un gráfico como se verá más adelante) se realiza un procedimiento al archivo arrojado por tcpdump (texto). El método “nada” genera una copia del archivo original pero con otro nombre y formato. En este caso el archivo de entrada a éste método corresponde a sys.argv[1], (primer parámetro recibido por listaIP.py), el segundo parámetro (“fileLimpio_listaIP.txt” o “fileLimpio_listaPuertos.txt”) será el nuevo nombre²⁴ del archivo modificado.

En este método se abre el archivo (se manejan errores) y luego en cada línea se hace una separación por columnas cada vez que se detecte un espacio en blanco. De esta manera se podrán acceder a los datos de las columnas de la forma data[0], data[1], data[2] etc. Finalmente cada línea se copia en el archivo nuevo.

A continuación los métodos listaIP() y listaPuertos() son llamados por listaIP.py y listaPuertos.py respectivamente.

listaIP()	<pre>def listaIP(): map = mapGroup("fileLimpio_listaIP.txt", "mycount", "notChange", "listaIP") reduce13 = reduce(map, "sum") txt = printListaIP(reduce13) f = open("/home/rfaccilo/Desktop/capturas/IP.txt", "w") f.writelines(txt) f.close()</pre>
listaPuertos()	<pre>def listaPuertos(): map = mapGroup("fileLimpio_listaPuertos.txt", "mycount", "notChange", "listaPu") reduce14 = reduce(map, "sum")</pre>

²⁴ Quizás un mejor nombre para estos archivos serían datosFiltrados_listaIP.txt y datosFiltrados_listaPuertos.txt ya que contienen los datos, asociados a IPs y puertos, ya procesados por un filtro.

	<pre> txt = printListaPuertos(reduce14) f = open("/home/rfaccilo/Desktop/capturas/Puertos.txt", "w") f.writelines(txt) f.close() </pre>
--	--

Tabla 3: Códigos de los métodos listaIP() y listaPuertos().

En ambos casos se llama a map y a reduce. Finalmente se llama a *printListaIP* o *printListaPuertos* que reciben como parámetro los datos reducidos. Los tres métodos mencionados anteriormente son importantes, sobre todo map y reduce, por lo tanto se explicarán detalladamente a continuación.

El método mapGroup (map) se encuentra en la clase Maper.py²⁵ que es importada por Main.py (clase donde están los métodos nada, listaIP y listaPuertos), su principal tarea es crear un diccionario con el formato llave - valor. El map recibe cuatro parámetros, el primero es un archivo (contenedor de los datos), el segundo es la función “*func*”, el tercero un identificador, y el cuarto la función “*defval*”. El comportamiento del método map dependerá, principalmente, de las 2 funciones que recibe, además del identificador entregado como tercer parámetro. El procedimiento se detalla a continuación:

1. Primero se intenta de abrir el archivo recibido como parámetro. Se manejan los errores en caso de que ocurran. Por cada línea del archivo, se separan los datos por los espacios para acceder a ellos fácilmente (data[0] será el primero por ejemplo).
2. Se genera un par key – val (llave - valor) vacío por el momento.
3. Se le aplica el método defval (último parámetro) a data.
4. Si val es nulo no se hace nada, en caso contrario se define un valor para key (el primero dato) y otro para val (el segundo dato sin saltos de línea en caso de existir).
5. Se aplica la función func (segundo parámetro) a val definido recientemente.
6. En caso de que el tercer parámetro sea “change”, en vez de indexar por llave (key), se hace por valor (val). Siempre y cuando proc (resultado de la llamada a func(val)) no sea nulo.
7. Se pregunta si out (inicialmente vacío) contiene la llave “val”, de ser así se le adjunta, en la posición val, el valor de proc = func(val). Si no, se le cambia el valor a out[val] por proc.
8. En caso de que el tercer parámetro del map no sea “change”, (“notChange” por ejemplo) y que proc no sea vacío, se pregunta si out tiene a key como llave, de ser así se le adjunta, en la posición del key, el valor de proc. Si no, se le cambia el valor a out[key] por proc.
9. Finalmente se cierra el archivo y se retorna el diccionario out.

²⁵ Código original en <http://aivwiki.alma.cl/~mauro/mapReduceOK.py>. Para este trabajo de tesis se le hicieron modificaciones.

El código del método *maperGroup* se puede apreciar a continuación:

```
def mapGroup(inFile, func, mode, defval=None):
    out = {}
    try:
        A = open(inFile)
    except:
        print"Error opening %s." % (inFile)
        sys.exit(1)
    for line in A:
        data = re.split(" ", line)
        (key, val)=("","")
        if(defval != None):
            valf = "%s(%s)" % (defval, data)
            (key, val) = eval(valf)
            if(val == None):
                pass
            else:
                (key, val) = (data[0], data[1].replace("\n", ""))
        mapf = "%s(%s)" % (func, val)
        proc = eval(mapf)
        if(mode == "change"):
            if(proc != None):
                if(out.has_key(val)):
                    out[val].append(proc)
                else:
                    out[val] = [proc]
            else:
                if(proc != None):
                    if(out.has_key(key)):
                        out[key].append(proc)
                    else:
                        out[key] = [proc]
    A.close()
```

```
return out
```

Tabla 4: Códigos del método mapGroup.

En el caso de listaIP(), map se llama de la siguiente manera:

```
map = mapGroup("fileLimpio_listaIP.txt",  
"mycount", "notChange", "listaIP").
```

Tabla 5: Ejemplo de llamada a mapGroup.

El primer parámetro corresponde al archivo que el map debe leer (el resultante del método nada), el tercero notChange indica que se quiere indexar los valores por llave (no por valor, en caso de ser "change" ocurre lo contrario). El segundo y tercer parámetro son los métodos a aplicar en los datos. Se detallan a continuación:

La primera función que se aplica es listaIP y luego corre mycount (ambos métodos existentes en Maper.py).

```
def listaIP(data):  
    (key, val) = (data[5]+"."+data[6]+"."+data[7]+"."+data[8], None)  
    return(key, val)
```

Tabla 6: Definición de listaIP.

Este método se encarga de hacer una lista con pares llave y valor (key,val). En este caso, la llave (key) es la dirección IP, se captura de esta manera dado que, por lo realizado anteriormente el archivo "fileLimpio_listaIP.txt" contiene los datos separados por columnas. Entre la posición 5 y la 8 (iniciando desde 0) se encuentran los 4 números que corresponden a la dirección IP de origen.

Recordar que los datos de entrada, gracias al tcpdump y el método "nada", quedan con la forma (IPs de la darknet reemplazadas por 1.1.1.x):

```
00 00 49 643174 IP 218 86 103 28 29005 > 1 1 1 121 62214: UDP, length 30  
00 00 54 301936 IP 190 22 101 23 1870 > 1 1 1 34 25: tcp 0  
00 00 52 967747 IP 58 9 108 89 > 1 1 1 239: ICMP echo request, id 8, seq  
38204, length 8
```

Tabla 7: Ejemplo de tráfico.

El valor (val) es el dato "None" (contar las apariciones de las distintas direcciones IP es sencillo, por lo tanto este campo no se utilizará en esta ocasión).

Finalmente se llama a mycount que únicamente asocia un número 1 a cada dirección IP (según cuantas veces aparezca).

Luego por ejemplo si cierto puerto fue accedido 7 veces en la red, después del mapeo se tendrá:
'186.129.251.169': [1,1,1,1,1,1,1].

Esto se utiliza para que próximamente el método reduce cuente las apariciones de cada dirección IP resultante del mapeo.

```
def mycount(val):  
    return 1
```

Tabla 8: Definición del método mycount.

Para los puertos ocurre algo similar, el map se llama de la siguiente forma en listaPuertos():

```
map = mapGroup("fileLimpio_listaPuertos.txt",  
              "mycount", "notChange", "listaPu")
```

Tabla 9: Ejemplo de llamada a mapGroup.

La única diferencia con listaIP() es el último parámetro, *listaPu*, lo que es razonable porque ahora no se quiere rescatar la IP de los datos, si no que el puerto accedido.

En este caso los datos llegan de la misma forma que en listaIP a excepción de que no existirán accesos (líneas) con protocolo ICMP dado que éstas no contienen el puerto accedido (recordar que este filtro se aplicó en *generadorListas.sh* antes de llamar a listaPuertos.py).

Entonces, teniendo datos de la forma (IPs de la darknet reemplazadas por 1.1.1.x):

```
00 00 49 643174 IP 218 86 103 28 29005 > 1 1 1 121 62214: UDP, length 30  
00 00 54 301936 IP 190 22 101 23 1870 > 1 1 1 34 25: tcp 0
```

Tabla 10: Ejemplo de tráfico.

Se llama al método listaPu que busca el puerto accedido en la darknet, ubicado en la 15 posición.

```
def listaPu(data):  
    (key, val) = (data[15], None)  
    return (key, val) .
```

Tabla 11: Código del método listaPu.

Dado que el maper ya está resuelto, se procede a realizar el reduce. En ambos casos (lista de puertos y lista de IPs) se llama de la misma manera:

```
reduce13 = reduce(map, "sum")
```

Tabla 12: Ejemplo de llamada a reduce.

El método reduce se encuentra ubicado en la clase Reduce, recibe como parámetros el resultado del map y un método que deberá aplicarle a los datos. Su procedimiento se detalla a continuación:

1. Se itera sobre los elementos del diccionario retornado por el map, compuestos por una llave y un valor (key, val).
2. Si el método a llamar corresponde a uno cuyo nombre contiene la palabra "Prot", res (el resultado) será un arreglo con 3 valores en 0. Esto es porque existen unos gráficos que despliegan la cantidad de paquetes que llegan a la red según protocolo (detallados más adelante). Estos accesos son almacenados en el arreglo res, donde la primera posición corresponde a los realizados por el protocolo TCP, el segundo por UDP y el tercero por ICMP (u otros). Si no se está en ese caso, res seguirá siendo un número (0).
3. A cada valor val del diccionario se le aplica el método (fx) dado como parámetro en el reduce. Generalmente vals (resultado del map) corresponde a varios 1's los que se van sumando acumulativamente.
4. Finalmente a cada key se le asocia el resultado del cálculo anterior, generalmente el resultado de una suma de 1's. Finalmente este valor se retorna.

El código de la función reduce es el siguiente:

```
def reduce(inDict, fx):
    out = {}
    for (key, val) in inDict.iteritems():
        res = 0
        if(fx.find("Prot")!=-1):
            res = [0,0,0]
        for aVal in val:
            exe = "%s(%s, '%s')" % (fx, res, aVal)
            res = eval(exe)
        out[key] = res
    return out
```

Tabla 13: Código el método reduce.

El método utilizado por el reduce en este caso es *sum*, el cual, tal como se mencionó anteriormente, simplemente suma los valores que recibe:

```
def sum(a, b):
    return int(a) + int(b)
```

Tabla 14: Código el método sum.

De esta manera por cada dirección IP (o puerto) diferente se suman (acumuladamente) los 1 que fueron creados en el Mapper asociados a cada una de ellas, de esta manera se tiene la cantidad de apariciones de cada IP (o puerto) en la red.

Al finalizar el reduce el ejemplo enseñado anteriormente quedaría así: `'186.129.251.169' : 7.`

Dado que ya se tienen los datos necesarios para crear los archivos con ambas listas, se llama a `printListaIP26` y a `printListaPuertos27` (existentes en la clase `XmlCreator.py`) recibiendo como parámetro el resultado del reduce.

En el caso de las IP, se crea una lista con una dirección IP, un espacio y (tab) luego la cantidad de veces que accedió a la red. Mientras que en el caso de los puertos ocurre lo mismo pero en vez de un espacio entre el puerto y sus accesos hay un pipe “|”.

Una vez que las lista ya se encuentren creadas, se abre un archivo llamado `IP.txt` o `Puertos.txt` y en ellos se escribe el resultado (recordar que en estos archivos las direcciones IP y puertos se encuentran ordenadas por orden descendente de número de accesos.)

Ejemplo:

IP.txt		Puertos.txt
200.6.117.121	47436	25 268526
91.215.77.97	27594	80 78956
202.133.99.45	26510	445 5892

Tabla 15: Ejemplos de IP.txt y Puertos.txt.

Luego `tiraListas.sh` toma estos archivos para continuar su ejecución.

El Diagrama 4 resume el proceso descrito anteriormente:

²⁶ Código disponible en apéndice P.

²⁷ Código disponible en apéndice Q.

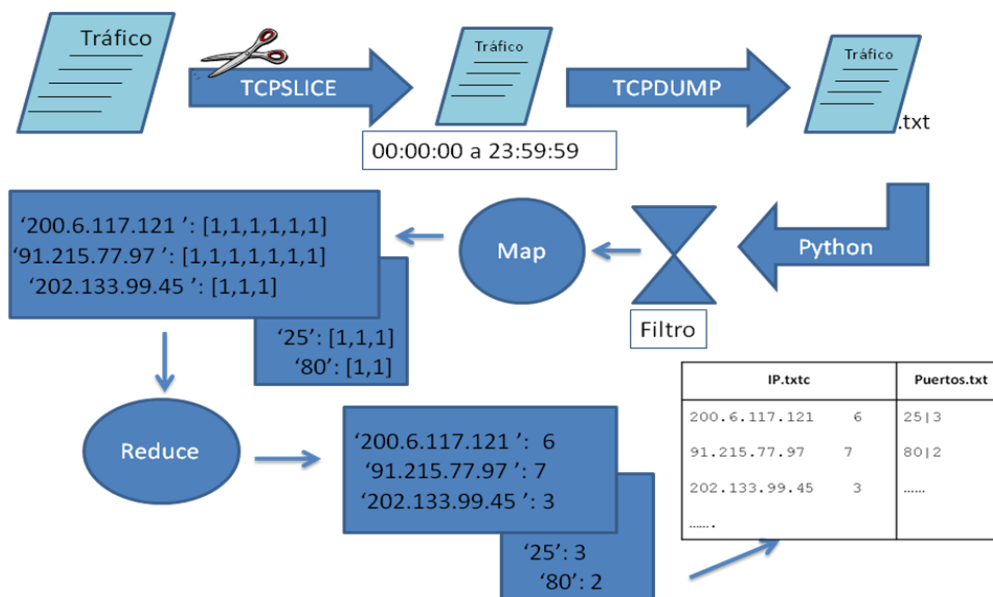


Diagrama 4: Representación del proceso de generación de listas de IPs y puertos. 2012.

Con el fin de crear dos de los gráficos de estado diario (principales 10 puertos accedidos por protocolo) se realizaron dos nuevas listas. Éstas se hacen de manera muy similar a como se obtuvo la lista de Puertos. Ellas contienen los 10 puertos más accedidos a través de protocolo UDP y la cantidad de accesos, lo mismo para el protocolo UDP. Este procedimiento no se detallará dado su parecido al proceso recientemente descrito, a excepción de que se aplica un filtro encargado de determinar si el acceso ocurrió a través del protocolo UDP o si fue a través del protocolo TCP. Por lo tanto en lugar de llamar al filtro “nada” se llama nadaUDP²⁸ y nadaTCP²⁹ sea el caso (copiado de líneas de tráfico que contengan la palabra UDP o TCP). El resto del proceso es prácticamente el mismo que el usado para obtener la lista de puertos globales. Los métodos asociados a ellas son listaPuertosUDP³⁰ y listaPuertosTCP³¹ (análogas a listaPuertos), el resto de los métodos son los mismos que para el caso de la lista de Puertos.

Además, junto con la lista de direcciones IP que generaron más accesos, se realiza una lista que calcula las direcciones IP de la darknet más accesadas, estos valores serán utilizados para los gráficos que despliegan las IPs atacadas. El procesamiento de esta lista es similar a la de la lista de IPs atacantes, con la única diferencia que se capturan las IPs atacadas del archivo de tráfico en lugar de las IPs atacantes. Esta lista no es procesada con el servicio del Team Cymru dado que se sabe que todas ellas pertenecen a

²⁸ Código disponible en apéndice FF.

²⁹ Código disponible en apéndice GG.

³⁰ Código disponible en apéndice HH.

³¹ Código disponible en apéndice II.

la darknet. El método asociado a ella es `listaAtaIP`³² (análogo a `listaIP`) y el utilizado en el map (análogo a `listaIP`) es "`listaAtaIP`"³³, el resto de los métodos son los mismos que para el caso de la lista de IPs atacantes.

4.b.2. Gráficos por omisión:

Esta sección es la más importante y compleja de este trabajo. Corresponde a la creación de gráficos estándar (con parámetros calculados previamente) cada noche para que se encuentren disponibles cada día, para ser revisados en la página web.

Tal como se vio anteriormente, el script `tiraListas.sh` llama al final a `paises.sh` para crear el gráfico de los 10 países que más acceden a la darknet. Su procedimiento es muy sencillo y será explicado a continuación, antes de detallar el complejo procedimiento realizado por `graficosFlash.sh`:

1. Se seleccionan las columnas de interés (país y cantidad de accesos) del archivo `paises${archivo}.txt` (archivo de tráfico del día de ayer) generado durante la ejecución de `tiraListas.sh`.
2. Se suman las cantidades de accesos por país y se seleccionan los 10 primeros resultados.
3. A continuación se comienza a crear el XML con el formato especificado por Maani Charts.
4. Se imprimen los valores calculados recientemente (país y accesos totales) en un formato especial para su posterior interpretación.
5. Se termina de crear el XML con los valores definidos por Maani Charts. Este archivo se almacena en el mismo directorio donde el resto de los gráficos fueron guardados (los creados por `graficosFlash.sh`).

Ahora se explica en detalle el script `graficosFlash.sh`, su complejo procedimiento, utilizado para la creación de prácticamente todos los gráficos, es el siguiente:

1. Primero se calcula la fecha de los 7 días pasados (el día de ayer se utiliza para todos los gráficos, el resto se utiliza para el gráfico de anillos que muestra los protocolos utilizados en los últimos 7 días).
2. A continuación se utilizan las listas que fueron creadas en el script anterior (`tiraListas.sh`) y que ahora se encuentran ubicadas en la carpeta llamada `listas`.

³² Código disponible en apéndice JJ.

³³ Código disponible en apéndice KK.

3. Se rescatan las tres direcciones IP que han enviado mayor cantidad de paquetes a la red, para luego dárselas como parámetro al gráfico de IPs en el tiempo. De esta manera el gráfico desplegará el comportamiento de las direcciones que más accedieron a la red durante el día.
4. Hay que ponerse en el caso en que el comando del Team Cymru haya fallado, y que por lo tanto el archivo almacenado en la variable *fileIP* se encuentre vacío. Es por eso que se cuenta con una lista de respaldo (*fileIP2*) que contiene todas las direcciones IP (archivo resultante del código python) y este es el momento de utilizarla.
5. A continuación se rescatan los cinco puertos más accedidos. Para esto se revisan las 5 primeras líneas del archivo de Puertos. Es necesario encontrar la posición del pipe “|” que se escribió en el archivo de Puertos. Antes de esa posición se encuentra el puerto accedido y después de ella se encuentra la cantidad de accesos. Estos 2 números se almacenan en variables para las 5 primeras líneas (5 puertos más accedidos y sus respectivas cantidades).
6. Ahora se escriben los datos calculados recientemente en un archivo que lleva la fecha de ayer en su nombre (*accesos-salida2012-01-01.txt* por ejemplo), y está ubicado dentro del directorio */home/rfaccio/Desktop/graficosFlash/*. De esta manera existe un registro en texto de los puertos más accedidos el día en que se están haciendo los cálculos.
7. Se rescatan los 10 puertos más accedidos mediante el protocolo UDP y lo mismo para TCP. Estos datos se almacenan en variables para entregárselos como parámetros a los generadores de gráficos de estado.
8. A continuación se realizan las llamadas a los generadores de gráficos, cada uno de ellos será explicado más adelante.

Finalmente comienza la creación de los gráficos flash, en realidad acá se hacen los archivos XMLs para que luego Maani Charts los interprete y los muestre como gráficos flash. Estos gráficos se consideran interesantes y entregan una vista global de lo que ocurrió durante el día. El usuario tendrá la posibilidad de crear gráficos desde la página web con los datos que le parezcan interesantes y ellos serán tomados como parámetros.

Para la creación de los gráficos, se le delega la responsabilidad a distintos scripts sh, ellos son *generador.sh*, *generadorRing.sh* y *generadorTime.sh* (dependiendo del tipo de gráfico que se desea crear). A cada uno de estos scripts hay que entregarle variados parámetros:

a) Hora de inicio y de fin para crear el corte en el archivo de tráfico (binario). En este caso se quiere tener una visión del día completo, por lo tanto se le da 00:00:00 y 23:59:59 hrs.

b) Archivo(s) de tráfico a observar: el archivo seleccionado corresponde al de ayer (recordar que esto se ejecuta en la madrugada de cada día para que los datos del día anterior estén listos y disponibles) o al de los últimos 7 días, fechas que fueron calculadas en el inicio de este script (para el caso del gráfico de anillos de la sección “Accesos por protocolo”).

c) Tipo de gráfico a crear: dado que se tienen solo 3 generadores (scripts) de gráficos y son 15 los gráficos a crear, hay que entregar como parámetro el tipo. El script *generadorRing.sh* no recibe este parámetro porque éste solamente genera el gráfico de anillos, por lo tanto no es necesaria la presencia de un identificador. Las palabras: *pie*, *pieProt*, *rad*, *bubble*, *rect*, *column*, *area*, *time*, *timeIP*, *timeAtaIP*, *columnAta*, *AreaAta*, *rectUDP* y *rectTCP* definen los gráficos existentes definidos anteriormente (aparte del gráfico de anillos)

Las categorías que los incluyen ve pueden ver en la Tabla 16:

Tipo de gráfico	Categoría (ataque asociado)³⁴	Generado por
Pie	Accesos por Puerto (escaneo de puertos)	<i>generador.sh</i>
pieProt	Accesos por protocolo (denegación de servicios)	<i>generador.sh</i>
Rad	Accesos por puerto y protocolo	<i>generador.sh</i>
Bubble	Accesos por puerto y protocolo	<i>generador.sh</i>
Rect	Accesos por puerto y protocolo (escaneo de puertos)	<i>generador.sh</i>
Column	Accesos por protocolo y dirección IP externa (denegación de servicios)	<i>generador.sh</i>
ColumnAta	Accesos por protocolo y dirección IP de la darknet (denegación de servicios y gusanos)	<i>generador.sh</i>
Area	Accesos por protocolo y dirección IP externa	<i>generador.sh</i>
AreaAta	Accesos por protocolo y dirección IP de la darknet	<i>generador.sh</i>
Time	Accesos a puertos en el tiempo (escaneo de	<i>generadorTime.sh</i>

³⁴ Los gráficos sin ataque asociado se considerarán como complementarios.

	puertos)	
timeIP	Accesos de IPs en el tiempo (denegación de servicios)	generadorTime.sh
timeAtaIP	Accesos de IPs de la darknet en el tiempo (denegación de servicios y gusanos)	generadorTime.sh
gráfico de anillos (Ring)	Accesos por protocolo	generadorRing.sh

Tabla 16: Gráficos, sus categorías y script generador.

d) Puertos escogidos: Esto permite crear un filtro para visualizar una cantidad legible de información. Si se solicitan todos los puertos existentes en el día se tienen demasiados datos y esto no permite visualizar los gráficos de buena manera. Es por esto que mediante muchas pruebas se estimó conveniente graficar los puertos más importantes. Para la mayoría de los gráficos los puertos escogidos son los 5 más accedidos por día (calculados recientemente en este script).

Para los gráficos bubble, column y area se determinó que los puertos a visualizar serían 21 23 25 53 80 443 110 por ser los más importantes en una red. Si a estos gráficos se les pide que grafiquen filtrando por los 5 puertos más accedidos (como se hace en los otros gráficos), el resultado no es el óptimo. Esto se debe a que los gráficos mencionados incluyen direcciones IP, y por lógica, las direcciones IP diferentes que acceden a los puertos más visitados son muchos, y esto genera un gráfico con muchísimos datos y difícil lectura (generalmente el puerto más accedido supera las 2 millones de visitas diarias y los siguientes bordean los cientos de miles). Finalmente, después de algunas pruebas, se optó por que el gráfico bubble no se despliegue en el sitio en la sección de gráficos por omisión porque al ser muchos datos la lectura era imposible (pero de todas maneras el XML asociado a este gráfico se calcula y se explicará en detalle tal como se hace con los otros tipos de gráficos). En la sección de creación de gráficos ad-hoc se puede apreciar este tipo de gráfico porque ahí se espera que el usuario desee ver datos más específicos y por lo tanto la lectura se simplifica (detalles en la sección 6).

e) IPs escogidas: existen solamente dos gráficos que realiza un filtro por direcciones IPs, estos son “timeIP” y “timeAtaIP” que despliegan el comportamiento de las IP en el tiempo. A estos gráficos se le da la variable IPs que contiene las 3 direcciones que realizaron más accesos y 3 direcciones IP que fueron más accedidas (calculadas en este script anteriormente).

La cantidad de puertos IPs utilizados por los filtros (3 direcciones IPs y 5 puertos como máximo) fue escogida en pos de la buena visualización. Si se eligen demasiadas direcciones o puertos el gráfico se satura o provoca que su lectura sea difícil. Otro problema que puede ocurrir cuando el gráfico contiene muchos datos es que su interpretación sea complicada para Maani Charts y se alcance el timeout

definido por Flash para su despliegue. La principal idea es tener gráficos simples, cuya lectura sea rápida y permite tomar decisiones asociadas a la seguridad de manera fundamentada. Los filtros escogidos son un tema que se puede mejorar, lo primordial es no dejar información importante de lado al realizarlos. Por ejemplo, puede ocurrir que al filtrar por las direcciones IPs que realizan más accesos a la darknet, se omitan aquellas que, a pesar de realizar pocos accesos, tienen un comportamiento interesante de analizar. Es por esto que la manera de rescatar información debe seguir siendo estudiada a fondo, por lo tanto, como mejora a este trabajo, se podrían proponer nuevos filtros.

Una vez que los gráficos ya han sido creados se organizan los XML resultantes en carpetas separadas por días dentro del directorio /home/rfaccilo/desktop/graficosFlash. Para esto se utiliza una carpeta “intermedia” para evitar sobre escrituras y permitir la creación de una carpeta por día (mkdir) para que luego en ella se encuentren todos los gráficos (archivos XML).

- Además el archivo de texto que comenta sobre los puertos y sus accesos se ubica en la misma carpeta diaria que contiene los archivos XML de determinado día (ayer).

El proceso anterior, creación de XMLs, se resume en el Diagrama 5:

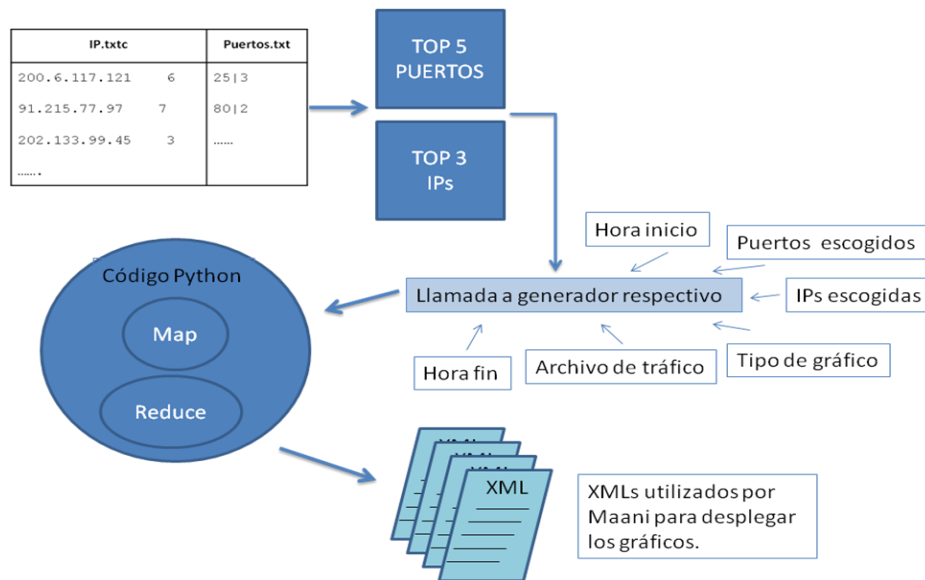


Diagrama 5: Representación del proceso de creación de XMLs. 2012.

Para continuar, se explicarán cada uno de los generadores de gráficos.

El primero será *generador.sh* dado que hace la mayor cantidad de gráficos, su procedimiento es el siguiente:

- Primero se define un directorio base, para esta ocasión será el Desktop del computador.
- Una vez dentro de la carpeta que contiene las capturas, se llama al comando *tcpslice* para que genere un nuevo archivo con el tráfico deseado, esto es, el tráfico comprendido entre los

horarios especificados (en este caso es el día completo nuevamente, de 00:00:00 a 23:59:59). Se escribe además un archivo log en caso de errores.

3. A continuación se llama al comando `tcpdump` para que lea el archivo generado por `tcpslice` y lo lleve a formato de texto. Cabe destacar que se usa el parámetro “-t” para que no incluya la hora como dato de salida (distinto al uso anterior en el generador de Listas). Se le hacen algunas transformaciones al texto para que el archivo resultante contenga los datos separados por espacios (con forma de columnas).
4. El archivo llamado `cortado_00h00m00s_23h59m59s_salida-2012-01-01.txt` por ejemplo, tiene la siguiente forma (IPs de la darknet reemplazadas por 1.1.1.x):

```
IP 178 125 248 41 10010 > 1 1 1 1 16301  UDP, length 33
IP 24 232 0 245 41611 > 1 1 1 1 25  tcp 0
IP 212 90 170 214 > 1 1 1 1  ICMP echo request, id 3, seq 57089,
length 8
```

Tabla 17: Ejemplo de tráfico.

5. Se almacenan los parámetros de entrada en variables para su posterior uso. Se hace un shift de los primeros 4 parámetros recibidos por el script (justamente los 4 guardados en las variables), de esta manera al usar “@” se tendrá acceso a todas las variables restantes, en este caso fueron los puertos más accedidos (\$p2 \$p3 \$p4 \$p5 del script anterior).
6. Ahora dependiendo del tipo de gráfico que se desee crear (cuarto parámetro llamado \$tipo) se ejecutarán distintos códigos python. En cada caso se escribe un archivo de log para revisar en caso de errores.

En caso de que se quieran monitorear todos los puertos, y no solo algunos en específico, se puede llamar a este script con la palabra “all” como cuarto parámetro, de esta manera el filtro (existente en el código python) dejará pasar a todos los puertos accedidos. En caso de querer monitorear todos los puertos conocidos (menores a 1024) se puede entregar la palabra “known” como cuarto parámetro. En este caso, las palabras claves no han sido utilizadas, pero pueden ser de gran ayuda para el analista cuando desee crear sus propios gráficos desde la página web.

7. Una vez finalizadas las posibles alternativas de gráficos para este script se eliminan los archivos intermedios que fueron creados.

Antes de explicar los código en python se detallarán los otros dos generadores de gráficos restantes: *generadorTime.sh*³⁵ y *generadorRing.sh*³⁶.

³⁵ Código disponible en apéndice S.

Continuando con *generadorTime.sh*, similar al anterior, su procedimiento es el siguiente:

1. Tal como en el script anterior se crea un directorio base por comodidad.
2. Una vez dentro de la carpeta que contiene las capturas se llama a *tcpslice* de la misma manera que en el script anterior, para rescatar el trozo de tráfico que se desea monitorear (en este caso abarca todo el día).
3. A continuación se llama a *tcpdump* pero en este caso no se ejecuta el comando “-t” porque ahora sí se quiere contar con la hora como dato. Luego se le aplican algunas transformaciones al texto para contar solamente con columnas (datos separados por espacios). Nótese que hubo que reemplazar con “sed” el carácter “:” por un espacio más veces que en el script anterior, esto es porque al tener la hora presente hay más de éstos caracteres por línea.
4. El archivo resultante de estos comandos tiene la forma (IPs de la darknet reemplazadas por 1.1.1.x):

```
00 00 49 643174 IP 218 86 103 28 29005 > 1 1 1 121 62214: UDP, length
30
00 00 54 301936 IP 190 22 101 23 1870 > 1 1 1 34 25: tcp 0
00 00 52 967747 IP 58 9 108 89 > 1 1 1 239: ICMP echo request, id 8,
seq 38204, length 8
```

Tabla 18: Ejemplo de tráfico.

5. Nuevamente se almacenan los parámetros en distintas variables y se hace un shift sobre ellas. De esta manera con “@” se tendrá acceso a las direcciones IPs solicitadas (en el caso del gráfico *timeIP*) o a los puertos requeridos (para el gráfico *time*).
6. Este script crea 3 tipos de gráficos distintos, uno de acceso de IPs atacantes en el tiempo (*timeIP*), otro de las IPs atacadas (*timeAtaIP*) y otro de puertos accedidos en el tiempo (*time*).
7. Para el caso de “*timeIP*” y “*timeAtaIP*” se ejecutan códigos python con las direcciones IP como parámetro (para la generación de gráficos por omisión se seleccionaron las 3 IPs que más accedían a la red y las 3 más atacadas). Mientras que en el caso de “*time*” se le da como parámetro los puertos más accedidos (a excepción del primero, porque al ser demasiados accesos, estropea la escala del gráfico final).
8. Finalmente se eliminan archivos intermedios que hayan sido creados.

³⁶ Código disponible en apéndice T.

A continuación se explicará el generador del gráfico “ring” llamado *generadorRing.sh*. Este script es parecido a *generador.sh*, la única diferencia relevante es que utiliza como entrada 7 archivos de tráfico (dado que muestra el comportamiento en la red durante la última semana). Es por esto que este gráfico tarda más tiempo en estar disponible pero el funcionamiento es similar, se detalla a continuación

1. Nuevamente se crea la variable con el directorio base y se entra a la carpeta de capturas.
2. Se llama a *tcpslice* y *tcpdump* (este último con el parámetro “-t” para que no se almacene el tiempo como dato) para cada uno de los archivos de tráfico dados como parámetro (\$3, \$4, \$5, \$6, \$7, \$8 y \$9) siendo las horas de corte los parámetros \$1 (inicio: 00:00:00) y \$2 (fin: 23:59:59) para todos los días. En este paso también se le realizan los cambios necesarios al texto para que queden los datos con forma de columnas (separadas por espacio).
3. En este punto se tienen siete archivos intermedios representando la semana recién pasada (7 días antes de la fecha de ejecución de *graficosFlash.sh*).
4. Se almacenan los parámetros en variables nuevamente y se realiza el shift.
5. Se llama a un código python con los 7 archivos de tráfico en texto generados anteriormente, además se mantiene un archivo para registro de errores (log).
6. Finalmente se eliminan los archivos auxiliares creados durante el proceso.

El Diagrama 6 resume el comportamiento de los tres generadores explicados anteriormente:

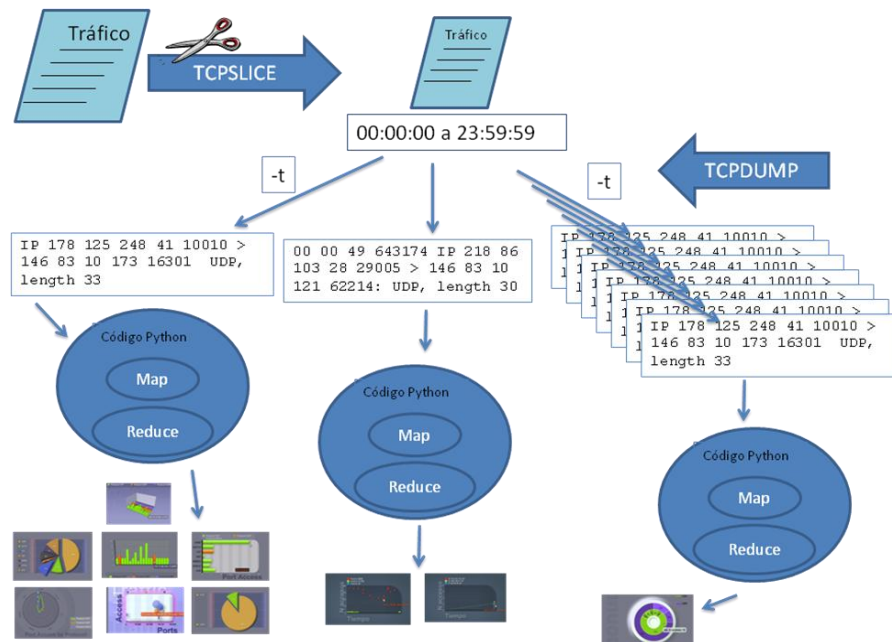


Diagrama 6 : Representación del proceso de los generadores de gráficos. 2012.

Cada uno de los códigos python utilizados para la creación de los archivos XMLs tiene la misma forma. Un ejemplo de esto sería la llamada a *“pie.py”* encargada de crear el gráfico “Accesos por puerto”. Su código³⁷ se puede visualizar a continuación en la Tabla 19.

³⁷ Código del resto de las llamadas python disponible en apéndice U.

Llamada	Código
pie.py	<pre> from Main import * limpia(sys.argv[1], "fileLimpio.txt"³⁸) pie() </pre>

Tabla 19: Código de para la generación del gráfico “Accesos por puerto”.

Como se puede apreciar, primero se llama a un filtro (llamado “limpia”) y luego a un método principal, “pie()” en este caso, quien hará el map, el reduce y como resultado final entregará el XML.

Los filtros existentes y su procedimiento se detallan a continuación:

a) limpia(archivo_entrada, archivo_salida)³⁹:

1. Primero se determinan cuales serán los parámetros a evaluar (puertos), en este caso los puertos seleccionados se encuentran ubicados desde la segunda posición del sys.argv en adelante.
2. Se intenta abrir el archivo a leer y el archivo donde se escribirán los datos filtrados.
3. Por cada línea del archivo de entrada se hace una separación de los datos por cada espacio, de esta manera se dispone de los datos en forma de arreglo pudiendo acceder a ellos de la forma data[1], data[2], data[3] etc.
4. Se revisa el protocolo, ubicado en la décimo tercera posición, ya sea TCP o UDP, las líneas con protocolo ICMP se excluyen dado que ellas no tienen la información del puerto accedido y en este caso se está filtrando por ellos. Además existe una verificación respecto al puerto, ubicado en la posición 11 de la línea, éste debe ser un número.
5. En la siguiente muestra se puede apreciar que en la posición 13 se encuentran las palabras UDP, o tcp, y que en la onceava posición está el puerto (entre el puerto y el protocolo hay una columna con un espacio vacío que corresponde a la posición 12, IPs de la darknet reemplazadas por 1.1.1.x):

³⁸ Quizás un mejor nombre para este archivo podría ser datosFiltrados.txt porque contiene los datos pasados por el filtro determinado.

³⁹ Código disponible en apéndice V.

<p>Protocolo TCP:</p> <pre>IP 178 125 248 41 10010 > 1 1 1 173 16301 UDP, length 33</pre> <p>Protocolo UDP:</p> <pre>IP 24 232 0 245 41611 > 1 1 1 11 25 tcp 0</pre>
--

Tabla 20: Ejemplo de tráfico.

En el caso del protocolo ICMP lo anteriormente mencionado no se cumple (IPs de la darknet reemplazadas por 1.1.1.x):

<p>Protocolo ICMP:</p> <pre>IP 212 90 170 214 > 1 1 1 173 ICMP echo request, id 3, seq 57089, length 8</pre>

Tabla 21: Ejemplo de tráfico.

6. Ahora, para cada uno de los parámetros rescatados en un inicio del método (puertos) se evalúa si el usuario está pidiendo los puertos conocidos (palabra clave: known), de ser el caso se revisa que el puerto de la línea actual sea menor que 1024, de ser así se copia toda la línea en el archivo de salida.
7. Luego se revisa si se está preguntando por todos los puertos (palabra clave: all), de ser así se copia la línea actual en el archivo de salida sin más verificaciones (si el usuario ingresa all, se copiarán todas las líneas del archivo original siempre y cuando tengan protocolo TCP o UDP)
8. Se entra al caso más común en donde el usuario especifica algunos puertos de interés, de ser así se revisa que en la línea exista el puerto escogido (gracias al *for* se revisan todos los puertos escogidos), si existe una coincidencia la línea es copiada.
9. Finalmente el manejo de errores en caso de que alguno de los archivos no se haya podido abrir. Se termina cerrando ambos archivos.

Existen además los filtros `limpiaUDP`⁴⁰ y `limpiaTCP`⁴¹ utilizados para los gráficos de estado. Éstos tienen a tarea de únicamente copiar líneas de tráfico que contengan el protocolo UDP o TCP según corresponda.

⁴⁰ Código disponible en apéndice W.

⁴¹ Código disponible en apéndice X.

b) limpiatime(archivo_entrada, archivo_salida)⁴²:

1. La primera parte de este método es igual a la del método anterior, es dentro del ciclo for donde comienzan las diferencias.
2. En este caso es necesario revisar si la quinceava posición es un dígito, (en el método anterior se revisaba la posición número 11), dado que el archivo de entrada tiene un formato diferente (recordar que para el gráfico que utiliza este filtro (time), el tcpdump no corría con el parámetro “-t” y por lo tanto la hora sí es considerada, por ello es que la posición del puerto es más adelante que en el método anterior donde la hora no era rescatada.)
3. Luego se hace la misma comparación hecha anteriormente para determinar si en la línea actual se encuentra el puerto solicitado.
4. Como se pudo apreciar, acá no se está verificando si en cierta posición se tiene el indicador de protocolo TCP o UDP. De todas maneras no se están dejando pasar líneas con protocolo ICMP, esto se hace al verificar que la posición 15 es un dígito, cuando se está presente a una línea con protocolo ICMP en la décimo quinta posición está la palabra “echo” en vez de un puerto por lo tanto el chequeo ocurre en ese momento. En la siguiente muestra se puede apreciar la quinceava posición destacada en cada caso (IPs de la darknet reemplazadas por 1.1.1.x):

```
Protocolo TCP:
00 00 49 643174 IP 218 86 103 28 29005 > 1 1 1 121 62214: UDP,
length 30

Protocolo UDP:
00 00 54 301936 IP 190 22 101 23 1870 > 1 1 1 34 25: tcp 0

Protocolo ICMP:
00 00 52 967747 IP 58 9 108 89 > 1 1 1 239: ICMP echo request,
id 8, seq 38204, length 8
```

Tabla 22: Ejemplo de tráfico.

5. Finalmente se manejan excepciones y se cierran ambos archivos.

⁴² Código disponible en apéndice Y.

c) `limpiaIP(archivo_entrada, archivo_salida)`⁴³:

1. Este método también es similar a los anteriores pero con la diferencia de que acá se buscan direcciones IP y no puertos (usado por el gráfico de IPs en el tiempo).
2. Los cuatro números que definen la dirección IP se encuentran en las posiciones 5, 6, 7 y 8 de cada línea. Por lo tanto se debe revisar que sean dígitos. Para todos los protocolos la dirección IP de origen se encuentra en la misma posición, por lo tanto en este caso, los accesos a través del protocolo ICMP no se están dejando de lado. A continuación un ejemplo (IPs de la darknet reemplazadas por 1.1.1.x):

```
Protocolo TCP:
00 00 49 643174 IP 218 86 103 28 29005 > 1 1 1 121 62214: UDP,
length 30
Protocolo UDP:
00 00 54 301936 IP 190 22 101 23 1870 > 1 1 1 34 25: tcp 0
Protocolo ICMP:
00 00 52 967747 IP 58 9 108 89 > 1 1 1 239: ICMP echo request,
id 8, seq 38204, length 8
```

Tabla 23: Ejemplo de tráfico.

3. Finalmente, la revisión de errores y el cierre de los archivos.

También existe *limpiaAtaIP*⁴⁴ utilizado para los gráficos que despliegan las IPs atacadas de la darknet. Su procesamiento es similar al recientemente descrito a excepción de que se procesan las IP de destino (diferentes posiciones en data, 11, 12, 13 y 14).

Los métodos llamados después del filtro son similares, todos situados en la clase `Main.py` excepto `ringProto()` que se encuentra en la clase `MainRing.py`. Cada uno de éstos llama a `map` y a `reduce` (con sus métodos respectivos) y al creador del XML final. La diferencia entre los métodos radica en las funciones que reciben como argumento para el `map` y el `reduce` ejecutado dentro de ellas.

Tal como se vio anteriormente, el método “`map`” recibe dos funciones, una que tiene como parámetro los paquetes dirigidos la red (`data`) y otra el valor (`val`, el resultado de la aplicación de una función (`func`) a `data`).

⁴³ Código disponible en apéndice Z.

⁴⁴ Código disponible en apéndice AA.

Los métodos llamados después de los filtros se listan a continuación, junto a cada uno de ellos se despliegan los métodos defval y func invocados por map:

Método	defval(data)	func(val)
column()	stacked3dColumn(data): Retorna el par llave, valor: la dirección IP de origen acompañada del largo de la línea.	protTrans(val): Teniendo el largo de la línea se puede conocer el protocolo al que corresponde y retornarlo.
columnAta()	stacked3dColumnAta(data): Retorna el par llave, valor: la dirección IP de destino acompañada del largo de la línea.	protTrans(val): Teniendo el largo de la línea se puede conocer el protocolo al que corresponde y retornarlo.
pie()	pie(data): Se retorna la dirección IP acompañada del puerto accedido. En caso de que el protocolo sea ICMP el puerto será "None".	mycount(val): Simplemente retorna un 1 por aparición para después sumar.
pieProt()	pieProt(data): Se retorna la dirección IP acompañada del protocolo correspondiente, éste se calcula a partir del largo de cada línea.	mycount(val): Actúa igual que en el caso de pie().
Rad()	rad(data): Se retorna el puerto acompañado del largo de la línea.	protTransRad(val): Teniendo el largo de la línea se puede conocer el protocolo al que corresponde y retornarlo.
Area()	stacked3dColumn(data): Actúa igual que en el caso de column().	protTrans(val): Actúa igual que en el caso de column().
AreaAta()	stacked3dColumnAta(data): Actúa igual que en el caso de columnAta().	protTrans(val): Actúa igual que en el caso de columnAta().
Rect()	rad(data): Actúa igual que en el caso de Rad().	protTransRad(val): Actúa igual que en el caso de Rad().

<code>bubble()</code>	<code>double(data)</code> : Se retorna el par key, val, pero en este caso el key contiene el protocolo (dependiendo del largo de la línea), un asterisco (*), la dirección IP, un guión (-) y finalmente el puerto accesado. Como "value" se retorna el número 0 simplemente. Esto es porque se quieren contar luego las apariciones iguales (en cuando a protocolo, puerto e IP) por lo tanto el campo "val" pierde su relevancia.	<code>mycount(val)</code> : Actúa igual que en el caso de <code>pie()</code> .
<code>time()</code>	<code>time(data)</code> : Se retorna en key la hora del acceso, y el puerto accesado (separados por un asterisco). Nuevamente el "value" se omite, retornándolo con valor "None".	<code>mycount(val)</code> : Actúa igual que en el caso de <code>pie()</code> .
<code>timeIP()</code>	<code>timeIP(data)</code> : Se retorna en key la hora del acceso, y la dirección IP de origen (separados por un asterisco). Nuevamente el value se omite, retornándolo con valor None.	<code>mycount(val)</code> : Actúa igual que en el caso de <code>pie()</code> .
<code>timeAtaIP()</code>	<code>timeAtaIP(data)</code> : Se retorna en key la hora del acceso, y la dirección IP de destino (separados por un asterisco). Nuevamente el value se omite, retornándolo con valor None.	<code>mycount(val)</code> : Actúa igual que en el caso de <code>pie()</code> .
<code>ringProto()</code>	<code>pieProt(data)</code> :): Actúa igual que en el caso de <code>pieProt()</code> .	<code>mycount(val)</code> : Actúa igual que en el caso de <code>pie()</code> .
<code>rectUDP()</code>	<code>rad(data)</code> : Actúa igual que en el caso de <code>Rad()</code> .	<code>protTransRad(val)</code> : Actúa igual que en el caso de <code>Rad()</code> .
<code>rectTCP()</code>	<code>rad(data)</code> : Actúa igual que en el caso de <code>Rad()</code> .	<code>protTransRad(val)</code> : Actúa igual que en el caso de <code>Rad()</code> .

Tabla 24: Funciones defval y func por método creador de gráfico.

Por otro lado, el reduce recibe otro método como parámetro. Para las funciones desplegadas anteriormente existen 3 métodos posibles a utilizar en la reducción, éstos son `groupByProt`, `groupByProtRad` y `sum`. Éstos se explicarán a continuación:

Método	fx (función reduce)
<code>column()</code>	<code>groupByProt(key, val)</code> : Existe un arreglo donde en su primera posición se guardan (acumuladamente) los accesos por TCP, en la segunda los accesos por UDP y en la tercera por ICMP (que siempre estará vacío por los filtros existentes).
<code>pie()</code>	<code>sum(key, val)</code> : Suma de los 2 parámetros recibidos.
<code>pieProt()</code>	<code>sum(key, val)</code> : Actúa igual que en el caso de <code>pie()</code> .
<code>Rad()</code>	<code>groupByProtRad(key, val)</code> : Actúa de la misma forma que <code>groupByProt</code> pero no maneja el caso del protocolo ICMP, si no que cualquiera distinto de TCP o UDP se considera como "Other" (esto se debe a que en <code>protTransRad</code> del map retorna "Other" en ese caso).
<code>Area()</code>	<code>groupByProt(key, val)</code> : Actúa igual que en el caso de <code>column()</code> .
<code>Rect()</code>	<code>groupByProtRad(key, val)</code> : Actúa igual que en el caso de <code>Rad()</code> .
<code>bubble()</code>	<code>sum(key, val)</code> : Actúa igual que en el caso de <code>pie()</code> .
<code>time()</code>	<code>sum(key, val)</code> : Actúa igual que en el caso de <code>pie()</code> .
<code>timeIP()</code>	<code>sum(key, val)</code> : Actúa igual que en el caso de <code>pie()</code> .
<code>ringProto()</code>	<code>sum(key, val)</code> : Actúa igual que en el caso de <code>pie()</code> .

Tabla 25: Función fx por método creador de gráfico.

A modo de ejemplo, se muestra a continuación una tabla con los resultados del map y del reduce para cada gráfico existente:

Método	Resultado Map	Resultado Reduce
column()	{'216.105.40.52': ['TCP', 'TCP', 'UDP', 'UDP', 'UDP'], '208.84.147.187': ['TCP', 'TCP', 'UDP']}	{'216.105.40.52': [2, 3, 0], '208.84.147.187': [2, 1, 0]}
columnAta()	{'1.1.1.39': ['TCP', 'TCP', 'UDP', 'UDP', 'UDP'], '1.1.1.11': ['TCP', 'TCP', 'UDP']}	{'1.1.1.39': [2, 3, 0], '1.1.1.11': [2, 1, 0]}
pie()	{'443': [1, 1], '3389': [1, 1, 1]}	{'443': 2, '3389': 3}
pieProt()	{'TCP': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], "UDP": [1, 1, 1, 1]}	{'TCP': 15, "UDP": 4}
Rad()	{'443': ['TCP', 'UDP'], '3389': ['TCP', 'TCP', 'UDP']}	{'443': [1, 1, 0], '3389': [2, 1, 0]}
Area()	{'216.105.40.52': ['TCP', 'TCP', 'UDP', 'UDP', 'UDP'], '208.84.147.187': ['TCP', 'TCP', 'UDP']}	{'216.105.40.52': [2, 3, 0], '208.84.147.187': [2, 1, 0]}
AreaAta()	{'1.1.1.39': ['TCP', 'TCP', 'UDP', 'UDP', 'UDP'], '1.1.1.11': ['TCP', 'TCP', 'UDP']}	{'1.1.1.39': [2, 3, 0], '1.1.1.11': [2, 1, 0]}
Rect()	{'443': ['TCP', 'UDP'], '3389': ['TCP', 'TCP', 'UDP']}	{'443': [1, 1, 0], '3389': [2, 1, 0]}
bubble()	{'TCP*208.84.147.187-443': [1, 1, 1], 'TCP*92.240.68.152-80': [1, 1, 1, 1, 1, 1, 1,	{'TCP*208.84.147.187-443': 3, 'TCP*92.240.68.152-80': 7, 'UDP*200.89.70.3-53':

	<pre>1], 'UDP*200.89.70.3-53': [1, 1, 1,}]</pre>	<pre>3}</pre>
<code>time()</code>	<pre>{'00:00*53': [1, 1, 1, 1, 1, 1], '00:04*443': [1, 1, 1, 1,], '00:01*53': [1, 1, 1, 1, 1, 1, 1, 1], '00:03*3389': [1, 1, 1, 1, 1, 1], '00:02*53': [1, 1]}</pre>	<pre>{'00:00*53': 6, '00:04*443': 4, '00:01*53': 8, '00:03*3389': 6, '00:02*53': 2}</pre>
<code>timeIP()</code>	<pre>{'00:03*201.186.43.46': [1, 1, 1, 1, 1, 1, 1], '00:02*200.89.70.3': [1, 1, 1, 1, 1], '00:02*65.55.90.85': [1]}</pre>	<pre>{'00:03*201.186.43.46': 7, '00:02*200.89.70.3': 5, '00:02*65.55.90.85': 1}</pre>
<code>timeAtaIP()</code>	<pre>{'00:03*1.1.1.39': [1, 1, 1, 1, 1, 1, 1], '00:02*1.1.1.11': [1, 1, 1, 1, 1]}</pre>	<pre>{'00:03*1.1.1.39': 7, '00:02*1.1.1.11': 5}</pre>
<code>ringProto()</code>	<pre>{'TCP': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], 'UDP': [1, 1, 1, 1]} {'TCP': [1, 1, 1, 1, 1], 'UDP': [1, 1, 1]} {'TCP': [1, 1, 1], 'UDP': [1, 1, 1, 1, 1, 1, 1, 1]} ... (para los 7 días de la semana)</pre>	<pre>{'TCP': 15, 'UDP': 4} {'TCP': 5, 'UDP': 3} {'TCP': 15, 'UDP': 8} ... (para los 7 días de la semana)</pre>
<code>RectUDP()</code>	<pre>{'80': ['UDP', 'UDP'], '25': ['UDP', 'UDP', 'UDP']}</pre>	<pre>{'80': [1, 1, 0], '25': [2, 1, 0]}</pre>
<code>RectTCP()</code>	<pre>{'443': ['TCP', 'TCP'], '3389': ['TCP', 'TCP', 'TCP']}</pre>	<pre>{'443': [1, 1, 0], '3389': [2, 1, 0]}</pre>

Tabla 26: Ejemplos de resultados de map y reduce.

Una vez que ya se tienen los datos parseados se procede a crear los archivos xml. Para esto, después de hacer map/reduce se llama a un método (que pertenece a la clase XmlCreator.py) el cual crea el XML y luego lo escribe en un archivo. La forma que estos XML deben tener está determinada por Maani Charts [MA11], cada tipo de gráfico tiene su formato específico y éste fue respetado para la implementación (los métodos creadores de XML no se presentan en este documento debido a su extensión). Un ejemplo⁴⁵ de dichos XMLs se muestra en la Tabla 27, para el caso del gráfico de “Accesos por Puerto”:

Gráfico	XML representativo
pie()	<pre> ... <row> <null/> <string>443</string> <string>63938</string> <string>53</string> <string>3389</string> </row> <row> <tring></string> <number bevel='data'>114</number> <number bevel='data'>119</number> <number bevel='data'>203</number> <number bevel='data'>272</number> </row> ... </pre> <p>El bloque (row) superior corresponde a los puertos y el inferior a los accesos por puerto (en orden).</p>

Tabla 27: Ejemplo de XML resultante para la creación del gráfico “Accesos por Puerto”.

Los ejemplos de XMLs corresponden a muestras de ellos (solo se mostró lo más interesante, los datos), además existen etiquetas que definen otras variables del gráfico (tales como el estilo, los colores, las sombras etc.).

Finalmente, el servidor web rescata cada día todos los XMLs creados para el despliegue de los gráficos en el sitio.

⁴⁵ Ejemplos del resto de los XMLs se pueden visualizar en apéndice BB.

4. b.3. Alertas

La página web cuenta con un sistema de alertas. El usuario debe llenar los siguientes campos:

Configuración de notificaciones

Notificar cuando existan más de paquetes en un día con el protocolo TCP.
Notificar cuando existan más de paquetes en un día con el protocolo UDP.
Notificar cuando existan más de paquetes en un día con el protocolo ICMP.
Etiqueta para ataque asociado a Protocolos:

Notificar cuando existan más de accesos en un día al (a los) puerto(s) .
Etiqueta para ataque asociado a puertos: .

Figura 23: Formulario de alertas. Imagen Original Sitio web. 2011.

Este formulario requiere que el usuario ingrese contraseña, esto con el fin de que solamente un analista (el administrador) haga cambios en los valores ingresados. Una vez que el botón “Configurar” es presionado, los valores existentes en los campos respectivos se copian a un archivo de texto (ubicado en el servidor web). Cada día a las 00:00 este archivo de texto, llamado datos.txt, es enviado al computador del laboratorio para utilizarlo como datos de entrada al momento de calcular las alertas.

A modo de ejemplo, considerando los parámetros ingresados en los campos de la imagen superior (Figura 23), datos.txt contendría los valores: Ataque Gusano 10000 10000 10000 10000 25 80 4404 445. Nótese la importancia de las etiquetas a ingresar, de esta manera, si el analista conoce la forma en que un ataque se realiza, puede ponerle nombre e identificarlo fácilmente si éste se gatilla.

Cuando el archivo con los datos es recibido, corre el script llamado *tiraAlerta.sh*⁴⁶. Éste calcula la fecha del día anterior (y por lo tanto el nombre del archivo que contiene el tráfico del día respectivo) y llama a dos nuevos scripts, *generadorAlertas.sh*⁴⁷ y *buscador.sh*⁴⁸.

El script *generadorAlertas.sh* funciona de manera similar a la de los generadores de gráficos vistos anteriormente, su procedimiento, en resumen, es el siguiente:

1. Utiliza a *tcpslice* para generar un corte del tráfico del archivo original (en este caso se toma el día completo ya que este script fue llamado con los parámetros 00h00m00s y 23h59m59s).
2. Luego llama a *tcpdump* y algunos otros comandos para generar el archivo de texto que contiene el tráfico en columnas (*tcpdump* corre sin *-t* por lo tanto el tiempo es considerado).

⁴⁶ Código disponible en apéndice CC.

⁴⁷ Código disponible en apéndice DD.

⁴⁸ Código disponible en apéndice EE.

3. Se guardan los parámetros de entrada en variables y se hace el shift ya visto en otras ocasiones.
4. Finalmente se llama a un código en python (alertaProtocolo.py o alertaPuerto.py) recibiendo como parámetro "all" ("todos" en inglés) en la posición de los puertos, esto para que todos los puertos sean considerados al momento de hacer el filtro (ninguno será excluido).

Los códigos de los métodos python alertaProtocolo.py y alertaPuerto.py son importantes de ver en detalle:

alertaProtocolo.py	alertaPuerto.py
<pre>from Main import * nada(sys.argv[1], "fileLimpio.txt"⁴⁹) alertaProtocolo()</pre>	<pre>from Main import * nada(sys.argv[1], "fileLimpio.txt") alertaPuerto()</pre>

Tabla 28: Códigos de alertaProtocolo.py y alertaPuerto.py.

El método llamado es "nada" por lo tanto, en este caso, no se dejan de lado ningún puerto ni tampoco paquetes enviados a la darknet a través del protocolo ICMP. Se puede notar que el cuarto parámetro ("all") dado a *generadoAlertas.sh* no fue usado, pero se deja en caso de que se quiera utilizar algún otro filtro existente, como por ejemplo "limpia", explicado anteriormente, el cual no discriminaría a ningún puerto (por recibir "all" como parámetro) pero no consideraría los accesos hechos a través del protocolo ICMP.

Los principales métodos son alertaProtocolo() y alertaPuerto(), llamados después de la función "nada", ubicados en Main.py al igual que la mayoría de los métodos que hacen map/reduce. Los métodos entregados al map como parámetros son, en cada caso:

Método	defval(data)	func(val)
alertaProtocolo()	alertProt(data)	mycount(val)
alertaPuerto()	listaPu(data)	mycount(val)

Tabla 29: Métodos defval y func para cada tipo de alerta.

En este caso, se sigue la misma estructura utilizada para la creación de los gráficos. Lo novedoso es el método defval entregado como parámetro al map de alertaProtocolo: *alertProt*. El resto de los métodos fueron utilizados y explicados con anterioridad.

La función alertProt() tiene la misma forma que el método pieProt(data) utilizado en la creación de gráficos, la única diferencia es que en este caso el tiempo sí es considerado (el comando tcpdump no

⁴⁹ Quizás un mejor nombre para este archivo podría ser datosFiltrados.txt porque contiene los datos pasados por el filtro determinado.

tiene el “-t” como parámetro como es en el caso de *generador.sh*) y por lo tanto para rescatar la dirección IP es necesario acceder a campos distintos de *data[]*.

Finalmente, luego de llamar a métodos para imprimir los datos de la manera correspondiente, se obtienen archivos de texto de la siguiente forma:

alertaProtocolo.txt	alertaPuerto.txt
""ICMP"" 63529	41019' 55
""ITCP"" 433211	'35231' 112
""UDP"" 243120	'65527' 477
	'61101' 162
	'44288' 575

Tabla 30: Ejemplos de alertaProtocolo.txt y alertaPuerto.txt.

Ahora que los datos se encuentran ordenados se procede a buscar en ellos ocasiones en que se hayan alcanzado o superado las alertas definidas en la página web, valores que se encuentran en el archivo *datos.txt*. Para esto se procede a llamar al script *buscador.sh*, cuyos pasos son los siguientes:

1. Primero se calcula la fecha el día anterior.
2. Se separan en variables cada uno de los números que se encuentran en *datos.txt* de manera de acceder a ellos fácilmente más adelante.
3. Primero se evalúa el caso de los protocolos, utilizando la lista creada por el map/reduce, *alertaProtocolo.txt*. Se pregunta si el número de accesos realizados con el protocolo TCP superan la cantidad máxima definida por el usuario. De ser así se escribe una línea (en el archivo llamado *resumen.txt*) indicando que el protocolo TCP fue usado “x” veces (siendo “x” la cantidad de accesos registrados en *alertaProtocolo.txt*) y se concatena la etiqueta ingresada (nombre del ataque). A continuación se hace lo mismo para los demás protocolos.
4. Es así como en el archivo *resumen.txt* queda un registro de los accesos por protocolo que han superado el número definido por el usuario.
5. Para continuar se sigue el mismo procedimiento con cada uno de los puertos. En caso de superar una alerta se escribe dicho suceso en el mismo archivo mencionado anteriormente, *resumen.txt*.
6. Finalmente el servidor web rescata el archivo *resumen.txt* para desplegarlo en el sitio como sigue:

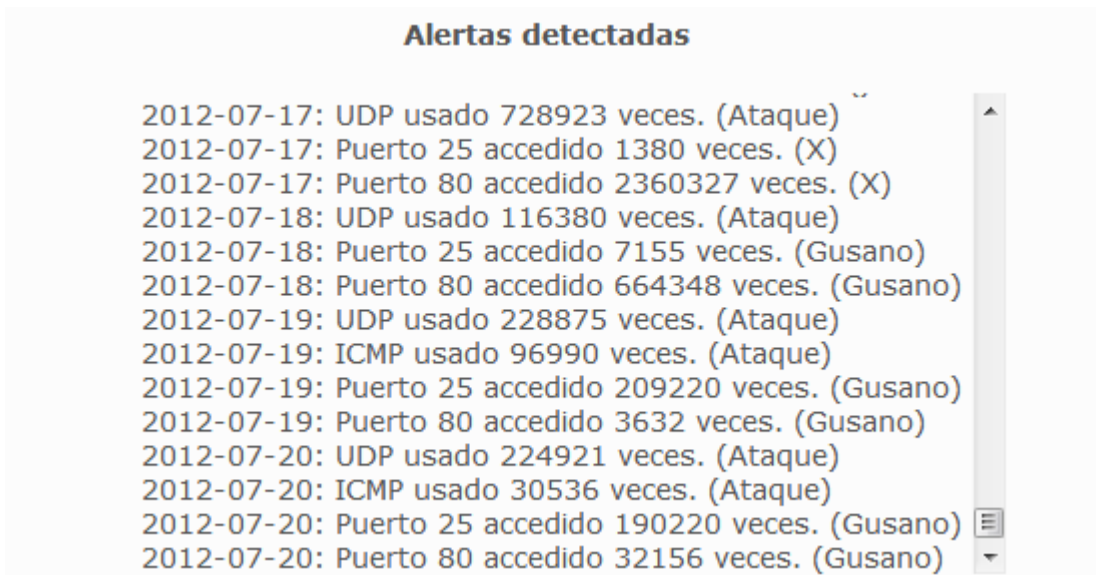


Figura 24: Ejemplo de lista de alertas. Imagen Original Sitio web. 2011.

Nótese que el archivo datos.txt que guarda la configuración establecida por el usuario para las alertas viaja todos los días desde el servidor web hacia el computador del laboratorio a medianoche, por lo tanto si la configuración es cambiada por el usuario durante un día en la tarde, el buscador de alertas actuará al día siguiente utilizando como entrada el nuevo archivo datos.txt.

El Diagrama 7 explica el funcionamiento del sistema de alertas:

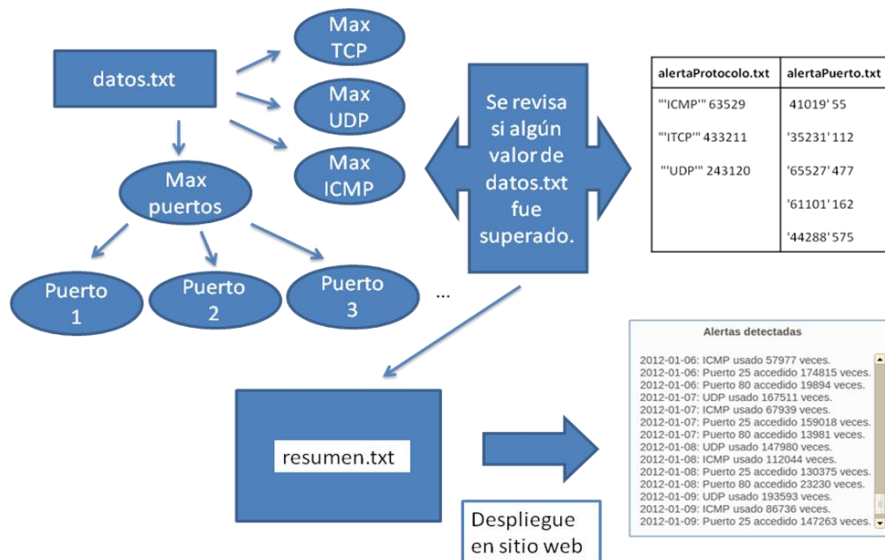


Diagrama 7: Representación del sistema de alertas. 2012.

5. Despliegue de gráficos en sitio web:

Todos los archivos XML creados para los gráficos viajan desde el computador del laboratorio hacia el servidor web para desplegarlos en el sitio, este copiado se hace a través de una tarea programada (crontab). Así, cada día se cuenta con los gráficos del día anterior para ser visualizados.

5.a. Selección de tipo de gráfico:

Una vez que se ha ingresado al sitio web, el usuario debe seleccionar (con un radio-button) qué tipo de gráfico quiere visualizar. En la parte superior se despliegan los gráficos y videos creados con las herramientas AfterGlow, Moncube, TCPstat y EtherApe. Si el usuario quiere ver alguno de éstos gráficos debe hacer click en él y presionar “Ver”.

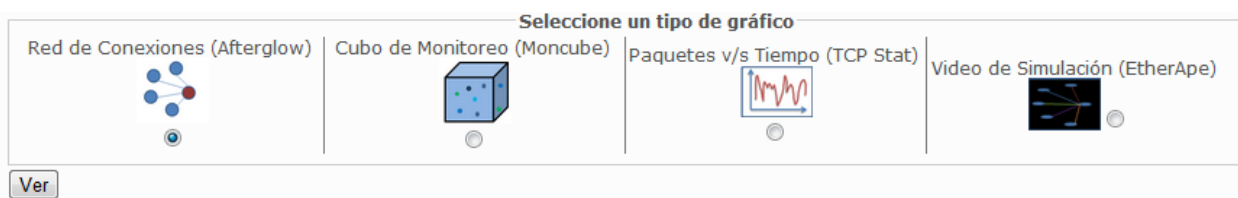


Figura 25: Interfaz para seleccionar gráfico a ver (generados a partir de herramientas existentes). Imagen Original Sitio web. 2012.

En la parte central de la página aparecen los gráficos creados a partir de los XML recién calculados. En este caso el usuario debe seleccionar qué categoría de gráfico desea ver y hacer click en “Ver”:

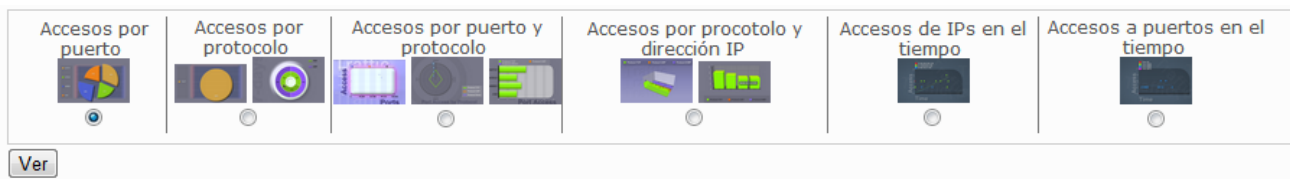


Figura 26: Interfaz para seleccionar gráfico a ver (generados a partir de XMLs). Imagen Original Sitio web. 2012.

Para ver los gráficos de estado de la red, el usuario deberá hacer click en “Ver” teniendo seleccionada la opción:



Figura 27: Interfaz para ver gráficos asociados al estado de la darknet. Imagen Original Sitio web. 2012.

5.b. Selección de fecha:

Una vez que el tipo de gráfico ya ha sido seleccionado, el sitio redirige al usuario a un calendario. En él se debe escoger el tráfico de qué día se quiere revisar. Si se ha escogido alguno de los gráficos (o videos) de la sección superior (AfterGlow, Moncube, TCPstat, EtherApe), se debe escoger un día pasado, es decir, en este caso no se debe pulsar sobre la fecha actual (marcada con un triángulo en la esquina del casillero). Esto es porque los gráficos se calculan con un día de desfase, entonces los gráficos correspondientes al día de “hoy” estarán disponibles “mañana”.

Además, el calendario lleva un registro de los días pasados en donde no hay gráficos (por un posible corte de luz, error en el copiado de las imágenes desde el computador del laboratorio al servidor web, etc.). Éstos días se calculan revisando los directorios del servidor web y se almacenan en un archivo de texto, que luego es leído por el calendario el cual deshabilita esas fechas para que no se puedan seleccionar. A continuación se muestra una vista del calendario con algunos días tachados y un día pasado seleccionado:



Figura 28: Calendario sitio web (herramientas existentes). Imagen Original Sitio web. 2011.

5.c. Despliegue de gráficos AfterGlow, Moncube, EtherApe, TCPstat:

Una vez que se presione “Visualizar” existirán distintas opciones dependiendo del tipo de gráfico o video.

a) Para el caso de AfterGlow, se debe seleccionar una hora del día y luego presionar el botón “Procesar”. Así se desplegarán las 6 imágenes asociadas a esa hora (por ejemplo si se eligen las 02, se mostrarán las imágenes de 02:00 a 02:10, de 02:00 a 02:20, de 02:00 a 02:30, de 02:30 a 02:40, de 02:30 a 02:50 y de 02:30 a 02:59). De todas maneras existen links para moverse a la hora siguiente o anterior.

b) Para los gráficos de Moncube no se debe seleccionar una hora previamente, solo se debe pulsar el botón “Procesar” y se desplegarán los 3 gráficos diarios correspondientes a vistas distintas del cubo con contiene todo el tráfico del día seleccionado.

c) En el caso de TCPstat primero se debe seleccionar la hora a visualizar y luego el tipo de gráfico (Paquetes por protocolo v/s tiempo o Paquetes SYN-ACK v/s tiempo). Antes de pulsar el botón “Procesar” es recomendable fijarse en los gráficos que ya se encuentran desplegados. Éstos corresponden a los gráficos diarios de ambos tipos existentes en esta categoría. Entonces, si en alguno de éstos se observa un pick a cierta hora, sería recomendable seleccionar esa hora para verla en detalle y analizar lo ocurrido. Para este gráfico nuevamente existen links para moverse a la hora siguiente o anterior.

d) Para los videos de EtherApe se selecciona una hora en particular y se pulsa “Procesar”. En la siguiente vista se desplegará una simbología de los colores de los protocolos que aparecen en los videos. Además estarán disponibles los links para ver los 4 videos existentes por cada hora. (Ejemplo, si se seleccionan las 00 hrs, se desplegarán los videos de las 00:00-00:10, de las 00:15-00:25, de las 00:30-00:40 y de las 00:45-00:55.) Nuevamente existen links para moverse a las horas siguientes y anteriores.

Para todas estas herramientas existen links al final de la página para cambiarse a otro tipo de gráfico (o video incluso). Esto facilita el uso del sitio dado que el usuario no tendrá que volver a la página de Inicio en cada momento.

5.d. Despliegue de gráficos pre-calculados (por omisión):

Si se escogen los gráficos generados a partir de los archivos XML cuyo procesamiento fue explicado con anterioridad primero se debe seleccionar un día en el calendario. La fecha escogida debe ser un día pasado si se quiere ver un gráfico pre-calculado, en caso de querer crear un gráfico con parámetros ad-hoc se puede escoger el día de “hoy”. En este calendario también aparecerán días tachados, pero el criterio no es el mismo que en el caso anterior (calendario de AfterGlow, EtherApe etc.): un día se marca como deshabilitado en caso de que no se cuente con el archivo de tráfico en binario (en el servidor web, ya que éste maneja los mismos pcaps que el computador del laboratorio, encargado de la creación de todos los gráficos). Lo común sería pensar que si un gráfico no se puso pre-calculado en el laboratorio, o si el XML no se copió de la correcta forma desde el computador del laboratorio al servidor web, el día debería aparecer como deshabilitado, pero no es así. Puede ser que el XML no exista pero el usuario igual tendrá la posibilidad de crear su propio gráfico en esta sección si y sólo si existe el pcap en binario, por lo tanto es éste el criterio de decisión.



Figura 29: Calendario sitio web (gráficos a partir de archivos XML). Imagen Original Sitio web. 2011.

Una vez que se pulsa el botón “Crear”, se redirige al usuario a la página donde se encuentra el gráfico pre-calculado y además un formulario para que el analista cree uno propio.

La traducción desde el XML a un gráfico en Flash es trabajo de Maani Charts. El requisito es que la carpeta con el sistema “Maani” se encuentre en un determinado directorio para acceder a él y lea los XML que se le pasan como parámetro. Maani Charts entrega un método en javascript (*insertFlash*) en su documentación diseñado para cargar el gráfico que uno desee dándole el nombre como parámetro. Éste método inserta el Flash en el lugar de la página que uno desee.

Arriba de todos los gráficos relacionados con los puertos (todos menos el de IPs en el tiempo), despliegan información de los puertos más accedidos y su cantidad de accesos. Estos datos son los que se escribieron en un archivo de texto durante la ejecución de *graficosFlash.sh* que luego se copió en el servidor web junto a todos los archivos XML de un determinado día.

Abajo de cada gráfico existe un links que da la posibilidad de descargar el XML original calculado en el computador del laboratorio. Además existen vínculos a otros tipos de gráficos para facilitar la visualización (de la misma forma que las otras herramientas mencionadas anteriormente lo hacían).

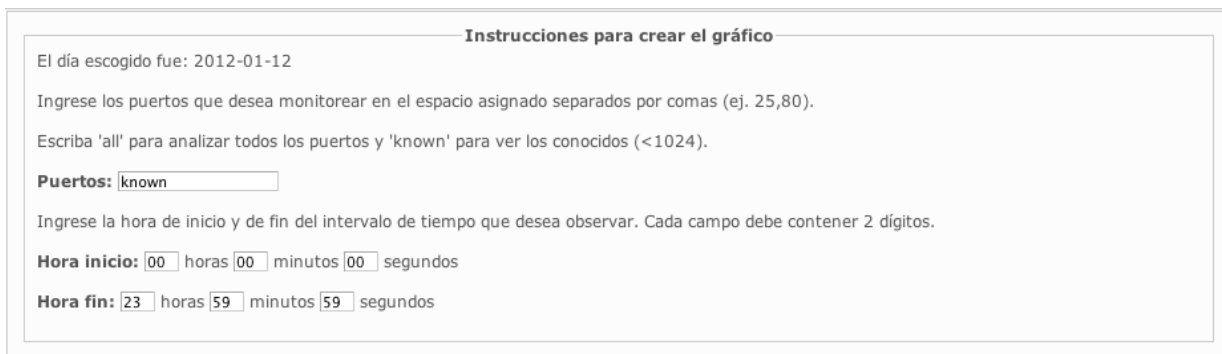
En la parte inferior de la página se puede apreciar un formulario para que el analista cree gráficos con valores personalizados. Éstos valores son tomados por ciertos scripts para la creación de nuevos XML, lo cuales se copian al servidor web y apenas el proceso haya finalizado se despliegan en el sitio.

Para el caso de los gráficos de estado la situación es diferente, luego de haber seleccionado la fecha en el calendario, aparecen los gráficos de los 10 puertos más accedidos en la darknet a través del protocolo UDP, los 10 puertos más accedidos a través de TCP y los 10 países que más han accedido a la darknet. En este caso no aparece el formulario mencionado anteriormente, la idea es que día a día estos gráficos informen del estado de la darknet (puertos y países) pero para saber mayor detalle al respecto el usuario debe utilizar el resto de los gráficos disponibles.

6. Creación de gráficos a pedido

6. a. Formularios:

Para la mayoría de los gráficos el formulario existente es el siguiente (excepto IP y puertos en el tiempo):



Instrucciones para crear el gráfico

El día escogido fue: 2012-01-12

Ingrese los puertos que desea monitorear en el espacio asignado separados por comas (ej. 25,80).

Escriba 'all' para analizar todos los puertos y 'known' para ver los conocidos (<1024).

Puertos:

Ingrese la hora de inicio y de fin del intervalo de tiempo que desea observar. Cada campo debe contener 2 dígitos.

Hora inicio: horas minutos segundos

Hora fin: horas minutos segundos

Figura 30: Formulario para crear gráficos. Imagen Original Sitio web. 2011.

En él se especifican los puertos por los cuales se desea filtrar el tráfico (existiendo la posibilidad de escribir “all” y “known”) y el intervalo de tiempo que se quiere visualizar.

Para el gráfico de IPs en el tiempo el formulario es más complejo. Éste está compuesto por la lista que se calculó al inicio del proceso, que contiene las direcciones IP, número de accesos, país de origen e ISP. Esta lista se puede ordenar por cualquiera de los criterios mencionados (las distintas columnas). Junto a cada dirección IP existe un checkbox que el usuario puede seleccionar, en caso de hacerlo los casilleros de IPs ubicados más abajo se irán rellenando con éstos valores. Se puede seleccionar un máximo de 3 IPs. En caso de hacer click en más, los casilleros contendrán los valores de las últimas 3 direcciones presionadas. Esto fue creado para facilitar la labor del analista, de esta manera puede saber qué direcciones generaron más accesos y cuáles son sus características, y así seleccionarlas para conocer su comportamiento en el tiempo. Dicho formulario tiene la siguiente forma:

Instrucciones para crear el gráfico

El día escogido fue: 2012-01-12

Ingrese las direcciones IP que desea monitorear

Algunas recomendaciones de direcciones IP (puede seleccionar máximo 3 o escribirlas en los casilleros)

Dirección IP	Accesos diarios	País	ISP
<input checked="" type="checkbox"/> 77.73.235.46	37624	RU	MEGANET-AS JSC _Meganet_
<input checked="" type="checkbox"/> 114.42.155.146	29172	TW	HINET Data Communication Bus
<input checked="" type="checkbox"/> 194.84.72.51	26624	RU	ROSPRINT-AS LLC Equant
<input type="checkbox"/> 164.77.239.174	25063	CL	ENTEL CHILE S.A.
<input type="checkbox"/> 164.77.239.18	23444	CL	ENTEL CHILE S.A.
<input type="checkbox"/> 164.77.239.201	19754	CL	ENTEL CHILE S.A.
<input type="checkbox"/> 189.23.132.117	14250	BR	Embratel
<input type="checkbox"/> 58.248.217.133	13805	CN	CNCGROUP-GZ China Unicom G
<input type="checkbox"/> 200.6.117.121	13629	CL	Ingeniería e Informática Asocia
<input type="checkbox"/> 213.198.206.143	13585	RS	YUNET-AS YUnet International c
<input type="checkbox"/> 88.247.188.136	11182	TR	TTNET Turk Telekomunikasyon
<input type="checkbox"/> 125.128.138.30	10890	KR	KIXS-AS-KR Korea Telecom
<input type="checkbox"/> 190.22.96.112	9933	CI	TELEFÓNICA CHILE S.A.

Direcciones IP:

IP 1

IP 2

IP 3

Ingrese la hora de inicio y de fin del intervalo de tiempo que desea observar. Cada campo debe contener 2 dígitos.

Hora inicio: horas minutos segundos

Hora fin: horas minutos segundos

Figura 31: Formulario para crear gráficos (IPs en el tiempo). Imagen Original Sitio web. 2011.

Para el caso del gráfico de puertos en el tiempo, el formulario es similar, en este caso se despliega la lista con los puertos accedidos (lista de puertos calculada junto a la lista de IPs en un inicio) y su cantidad de accesos diarios. En este caso se puede seleccionar un máximo de 5 puertos. A continuación se despliega dicho formulario:

Instrucciones para crear el gráfico

El día escogido fue: 2012-01-12

Ingrese los puertos que desea monitorear en los espacios asignados.

Algunas recomendaciones de Puertos

Puerto	Accesos diarios
<input type="checkbox"/> 25	176272
<input type="checkbox"/> 80	61889
<input type="checkbox"/> 3128	48117
<input type="checkbox"/> 3389	41577
<input type="checkbox"/> 23	33526
<input type="checkbox"/> 8080	28855
<input type="checkbox"/> 514	22606
<input type="checkbox"/> 1434	21180
<input type="checkbox"/> 5060	16393
<input type="checkbox"/> 53	9593
<input type="checkbox"/> 22	8806
<input type="checkbox"/> 22487	6490
<input type="checkbox"/> 4899	4774

Puertos:
 Puerto 1
 Puerto 2
 Puerto 3
 Puerto 4
 Puerto 5

Ingrese la hora de inicio y de fin del intervalo de tiempo que desea observar. Cada campo debe contener 2 dígitos.

Hora inicio: horas minutos segundos

Hora fin: horas minutos segundos

Figura 32: Formulario para crear gráficos (Puertos en el tiempo). Imagen Original Sitio web. 2011.

En todos los formularios existe el botón “Validar” (aunque haya sido omitido en las imágenes) que al presionarlo ejecuta métodos de validación de todos los campos tanto en javascript como en php para evitar problemas de seguridad (validación positiva). En caso de que algún campo sea vulnerado se desplegará una alerta y el usuario deberá reintentar el ingreso de datos.

Cabe destacar que cuando un usuario ingresa a esta página (contenedora del formulario) se le asocia un número aleatorio. Esto con el fin de controlar distintos accesos a la página y que cada usuario reciba el gráfico que solicitó y no el de otro usuario (a cada nuevo XML se le adjunta el número identificador del usuario). Este número aleatorio es considerado en todos los scripts como un nuevo parámetro, es por esto que se tuvo que crear una copia de todos los generadores de gráficos y métodos en python para que estén preparados para recibir un nuevo valor. De todas formas su funcionamiento es exactamente el mismo, solo que utilizan el número aleatorio para diferenciar los archivos intermedios (cortado, uno, temp, etc.) y finales (XML).

Luego de la validación se ejecuta la creación del gráfico solicitado, solo en caso de que no haya campos mal ingresados. Tal como fue explicado en la sección 4 (Solución Propuesta) existen variados gráficos pero éstos se agruparon por categorías (tal como se aprecian en el radio button de la página de inicio del sitio web). En esta sección se realizan variadas llamadas a los generadores de gráficos (ubicados en el

computador del laboratorio) dependiendo de qué opción se haya elegido (qué grupo de gráficos se desea ver).

6. b. Generación de los gráficos:

Los generadores de gráficos son iguales a los explicados con anterioridad, sólo que sus nombres ahora tienen un número 2 concatenado. Esto es para diferenciar entre los generadores utilizados para crear gráficos con valores por omisión, y entre los hechos por el usuario, ya que éstos últimos manejan un número aleatorio asociado a cada usuario para evitar sobre escritura en los archivos XMLs creados.

En cada uno de los casos siguientes se ejecuta el generador respectivo en el computador del laboratorio. Para esto se utiliza un script llamado lanzar, descrito más adelante, que recibe como parámetro el script a correr y sus respectivos parámetros, donde:

- a) hi, mi, si = hora inicial, minuto inicial, segundo inicial (campos del formulario).
- b) hf, mf, sf = hora final, minuto inicial, segundo inicial (campos del formulario).
- c) arch = nombre del archivo contenedor del tráfico (formado por "salida-" y la fecha seleccionada en el calendario de la vista anterior).
- d) random = número aleatorio asociado a cada usuario.
- e) arrPuertos = puertos seleccionados (campo existente para todos los gráficos a excepción de IPs en el tiempo y Puertos en el tiempo).
- f) puerto1, puerto2, puerto3, ..., ip1, p2, ... : Puertos e IPs seleccionadas en las listas desplegadas en los gráficos de IPs en el tiempo y Puertos en el tiempo.

Por ejemplo, para la creación de los gráficos: Accesos por puerto, Accesos de direcciones IPs en el tiempo y Accesos a puertos en el tiempo, las llamadas respectivas serían las siguientes:

```

if($tipo=="puerto"){
    $output = exec("/var/www/html/darknet/darknet/lanzar
/home/rfaccilo/Desktop/generador2.sh ${hi}h${mi}m${si}s
${hf}h${mf}m${sf}s $arch pie $random $arrPuertos");
}
if($tipo=="tiempoIP"){
    $output_i = exec("/var/www/html/darknet/darknet/lanzar
/home/rfaccilo/Desktop/generadorTime2.sh ${hi}h${mi}m${si}s
${hf}h${mf}m${sf}s $arch timeIP $random $ip1 $ip2 $ip3 $ip4 $ip5");
}
if($tipo=="tiempoPuerto"){
    $output_h = exec("/var/www/html/darknet/darknet/lanzar
/home/rfaccilo/Desktop/generadorTime2.sh ${hi}h${mi}m${si}s
${hf}h${mf}m${sf}s $arch time $random $puerto1 $puerto2 $puerto3
$puerto4 $puerto5");
}

```

Tabla 31: Ejemplos de llamadas a la creación de gráficos desde el sitio web.

El código del script lanzar utilizado para correr los generadores de gráficos en el computador del laboratorio es el siguiente:

```

#!/usr/bin/expect --
set prompt "(%|#|\\$|%\\]) $"
spawn ssh rfaccilo@172.17.69.119
expect "*?assword:*" {
    send -- "*****\r"
}
expect -re "$prompt"
send -- "$argv\r"
set timeout 120
expect -re "$prompt"
send -- "exit\r"

```

Tabla 32: Código del script lanzar.

En él se conecta a través de ssh al computador del laboratorio y una vez logeado ejecuta los scripts (*generador2.sh*, *generadorTime2.sh* y *generadorRing2.sh*) con sus parámetros respectivos. Se optó por

no utilizar llave pública para resguardar la seguridad (es el usuario apache quien está ejecutando los scripts, no se desea que pueda correr cualquier programa del computador del laboratorio, solo los precisos).

6. c. Envío y despliegue de archivos XMLs:

Cada uno de los generadores de gráficos a pedido, tienen como último comando enviar los XML al servidor web con scp. Mientras que el archivo es creado y enviado, el sitio web está cargado pero esperando por el archivo XML para desplegar el gráfico con Maani Charts. Cada segundo se consulta si el archivo se encuentra en el directorio especificado, en caso de que así sea, se llama al método "insertFlash" proveniente de la biblioteca de Maani Charts (mencionado anteriormente) pero con algunas modificaciones en el código (si el archivo no está, se espera por él). Comúnmente, al llamar a "insertFlash" el gráfico se inserta en la posición de la página donde el método es llamado. Pero en este caso, la página ya se encuentra cargada y se desea que el gráfico, cuando llegue al servidor web, aparezca en una posición específica. Es por esto que se tuvo que hacer algunos cambios en el código de Maani Charts para posicionar el gráfico en el lugar correcto y que no se abra una nueva página que contenga sólo el gráfico.

Mientras que el gráfico se crea y viaja al servidor web, aparece el texto "Cargando..." donde los puntos suspensivos aumentan en cantidad. Esto le da al usuario feedback de que el proceso está en ejecución. Si por algún motivo el sistema falla, se pierde conexión con el servidor, o simplemente el proceso toma más de 10 minutos en finalizar, la página despliega una advertencia diciendo que probablemente se ha perdido conexión con el servidor y entrega la posibilidad de volver al inicio o seguir esperando por el gráfico.

El tiempo que puede tardar el despliegue de un gráfico creado por el usuario es variable. Dependerá del filtro que se realice, de la carga del procesador del computador del laboratorio, de la velocidad de la red dentro de la Universidad, de la cantidad de datos existentes el día seleccionado etc. Según las pruebas realizadas ningún gráfico tarda más de 10 minutos en ser desplegado (por esto la advertencia se lanza en ese momento), pero en caso de que sea necesario más tiempo, el usuario siempre tendrá la posibilidad de seguir esperando (en 10 minutos más se desplegará la alerta nuevamente).

En el Diagrama 8 se resume el proceso de creación de gráficos ad-hoc por analista desde el sitio web:

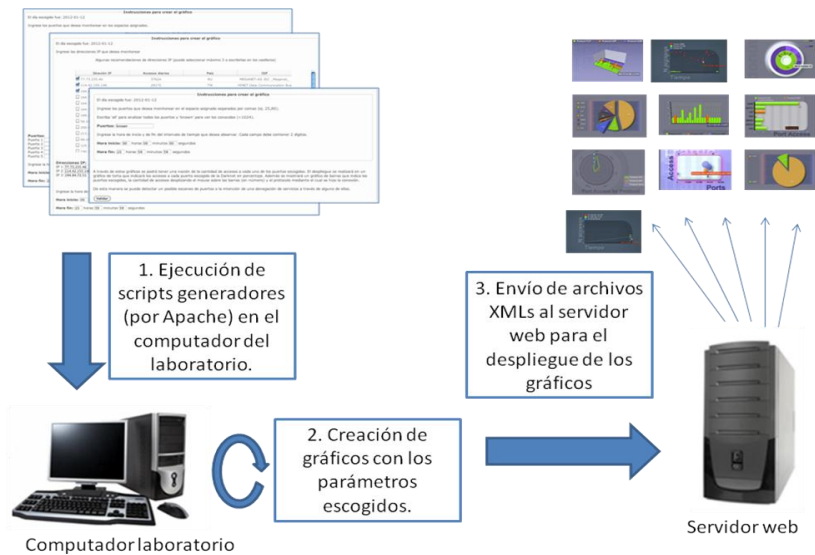


Diagrama 8: Representación del sistema de creación de gráficos ad-hoc. 2012.

Existen ocasiones en que el gráfico especificado por el usuario no desplegará datos (esto puede ocurrir si por ejemplo se filtra el tráfico por un puerto que no fue accesado en todo el día). En caso de que esto ocurra, se visualizará la siguiente imagen:

The chart_data must have at least one data value.

Figura 33: Mensaje en caso de no haber datos. Maani Charts. 2011.

De suceder se recomienda filtrar el tráfico con otro puerto o dirección IP o ampliar el intervalo horario escogido.

Existe otro caso desfavorable que ocurre cuando son demasiados los datos a desplegar y Flash arroja alguno de los siguientes mensajes de error:

Check Attribute: Error: Error #1502

parser: Error: Error #1502

Figura 34: Mensaje en caso error. Maani Charts. 2011.

En este caso es recomendable volver atrás y solicitar un gráfico con filtros más exigentes y así obtener una visión más específica.

Cabe destacar que junto a cada gráfico existe un link que da la alternativa de descargar el XML que se utilizó para la creación de éste. Tal como se mencionó anteriormente, el formato de dicho XML es el determinado por Maani Charts para su interpretación. Con la intención de compartir esta información

con otros CERTS, se propone que a partir de los XML ya creados (contenedores de todos los datos a procesados) se genere uno nuevo en algún formato estándar como es IODEF (Incident Object Description and Exchange Format) utilizado a nivel mundial [IODEF12].

7. Ejemplo realizado en Big Data

Dada la alta probabilidad de que la cantidad de paquetes que llegan a la darknet vaya en aumento a lo largo del tiempo, se optó por generar un gráfico cuya creación esté vinculada al procesamiento de grandes cantidades de información. Es importante que este gráfico sea útil para el analista, por ejemplo, un estudio interesante sería evaluar la cantidad de accesos diarios existentes a la red durante el último año. Esto le dará al analista una visión de cómo han variado los ataques a la darknet, qué meses han sido más críticos, en qué épocas del año existe menos actividad etc.

A continuación se explicará cómo se prepara la información que se deberá procesar para la creación de este gráfico. Cada día se toma el archivo contenedor del tráfico en binario de ayer, se pasa a texto plano y en cada línea se escribe la fecha del día. Posteriormente se borra todo el texto que no corresponda a la fecha del día (direcciones IPs, puertos etc.), de esta manera se tiene un archivo que sólo tiene líneas iguales (la fecha). El largo del archivo equivale a la cantidad de accesos ocurridos durante ese día a la darknet. A modo de ejemplo, si el día 2012-01-07 ocurrieron 3 accesos, el archivo asociado a esa fecha será así (el formato utilizado se explicará más adelante.):

```
newDate2012x01x07  
newDate2012x01x07  
newDate2012x01x07
```

Tabla 33: Ejemplo de archivo para el día 2012-01-07.

Una vez que se tiene finalizado el archivo diario, éste se concatena a un gran archivo que contiene las líneas de los días pasados del mes (existe un archivo por cada mes). Por lo tanto el texto resultante (mensual) tendrá la forma:

```
newDate2012x01x01  
newDate2012x01x02  
newDate2012x01x02  
newDate2012x01x03  
newDate2012x01x03  
newDate2012x01x04  
newDate2012x01x05  
newDate2012x01x06  
newDate2012x01x07
```

```
newDate2012x01x07
newDate2012x01x07
```

Tabla 34: Ejemplo de archivo para el mes 01.

Con datos reales, este archivo mensual pesa alrededor de 1 GigaByte (GB), si el gráfico a realizar considera un año de tráfico, la cantidad total de datos a procesar se eleva a 12 GB. Procesar toda esta información con los equipos y scripts detallados anteriormente sería lento. Es por esto que se optó por utilizar las herramientas proporcionadas por Amazon Web Services [AWS12]. Entre los servicios ofrecidos, existe Map Reduce [AEMR12] que contiene un contador de palabras (a modo de ejemplo implementado por Amazon: *wordSplitter.py*) que es justamente lo que se necesita en este caso. El código es el siguiente:

```
#!/usr/bin/python
import sys
import re

def main(argv):
    pattern = re.compile("[a-zA-Z][a-zA-Z0-9]*")
    for line in sys.stdin:
        for word in pattern.findall(line):
            print "LongValueSum:" + word.lower() + "\t" + "1"

if __name__ == "__main__":
    main(sys.argv)
```

Tabla 35: Código de wordSplitter.py.

En este caso, se busca contar cuántas veces aparecen repetidas las fechas en los archivos creados (es por esto que las fechas se escribían con ese extraño formato, sólo usando números y letras para que parezcan “palabras”). Para el ejemplo descrito se busca tener lo siguiente:

```
newDate2012x01x01      1
newDate2012x01x02      2
newDate2012x01x03      2
newDate2012x01x04      1
newDate2012x01x05      1
newDate2012x01x06      1
newDate2012x01x07      3
```

Tabla 36: Ejemplo de accesos por día.

Lo que se utiliza como datos de entrada para el gráfico final. La creación del archivo diario, concatenación con el archivo “mensual” y la interacción con Amazon se realiza mediante un script, el que se detalla a continuación, por partes:

Primero se calcula la fecha del día de ayer.

```
#!/bin/bash
HOME_DIR=/home/rfaccilo/Desktop
cd $HOME_DIR/capturas

year=$(date --date='1 day ago' +%Y)
month=$(date --date='1 day ago' +%m)
day=$(date --date='1 day ago' +%d)
dia=$year"-"$month"-"$day
```

Se pasa el archivo binario a texto con el comando tcpdump, y se remueven ciertos caracteres tal como en casos anteriores.

```
/usr/sbin/tcpdump -nqtr salida-$dia &> big-uno-${dia}.txt
sed '1d' big-uno-${dia}.txt > big-temp-${dia}.txt
cat big-temp-${dia}.txt | sed 's/\. /g' | sed 's:// /' > big-
${dia}.txt
```

Al principio de cada línea de tráfico existe la palabra “IP”, ésta es reemplazada por la fecha calculada pero considerando el formato necesario (sólo letras). Luego se deja en el documento solamente la fecha (primera columna, similar a haber eliminado todo lo distinto a ésta.)

```
sed -i 's/IP/newDate'${year}'x'${month}'x'${day}'/' big-$dia.txt
awk '{print $1}' big-$dia.txt > big2-$dia.txt
```

Se consulta si existe el archivo mensual (para el mes actual), en caso de que así sea se concatena la información del día a éste.

```
if [ -f salida-bigdata-${month}-${year}.txt ]; then
    cat salida-bigdata-${month}-${year}.txt big2-${dia}.txt >
salida-bigdata2-${month}-${year}.txt
    mv salida-bigdata2-${month}-${year}.txt salida-bigdata-
${month}-${year}.txt
```

En caso contrario se crea el archivo antes de concatenar. Se eliminan todos los archivos intermedios creados durante el proceso.

```
else
    touch salida-bigdata-$(month)-$(year).txt
    cat salida-bigdata-$(month)-$(year).txt big2-$(dia).txt >
salida-bigdata2-$(month)-$(year).txt
    mv salida-bigdata2-$(month)-$(year).txt salida-bigdata-
$(month)-$(year).txt
fi
rm big*
```

Para comenzar la interacción con Amazon es fundamental crearse una cuenta y luego un “bucket” en S3 [AS312] para guardar los datos que se utilizarán como entrada del MapReduce. Para utilizar este software de procesamiento de datos se debe instalar en el computador local (equipo del laboratorio) y seguir el instructivo disponible en los tutoriales de Amazon [AMTU12].

Continuando con el script, se deben subir los datos recién generados al “bucket” disponible en Amazon S3 [AS312]. Para esto se utilizó la herramienta s3cmd [SCMD12] que permite realizar operaciones sobre el sistema de almacenamiento. Hay que destacar que primero se debe borrar, del bucket, el archivo del mes actual y los resultados obtenidos el día anterior (dado que la sobre-escritura no es automática).

```
s3cmd del -r s3://mylog-uri-rfaccilo/datos/salida-bigdata-$(month)-
$(year).txt
s3cmd del -r s3://mylog-uri-rfaccilo/results
```

Ya se está en condiciones de subir los datos almacenados en el archivo de texto mensual utilizando el comando s3cmd.

```
s3cmd put /home/rfaccilo/Desktop/capturas/salida-bigdata-$(month)-
$(year).txt s3://mylog-uri-rfaccilo/datos/
```

A continuación se corre Map Reduce (en el directorio donde se encuentra el software instalado según el tutorial de Amazon) con los siguientes parámetros:

- mapper: el contador de palabras wordSplitter.py (código más descrito anteriormente)
- input: el archivo recién subido a S3.
- output: directorio donde se guardarán los resultados.
- reducer: “aggregate” suma el conteo de palabras entregado por el mapper.

- master-instance-type m1.large: instancia a utilizar de Amazon Elastic Cloud Computing (EC2) [AEC2] (detalles más adelante).

```
cd /home/rfaccilo/elastic-mapreduce-cli
./elastic-mapreduce --create --stream \
    --mapper s3://elasticmapreduce/samples/wordcount/wordSplitter.py \
    --input s3n://mylog-uri-rfaccilo/datos/ \
    --output s3n://mylog-uri-rfaccilo/results \
    --reducer aggregate
    --master-instance-type m1.large > jobID.txt
```

Este comando entrega como respuesta un mensaje de confirmación indicando que la tarea fue creada y su respectivo identificador: `Created job flow JobFlowID`. Este mensaje se guarda en un archivo de texto para rescatar luego el identificador al inicio del siguiente ciclo.

```
while :
do
sed -i 's/Created job flow //g' jobID.txt
    jobID=$(awk '{print $1}' jobID.txt)
```

Ahora que se tiene guardado el identificador de la tarea (job) se pregunta por sus detalles con la opción “--describe”. Esto entrega como salida un archivo json (contenedor de toda la información del job) que es almacenado y luego procesado por un código python que rescata su estado (se desea saber cuándo el job ha finalizado).

```
./elastic-mapreduce --describe --jobflow $jobID > details.json
    python2.6 /home/rfaccilo/elastic-mapreduce-cli/parser.py >
estado.txt
    terminado=$(grep "COMPLETED" estado.txt)
```

Si el estado es “COMPLETED” la tarea ha finalizado con éxito, por lo tanto se procede a descargar el resultado del MapReduce (utilizando el script *bajarEnviar.sh*). En caso contrario, se espera un minuto y se vuelve a preguntar por el estado del job.

```
if [ $terminado ];
then
    cd /home/rfaccilo/Desktop
```

```
        ./bajarEnviar.sh
        break
    else
        sleep 60

    fi
done
```

El script de *bajarEnviar.sh* (llamado al momento de finalizar el job) realiza el siguiente procedimiento:

1. Primero se descargan los resultados del MapReduce almacenados en la carpeta *results* en un archivo llamado *part-00000* dentro del bucket con el comando:

```
s3cmd get s3://mylog-uri-rfaccilo/results/part-00000
datosamazon.txt
```

Tabla 37: Llamada para la descarga de datos.

2. Luego se aplican algunos cambios con el comando “sed” para que los datos tengan la forma que se utilizarán en la creación del gráfico.
3. Finalmente se envían al servidor web para que sean leídos por la página de inicio y se despliegue el gráfico utilizando Google Charts.

Cabe destacar que MapReduce corre sobre una instancia de Amazon Elastic Cloud Computing (EC2) la cual debió ser configurada para obtener un mejor rendimiento. Predeterminadamente el usuario cuenta con la instancia más pequeña cuyas especificaciones son las siguientes:

Instancia pequeña:
1,7 GB de memoria
1 unidad informática EC2 (1 núcleo virtual con 1 unidad informática EC2)
160 GB de almacenamiento de instancias
Plataforma de 32 bits
Rendimiento de E/S: moderado

Tabla 38: Detalles de Instancia pequeña.

Utilizando esos recursos, el tiempo tomado por el MapReduce de Amazon, considerando el tráfico de un año, fue de 4 horas y 10 minutos aproximadamente (a esto hay que sumarle el tiempo que toma la creación del tráfico diario, subida de datos a S3, la respectiva descarga de los resultados y envío al servidor web, pero estos valores son despreciables en relación al tiempo tomado por el procesamiento).

Con el fin de acelerar el proceso se utilizó una nueva instancia de EC2, la grande:

<p>Instancia grande</p> <p>7,5 GB de memoria</p> <p>4 unidades informáticas EC2 (2 núcleos virtuales con 2 unidades informáticas EC2 cada uno)</p> <p>850 GB de almacenamiento de instancias</p> <p>Plataforma de 64 bits</p> <p>Rendimiento de E/S: alto</p>
--

Tabla 39: Detalles de Instancia grande.

El proceso se corrió con esta instancia y tiempo del procesamiento cambió a: 44 minutos. Se optó por seguir utilizándola para ahorrar tiempo (y dinero).

El Diagrama 9 muestra una visión general del procesamiento de datos en Amazon, utilizando Elastic Map Reduce:



Diagrama 9: Sistema de procesamiento de datos de la darknet en la nube.2012.

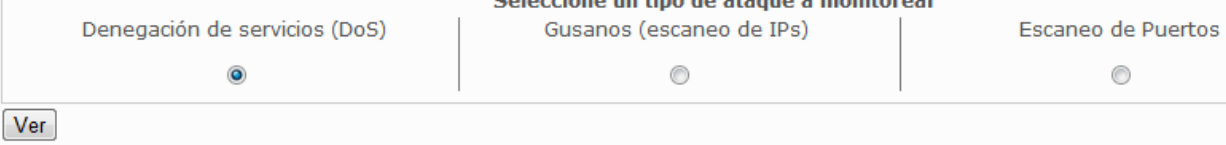
Por otro lado, el procesamiento de los datos de solo 40 días, en el computador del laboratorio, utilizando scripts similares (simplificados) a los ya descritos para la creación de gráficos con Maani Charts, fue de 1 hora y 40 minutos aproximadamente (1,3 horas para el Map y 0,35 para el Reduce). El tiempo necesario para procesar los datos de un año hubiese sido cercano a las 11 horas.

Hay que destacar que el peso máximo que puede tener un archivo que se está subiendo mediante un “put” a S3 es de 5GB. En un inicio este script se realizó creando un gran archivo con todos los accesos ocurridos en un año en su interior. Este archivo pesaba más que lo permitido por lo tanto se optó por crear documentos mensuales. En caso de que el tráfico mensual llegue a sobrepasar el límite (impuesto por Amazon) será necesario hacer un *refactor* de este código y utilizar alguna herramienta como Split [SPL12] para separar los archivos en unos de menor tamaño (cabe destacar que para que esto ocurra el tráfico de la darknet se debería quintuplicar).

Este ejemplo deja en manifiesto que en caso de que la cantidad de accesos diarios aumente significativamente, de tal manera que en un día se tenga la misma cantidad de tráfico que en un año, aún así existen alternativas para seguir procesando los datos y obtener los resultados en un período de tiempo razonable: procesamiento en la nube (Cloud Computing [DG04, JDV09]).

8. Detección de patrones y ataques:

El sitio web cuenta con una sección dedica a la detección de ataques y patrones utilizando los gráficos mencionados con anterioridad. Para esto, el usuario debe seleccionar qué ataque desea monitorear entre alguno de los siguientes:



Seleccione un tipo de ataque a monitorear

Denegación de servicios (DoS)	Gusanos (escaneo de IPs)	Escaneo de Puertos
<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

Ver

Figura 35: Interfaz para seleccionar ataque a monitorear. Imagen Original Sitio web. 2012.

Luego el usuario debe seleccionar una fecha, a través de un calendario, tal como se vio en casos anteriores. Dependiendo del ataque seleccionado, se desplegarán distintos grupos de gráficos:

- Denegación de servicios: Accesos por protocolo (pieProt), Accesos de IPs externas por protocolo (column), Accesos a IPs de la darknet por protocolo (columnAta), Accesos de IPs en el tiempo (timeIP), Accesos a IPs de la darknet en el tiempo (timeAtaIP), TCPstat (por protocolo y por paquetes SYN-ACK) y Moncube.
- Gusanos (escaneo de IPs): Accesos a IPs de la darknet por protocolo (columnAta), Accesos a IPs de la darknet en el tiempo (timeAtaIP), TCPstat (por protocolo y por paquetes SYN-ACK), Moncube y AfterGlow.
- Escaneo de Puertos: Accesos por puerto (pie), Accesos por puerto y protocolo (rect), Accesos a puertos en el tiempo (time) y Moncube.

Estos grupos de gráficos han sido seleccionados porque tienen mayor relación con dichos ataques dada la información que despliegan, por lo tanto el resto de ellos se consideran como complementarios.

Una fuerte hipótesis de esta tesis es que los gráficos creados son útiles para la detección de malware, sin embargo no se alerta al analista sobre un posible ataque. No existe un algoritmo que indique de manera automática “ocurrió cierto ataque”, pero se otorga toda la información para que el usuario lo detecte.

Para probar la hipótesis enunciada se realizaron diversas pruebas. Se realizaron ataques a la darknet utilizando hping3⁵⁰, herramienta que permite crear y analizar paquetes. Luego, utilizando el sitio web, se generaron gráficos en los horarios en que los ataques fueron lanzados con el fin de buscar patrones que puedan ayudarle al analista a identificar un ataque en particular. Dada la enorme cantidad de paquetes (y por lo tanto datos) que genera la herramienta utilizada, se realizaron cada una de las pruebas por alrededor de un minuto y medio. Esto con el fin de no sobrecargar los gráficos ni el computador del

⁵⁰ Más información en el sitio <http://www.hping.org/>.

laboratorio con muchos datos a analizar (una hora por ataque llegó a generar 3Gb de información, mientras que el tiempo escogido generó algo más de 300Mb, tráfico levemente superior al recibido generalmente por la darknet en un día)

Para el caso de Denegación de servicios se enviaron paquetes a la darknet con determinadas características (IPs de la darknet reemplazadas por 1.1.1.x):

- Ataque 1: Desde IPs escogidas aleatoriamente se enviaron paquetes a toda velocidad (--flood) a distintos puertos (partiendo del 80) a una determinada dirección de la darknet:

hping3 -I eth0 --rand-source -p ++80 -S --flood 1.1.1.195

- Ataque 2: Desde IPs escogidas aleatoriamente se enviaron paquetes a toda velocidad (--flood) al puerto 80 a IPs random dentro de la darknet:

hping3 -I eth0 --rand-source -p 80 -S --flood 1.1.1.x --rand-dest

- Ataque 3: Desde IPs escogidas aleatoriamente se enviaron paquetes a toda velocidad (--flood) al puerto 80 a una determinada dirección de la darknet utilizando el protocolo UDP (Denegación de servicio a través de respuestas de un servidor DNS):

hping3 -I eth0 --udp --rand-source --destport 53 -S --flood 1.1.1.195

Algunos ejemplos de patrones detectados tras lanzar los ataques 1,2 y 3 se describen a continuación:

En la Figura 36 se pueden visualizar los ataques de Denegación de Servicios de parte de tres direcciones IPs, ellas en un mismo minuto envían cerca de 200 paquetes en promedio a la red, probablemente haya existido el intento de ataque por la alta cantidad de accesos en tan corto período de tiempo. Las horas escogidas para la creación de este gráfico fueron aquellas donde los ataques 1, 2 y 3 fueron lanzados.



Figura 36: Imagen relacionada con ataque DoS (timelP). 2012.

En la Figura 37 se muestra un alza en la cantidad de paquetes justo en el instante en que se lanzaron ataques de Denegación de Servicios (ataques 1, 2, y 3), nuevamente, la gran cantidad de paquetes enviados en un corto período de tiempo llama la atención.

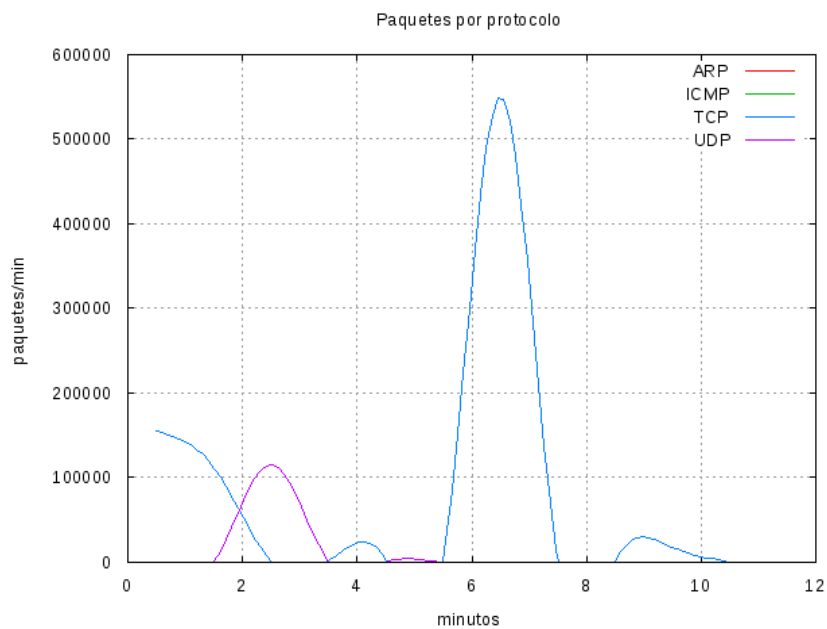


Figura 37: Imagen relacionada con ataque DoS (TCPstat). 2012.

En la Figura 38 se observa claramente cómo una dirección IP realiza una alta cantidad de accesos (350 aproximadamente) en un minuto (tiempo seleccionado en la creación del gráfico) seleccionando los puertos que tenían mayor cantidad de accesos (25, 628, 3389, 80). Esta conducta nuevamente puede llevar a sospechar de un ataque de Denegación de Servicios a la darknet (ataque 1).

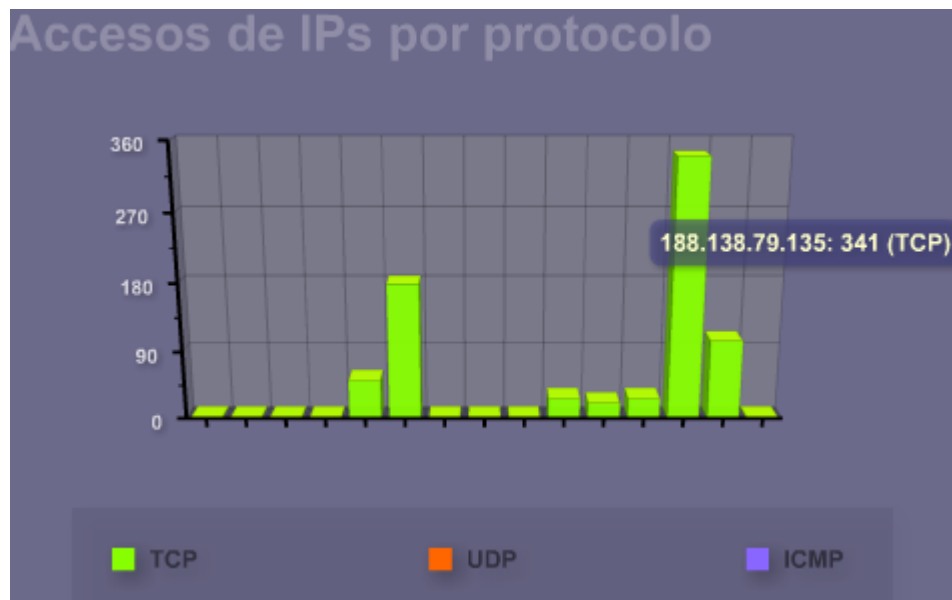


Figura 38: Imagen relacionada con ataque de DoS (column). 2012.

En la Figura 39 se pueden ver las IPs de la darknet que fueron accedidas por el ataque de Denegación de Servicios (ataque 1) considerando los puertos conocidos. Se observa claramente cómo la IP de destino 1.1.1.195 recibió aproximadamente 3000 paquetes en un minuto, cifra alarmante.

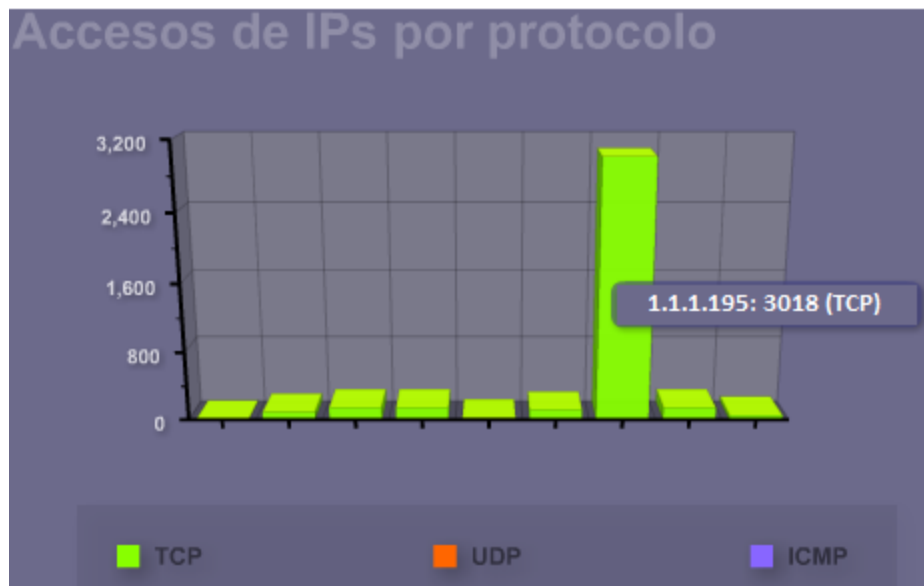


Figura 39: Imagen relacionada con ataque de DoS (columnAta). 2012.

Del mismo modo, se puede ver cómo se realizaron los accesos a la dirección 1.1.1.195 de la darknet con el gráfico de la Figura 40. Con este gráfico el analista también puede detectar que se encuentra frente a una Denegación de Servicios dedicado a la dirección IP mencionada anteriormente debido a la alta cantidad de accesos en tan poco tiempo.



Figura 40: Imagen relacionada con ataque de DoS (columnAta). 2012.

En la Figura 41 se puede visualizar que prácticamente todos los accesos realizados en un determinado período de tiempo (cerca de 1 minuto, tiempo del ataque 3) utilizaron el protocolo UDP. Esto, junto con la información relacionada a puertos (acceso al puerto 53) podría hacer sospechar de un posible ataque de Denegación de Servicios a través de respuestas de un servidor DNS.

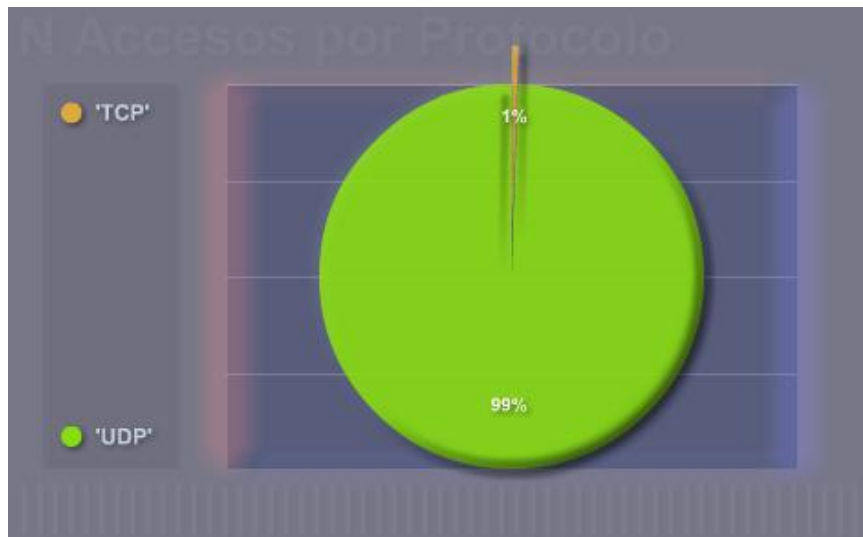


Figura 41: Imagen relacionada con ataque de DoS a través de servidores DNS (pieProt). 2012.

Continuando con el ataque 3, en la Figura 42 se puede ver cómo la IP de destino 1.1.1.195 recibe una alta cantidad de solicitudes durante la Denegación de Servicios. Ahora, el protocolo con que se realizó el ataque fue UDP.

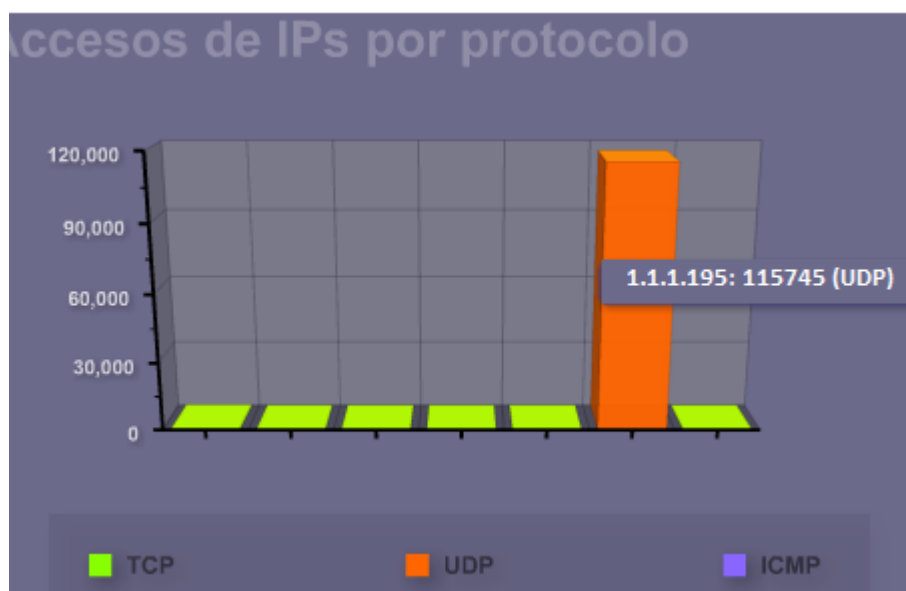


Figura 42: Imagen relacionada con ataque de DoS con protocolo UDP (columnAta). 2012.

Para generar gusanos (escaneo de IPs) se hicieron las siguientes pruebas:

- Ataque 4: Desde una dirección IP en particular se envió un paquete al puerto 80 de 31 direcciones IP de la darknet (con i entre 1 y 31).

hping3 -I eth0 -a 111.122.53.179 -p 80 -c 1 -S 1.1.1.i

- Ataque 5: Desde direcciones IPs aleatorias se envió un paquete al puerto 80 de 31 direcciones IP de la darknet (con i entre 1 y 31).

hping3 -I eth0 --rand-source -p 80 -c 1 -S 1.1.1.i

- Ataque 6: Desde 25 direcciones IPs específicas se envió un paquete al puerto 80 de 250 direcciones IP de la darknet de manera escalonada (primero una IP envía paquetes a la dirección 1, 2, 3 ...10 de la darknet, luego otra dirección IP hace lo mismo con las siguientes direcciones de la darknet 11,12,13 ... 20 y así sucesivamente). Esto con el fin de esconder el ataque (e entre 0 y 25, m entre 0 y 250 variando de 10 en 10).

hping3 --debug -I eth0 -a 111.122.53.e -p 80 -c 1 -S 1.1.1.m

Algunos ejemplos de patrones detectados tras lanzar los ataques 4, 5 y 6 se describen a continuación:

Con la Figura 43 se observa cómo se realizó el ataque de escaneo de IPs número 4 (gusano) donde la dirección IP 111.122.53.179 (en color azul al centro) accede a cerca de 30 direcciones IPs pertenecientes a la darknet, escaneándolas en un corto período de tiempo (cerca de 1 minuto).

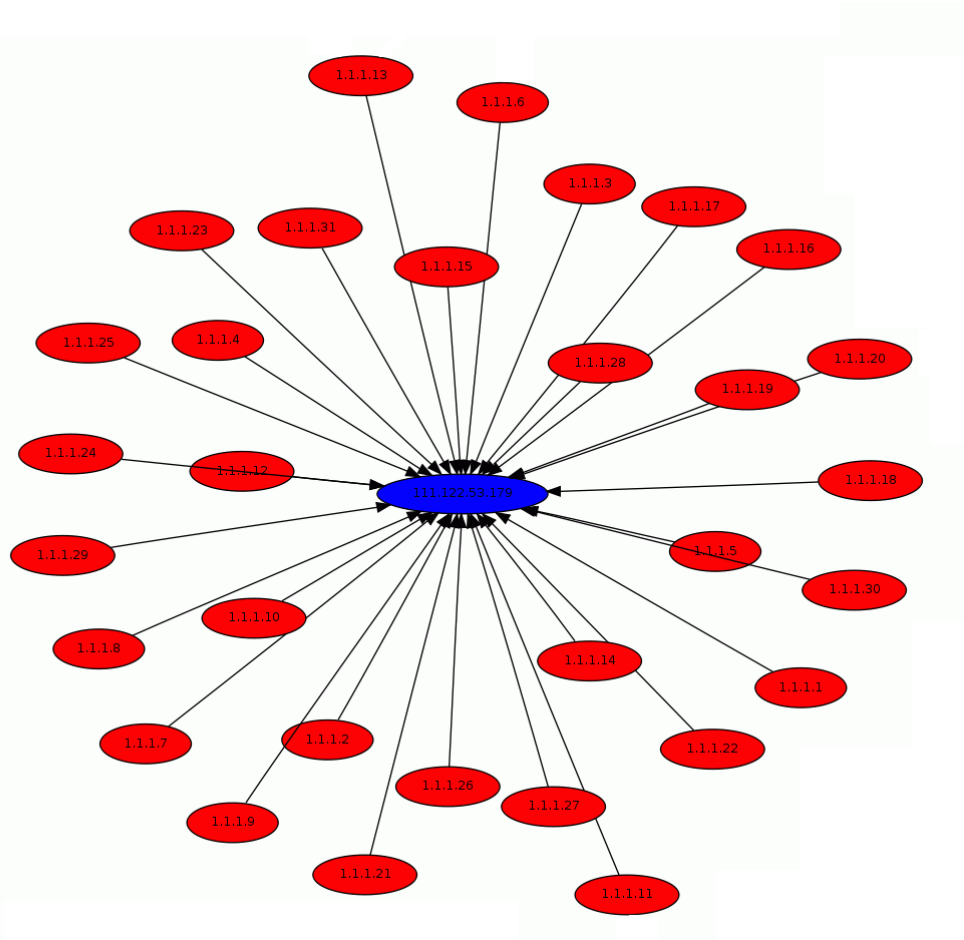


Figura 43: Imagen relacionada con ataque de escaneo de IPs (gusanos) desde una IP (AfterGlow). 2012.

En la Figura 44 se puede apreciar un grupo de IPs de la darknet que fueron accedidas de la misma manera (alrededor de 30 accesos a cada una). Estos paquetes fueron enviados desde la IP externa 111.122.53.179 al igual como se aprecian en el diagrama anterior. Este es otro claro ejemplo de escaneo de IPs (ataque 4).

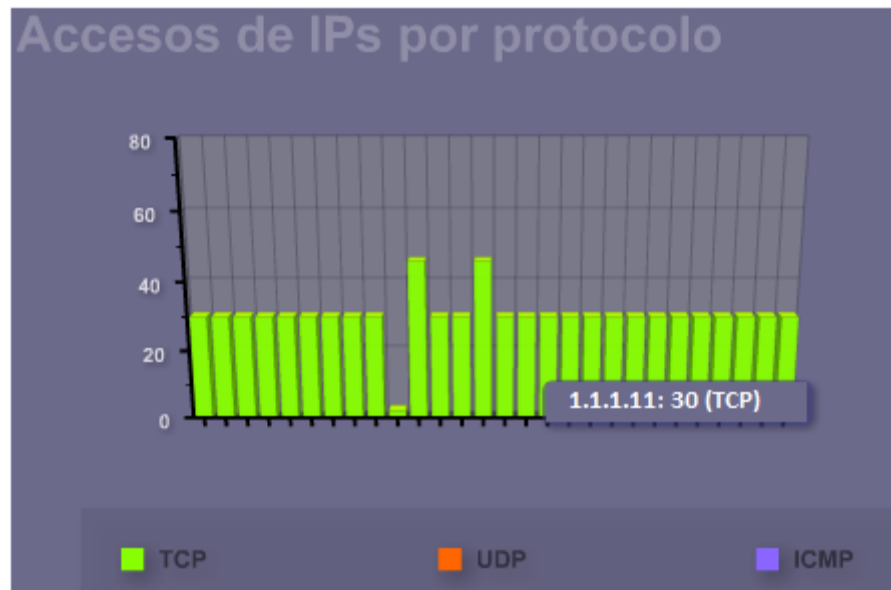


Figura 44: Imagen relacionada con ataque de escaneo de IPs (gusanos) (columnAta). 2012.

En la Figura 45 se observa el comportamiento del ataque 5, un grupo de IPs aleatorias acceden a un grupo de IPs aleatorias dentro de la darknet, una a una. Se puede decir que el grupo de IPs atacantes hacen un escaneo de 31 IPs de la darknet de manera conjunta. Incluso se puede postular que las direcciones 188.138.79.134 y 188.138.79.135 trabajan de manera coludida dado que acceden a las mismas 3 direcciones IP de la darknet: 1.1.1.13, 1.1.1.11 y 1.1.1.39.

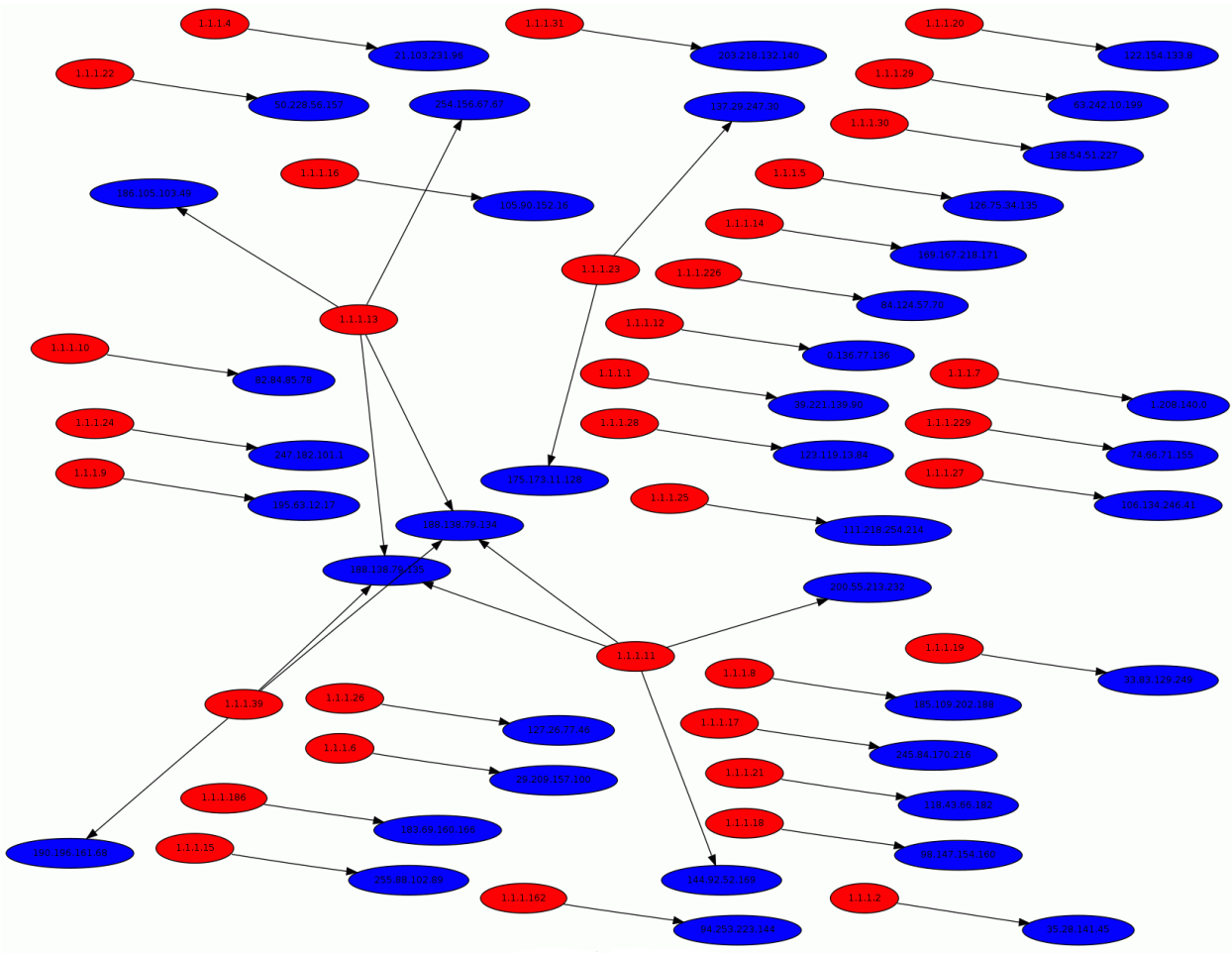


Figura 45: Imagen relacionada con ataque de escaneo de IPs (gusanos) desde varias IPs (AfterGlow). 2012.

En la Figura 46 se puede ver otra forma de escaneo de IPs (ataque 6), donde cada dirección IP escanea un grupo de 10 direcciones IPs de la darknet, en forma escalonada para ocultar el ataque.

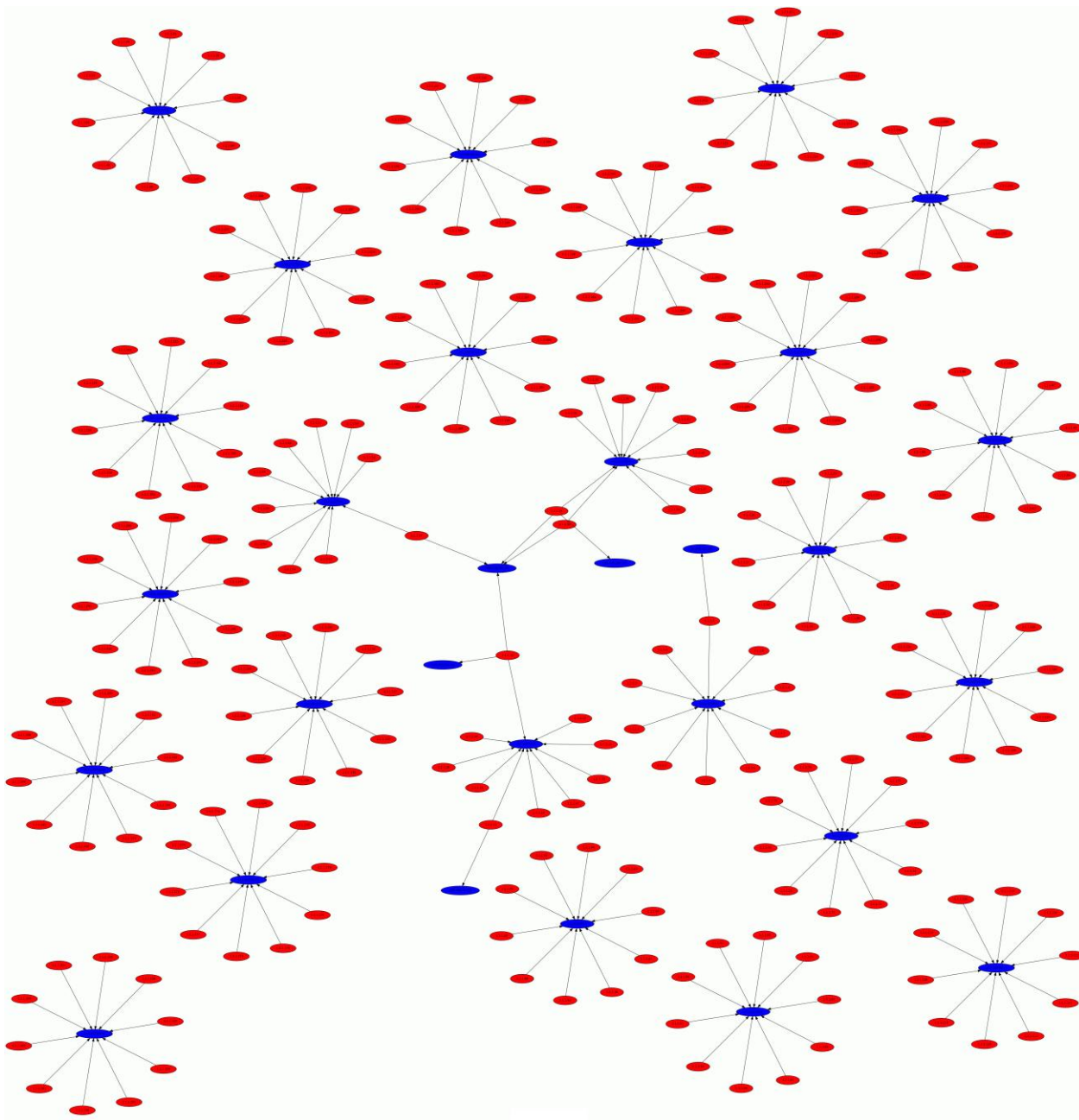


Figura 46: Imagen relacionada con ataque de escaneo de IPs (gusanos) desde varias IPs a grupos (AfterGlow) 2012.

En la Figura 47 y la Figura 48 se puede apreciar el comportamiento del escaneo de IPs (ataques 4, 5 y 6) desde el lado de la darknet. Se ve como las IPs seleccionadas fueron escaneadas durante las 15:38 y 15:39 hrs entre 20 y 190 veces cada una.



Figura 47: Imagen relacionada con ataque de escaneo de IPs (gusanos) a un grupo de IPs de la darknet. 2012.



Figura 48: Imagen relacionada con ataque de escaneo de IPs (gusanos) a un grupo de IPs de la darknet. 2012.

Para detectar que a una dirección IP fuera de la darknet se le está haciendo un ataque de denegación de servicios se realizó lo siguiente:

- Ataque 7: Desde una determinada dirección IPs se enviaron paquetes a toda velocidad a direcciones aleatorias de la darknet con el flag SYN-ACK activado (es decir, una IP responde a la darknet, cosa que no debiese ocurrir a menos que un atacante le haya hecho creer a esa IP que la darknet se comunicó con ella)

hping3 -I eth0 -a 111.122.53.179 -p 80 -SA --flood 1.1.1.x --rand-dest

Algunos ejemplos de patrones detectados tras lanzar el ataque 7 se describen a continuación:

Existen ciertos filtros que no permiten la llegada de paquetes SYN-ACK a la darknet, los cuales serían reveladores de un ataque de Denegación de servicios a una dirección IP de fuera de ésta (ataque 7). Si este filtro no existiera se podrían apreciar patrones indicadores de este ataque y se vería la situación representada en la Figura 49, por ejemplo:

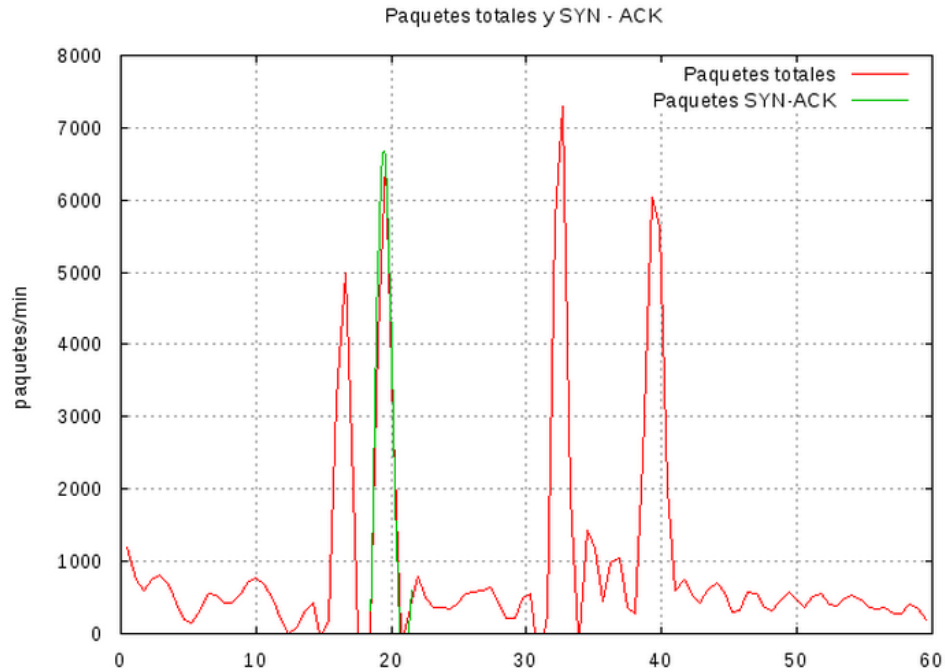


Figura 49: Imagen relacionada con ataque de DoS a otros (TCPstat). 2012.

En la Figura 50 se puede ver que la dirección IP 111.122.53.179, utilizada en el ataque 7, fue la única accediendo la darknet. Esta IP envió paquetes SYN-ACK a la darknet, por lo tanto este gráfico, acompañado del TCPstat que grafica la existencia de paquetes con dicho flag activado, puede ayudar a detectar Denegación de Servicios a servidores fuera de la darknet.



Figura 50: Imagen relacionada con ataque de DoS a otros (column). 2012.

Para detectar escaneos de puertos también se realizaron pruebas, las cuales se muestran a continuación:

- Ataque 8: Desde una determinada dirección IP, se escanearon los puertos conocidos (<1024) de direcciones aleatorias dentro de la darknet.

hping3 -I eth0 -a 111.122.53.179 --scan 1-1024 -S 1.1.1.x --rand-dest

- Ataque 9: Desde direcciones IPs aleatorias, se escanearon los puertos conocidos (<1024) de una dirección particular de la darknet.

hping3 -I eth0 --rand-source --scan 1-1024 -S 1.1.1.195

- Ataque 10: Desde direcciones IPs aleatorias, se escanearon los puertos conocidos (<1024) de direcciones aleatorias de la darknet.

hping3 -I eth0 --rand-source --scan 1-1024 -S 1.1.1.x --rand-dest

Dentro de los ataques de escaneos de puertos realizados, el único que entregó patrones y gráficos interesantes fue el ataque número 9. Analizando el tráfico asociado a los ataques 8 y 10, se notó que hping3 lanzó paquetes de manera muy dispersa (variados puertos y direcciones IP), esto se debe a que tanto la IP de destino como los puertos fueron escogidos aleatoriamente. Se postula que el tiempo durante el cual se lanzaron ambos ataques no fue suficiente para visualizar patrones de un escaneo específico (envió paquetes a un amplio grupo de direcciones y no alcanzó a acceder a varios puertos de una misma IP). Probablemente si se hubiese ejecutado hping3 por más tiempo, se podrían haber detectado dichos patrones, pero el tráfico generado habría sido muchísimo y Maani Charts no podría haberlo graficado. Quizás el uso de otra herramienta hubiera sido útil en este caso.

Algunos ejemplos de patrones detectados tras lanzar los ataques 8, 9 y 10 se describen a continuación:

En la Figura 51 se puede apreciar cómo algunos puertos fueron escaneados en un corto período de tiempo. Los puertos 1, 443, 587, 1010 y 80 fueron accedidos en el mismo minuto una alta cantidad de veces, y en particular los puertos 587, 80 y 1 fueron escaneados de manera similar en cuanto a cantidad. Este comportamiento no es normal, generalmente los puertos accedidos se distribuyen por el tiempo y no se forman líneas verticales como la que se puede apreciar en la figura. Las horas escogidas para la creación de este gráfico fueron aquellas donde los ataques 8, 9 y 10 fueron lanzados.



Figura 51: Imagen relacionada con el escaneo de puertos (time). 2012.

En la Figura 52 se puede apreciar lo ocurrido en un escaneo de puertos (específicamente en los horarios en que se lanzó el ataque número 9). Aquí se ve claramente cómo parte de los puertos conocidos fueron accedidos en la misma proporción, esto es una señal notoria de un escaneo a dichos puertos.

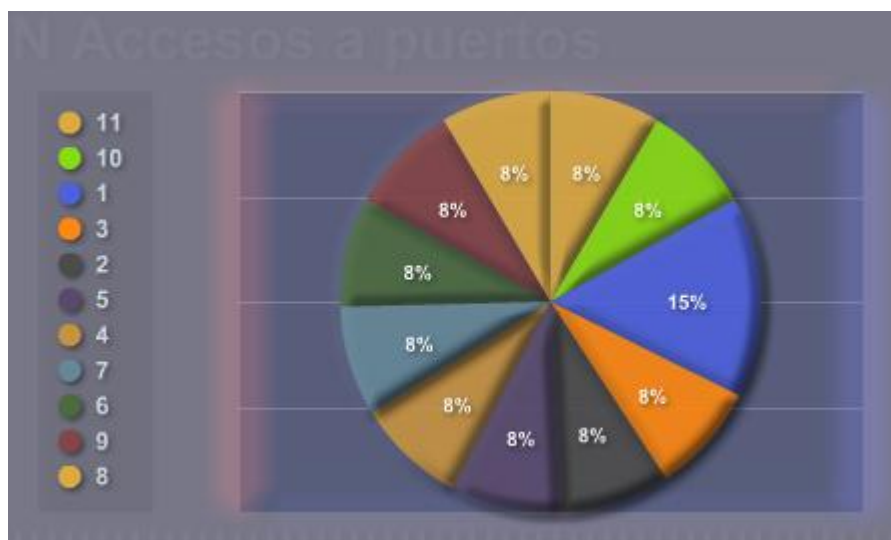


Figura 52: Imagen relacionada con el escaneo de puertos (pie). 2012.

En la Figura 53 se vuelve a apreciar cómo algunos puertos fueron accedidos la misma cantidad de veces (específicamente en los horarios en que se lanzó el ataque número 9). Al ver patrones así se puede afirmar que es altamente probable de que ellos hayan sido escaneados.

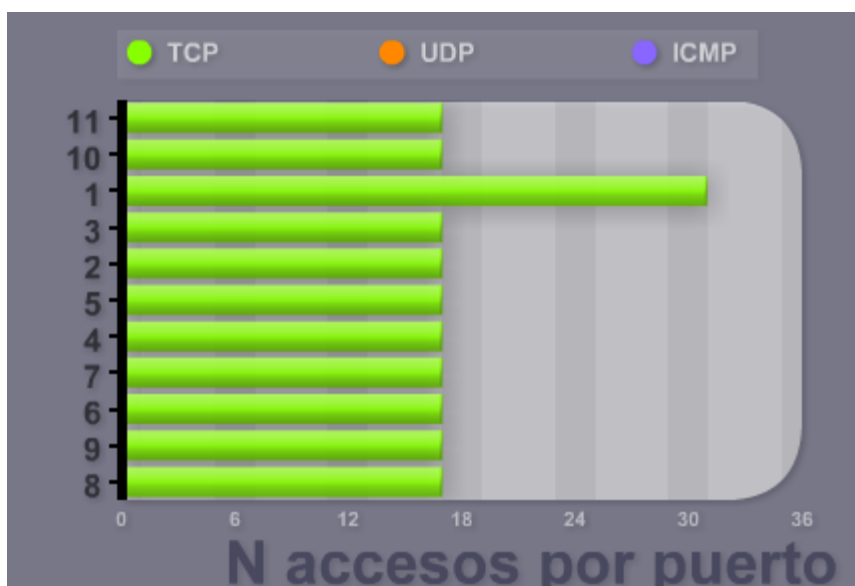


Figura 53: Imagen relacionada con el escaneo de puertos (rect). 2012.

En la Figura 54 se pueden apreciar todos los ataques en conjunto, es decir, escaneos de IPs (líneas paralelas al eje rojo que contiene las IPs de la darknet), Denegación de Servicios (líneas paralelas al eje azul que contiene las IPs atacantes) y escaneo de puertos (líneas paralelas al eje verde que contiene los puertos accedidos).

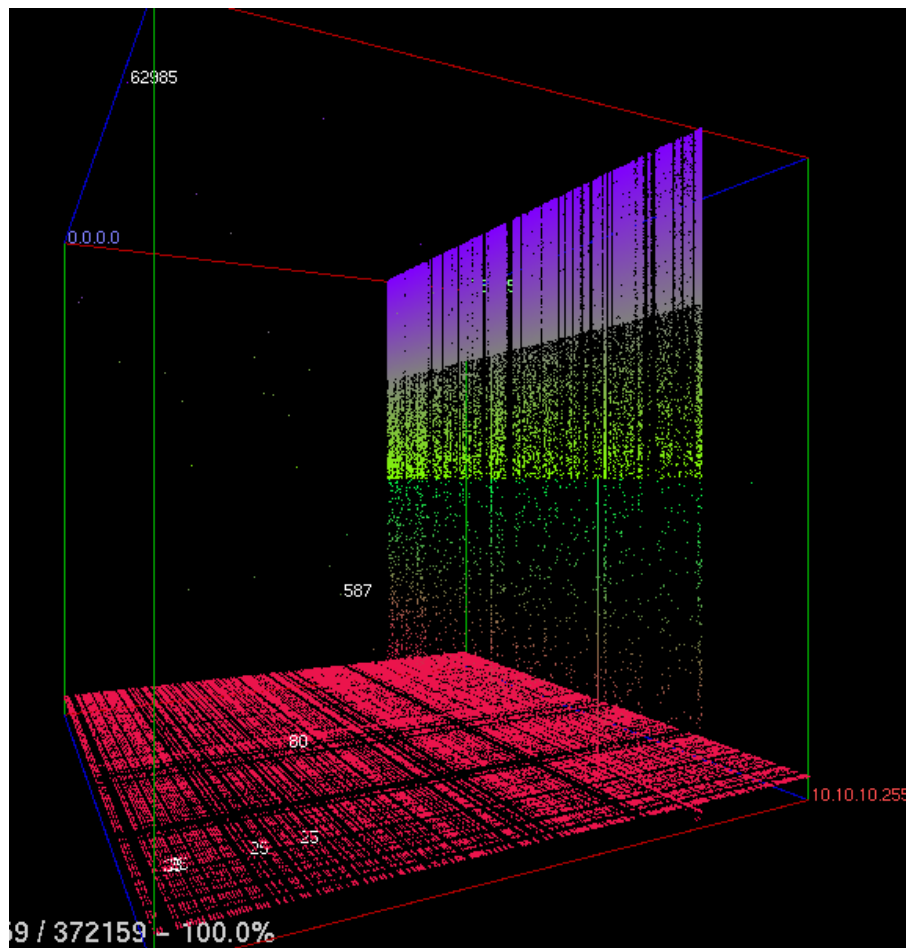


Figura 54: Imagen relacionada con todos los ataques (Moncube). 2012.

En la Figura 54, cada una de las líneas verticales que conforman la cortina de colores, corresponde a un escaneo de muchos puertos a una sola dirección IP (ataque 9) esto según los patrones existentes en el sitio de moncube⁵¹.

Con estas pruebas se puede afirmar que, efectivamente, en caso de existir un ataque a la darknet éste puede ser visualizado utilizando la información entregada por los distintos gráficos.

⁵¹ Patrones de escaneos en Moncube <http://www-moncube.cea.fr/doku.php/en:cube:patterns>.

9. Conclusiones y Aportes

En este trabajo de tesis se pueden destacar variados aspectos relacionados a su aporte científico y rescatar conclusiones, las cuales se listan a continuación:

- Primero se debe recalcar que este es un sistema de monitoreo dedicado 100% a una darknet, en la extensa investigación realizada no se encontró un software de similares características.
- Además los gráficos tienen como prioridad que, a pesar de manejar muchos datos, se desplieguen bien y sean entendibles. Para poder lograr esto se crearon filtros y criterios de búsqueda que facilitaron la comprensión de los datos, pero que no restaron información importante.
- El tiempo de respuesta es rápida considerando la cantidad de datos que se deben procesar y desplegar (beneficio de tener gráficos pre-calculados). En el caso de los gráficos a pedido, se realizaron pruebas con el día que ha registrado más tráfico hasta el momento (14-11-2011 con 494 MegaBytes) y los tiempos de respuesta se detallan en la siguiente sección.
- Estos tiempos podrían ser aún menores dado que el sistema fue diseñado para ser paralelizable y escalable. Si los scripts se lanzaran en distintos computadores y se distribuyeran los procesos (por ejemplo que cada computador genere un XML o un gráfico) la respuesta entregada al usuario sería aún más rápida, sólo sería necesario que en cada equipo se encuentren los archivos contenedores tráfico diario. Además, contando con equipos de mayor capacidad de procesamiento, memoria RAM etc. se espera que los tiempos sean considerablemente menores.
- El sistema, además, fue diseñado para ser extensible sin mucha dificultad, dado que con la integración de otros softwares dedicados a la seguridad (como es el sistema de detección de intrusos, Snort [SNT12]) se podría tener información agregada de gran utilidad (por ejemplo, junto a cada gráfico del sitio web especificar si Snort detecta algún tipo de ataque o vulnerabilidad en el paquete de datos recibido). Por otro lado, sería posible generar gráficos o videos de varias redes darknets en conjunto, para hacerlo sólo bastaría con concatenar los archivos de tráfico diario de cada una de ellas utilizando tcpdump. Si se desearan crear gráficos de distintas darknet de forma separada, también sería posible. Se debería contar con el tráfico de cada una de ellas en directorios separados (con el mismo formato), adaptar los scripts para que no solamente lean tráfico de una carpeta, crear nuevos nombres para los XMLs resultantes y hacer algunos cambios en el sitio web encargado del despliegue para que muestre todos los gráficos generados.
- Además, este sistema se puede utilizar en variantes de redes darknets, como son las Honeynets [JR04] (tal como se mencionó anteriormente, la diferencia radica en que éstas últimas no son silenciosas, sino que generan algunas respuestas a las solicitudes entrantes para intentar crear algún tipo de “comunicación” con el atacante). En este caso se analizaría el tráfico de entrada

solamente y no se considerarían las respuestas dadas por la Honeynet, dado que esta información tiene características distintas a las observadas en una darknet.

- La creación de gráficos pre calculados ayudan al usuario a encontrar información útil y rápidamente, sin tener que solicitar la creación de un gráfico ad-hoc en cada momento, si no que sólo cuando es estrictamente necesario.
- Cabe destacar que la elección del software utilizado (AfterGlow, Moncube, TCPstat, EtherApe) y de los gráficos realizados no es arbitraria, hubo un estudio donde se priorizó la claridad y se seleccionaron los gráficos más auto explicativos, además de seguir las recomendaciones del libro Applied Security Visualization [RM10]. En los 10 gráficos creados mediante XML muestran la información que entrega la darknet de variadas formas (IPs de origen, puertos de destino y origen, protocolos utilizados), estas combinaciones son las más interesantes en lo que a seguridad informática respecta (considerando los datos de entrada).
- Los filtros que se realizaron en los datos (por puertos) fueron importantes, si no se aplicaban, los gráficos eran prácticamente incomprensibles o pesaban demasiado lo que provocaba errores de Flash. Es por esto que mediante muchas pruebas realizadas y soluciones propuestas se llegó a una óptima: filtrar por los puertos más accedidos (para la mayoría de los gráficos) o los más importantes (para los gráficos asociados a direcciones IP).

Los puertos considerados como los más importantes en una red son los siguientes: 25 53 80 443 110

- TCP port 25 - SMTP (Simple Mail Transfer Protocol)
- TCP and UDP port 53 - DNS (Domain Name System)
- TCP ports 80 and 443 - HTTP (Hypertext Transport Protocol) and HTTPS (HTTP over SSL)
- TCP port 110 - POP3 (Post Office Protocol version 3)

De todas maneras los más accedidos en esta darknet son:

- 25 SMTP Simple Mail Transfer Protocol (Protocolo Simple de Transferencia de Correo) (TCP).
- 445 Microsoft-DS compartición de ficheros (UDP) usado por el servicio Microsoft-DS (Active Directory, compartición en Windows por el gusano Sasser [SASS12] y el Troyano Agobot [AGO12]) (TCP).

Tal como se pudo apreciar con anterioridad, se escogen 5 puertos para los gráficos creados por omisión (y también un máximo de 5 puertos para la creación del gráfico de puertos en el tiempo). Este número (cinco) fue escogido arbitrariamente, pero en caso de querer cambiarlo se puede hacer fácilmente dado que el sistema tiene una implementación flexible en este punto.

- Es importante que el sistema se pueda acceder desde cualquier lugar y de manera sencilla, es por esto que se optó por desplegar la información en un sitio web. De esta manera los datos estarán siempre disponibles y se podrán acceder desde cualquier computador (cuyo browser debe tener flash).
- La creación de un gráfico contenedor de muchos datos (demostración de big data) tuvo que ser ideada de manera que sea un gráfico útil de todas maneras (no que solo tenga una finalidad teórica, sino que también práctica). Se ideó un gráfico que muestre los accesos por distintas IPs a la darknet en el tiempo, con la intención de tener una vista generalizada de lo que ocurre y no simplemente diaria o de algunas horas. Este punto es importante ya que da la posibilidad al analista de hacer seguimiento y además demuestra que este trabajo de tesis puede ser funcional en caso de big data.
- El software creado permite ser aplicado a cualquier tipo de red que entregue su información de tráfico de la manera especificada (para no tener problemas de formato). Además este trabajo puede ser aplicado a más una darknet simultáneamente, solo bastaría replicar las carpetas contenedoras de tráfico y diferenciarlas con un indicador. A su vez, los scripts se duplicarían, unos correrían sobre los datos de una red, y los otros sobre los de la otra.
- El manejo de distintos errores o problemas ha llevado que este trabajo se perfeccione cada vez más. Un claro ejemplo de esto es la instalación del servidor virtual X llamado Real VNC [XSERV11] para que el computador del laboratorio continúe creando gráficos a pesar de que la pantalla esté apagada o no se haya iniciado sesión. Moncube, una de las herramientas utilizadas, necesita OpenGL para su funcionamiento [OGL12]. El servidor X instalado no cuenta con él, por lo tanto en caso de que el corte de luz sea prolongado (superior a lo manejado por el UPS) estos gráficos no serán generados automáticamente hasta que alguien, presencialmente, inicie sesión en el computador. Mejorar este inconveniente se propone como trabajo futuro.
- Otro problema que se está manejando relacionado con la disponibilidad del sistema es que, si ocurre algún error en la darknet o en la creación de gráficos, el sistema le informa al usuario a través de su calendario indicándole explícitamente qué días puede ver gráficos (o crearlos), eso incurre en un ahorro de tiempo para el analista, ya que no buscará información en días en que el sistema no se encontró disponible.
- Existe la posibilidad de que un usuario desee crear un gráfico a través del sitio web pero, por ejemplo, por una pérdida de conexión con el servidor, éste no lo logra ver. Es por esto que se implementó una alerta que le informará de dicho error a través de un mensaje (ventana pop up), dándole la posibilidad de seguir esperando o de crear un nuevo gráfico. Estos últimos puntos son ejemplos de cómo se han manejado problemas ajenos al sistema propiamente tal (cortes de luz, lentitud en la red, saturación del hardware, etc).
- La existencia de una sección de gráficos ad-hoc (creados por el usuario) permite al analista poder especificar y observar el comportamiento que se desea estudiar, con el detalle que quiera. Incluso es capaz de ver lo que sucede el día de hoy (el gráfico creado por defecto del día

completo no lo podrá ver ya que se genera en la madrugada del día siguiente, pero sí podrá crear nuevos gráficos).

- Los gráficos creados por los distintos softwares existentes son de gran calidad, muestran de manera creativa e intuitiva información (por primera vez utilizados en una darknet, al menos en publicaciones revisadas). Los videos crean una simulación en tiempo real permitiendo tener mayor claridad, dado que muestra tal como ocurrieron los accesos.
- La existencia de distintos tipos de gráficos, videos y diagramas permite al analista a hacer una combinación de toda la información y tomar decisiones con fuertes fundamentos (acompañados de las alertas definidas por él mismo en el sitio web). En general, en los SW existentes de monitoreo de redes (no específicos para darknet) despliegan pocos tipos de gráficos, en cambio en este trabajo se tiene una amplia gama: 10 tipos de gráficos Flash, diagrama Afterglow, cubo 3D Moncube, 2 tipos de gráficos 2D con Tcpslice y los videos de EtherApe, aparte del sistema de alertas. Esto es valioso porque permite comparar la información y tener más fundamentos para justificar cualquier riesgo detectado.
- Gracias a la sección de detección de ataques existente en el sitio, el usuario es capaz de detectar un determinado ataque en caso de que este haya existido en la red. Además el hecho de que el analista pueda asociar un nombre a cada una de las alertas especificadas (etiquetas) lo ayudan en esta tarea.

Éstos son los aportes principales detectados en este trabajo de tesis. Cada uno de ellos fue un desafío que al ser solucionado otorgó valor al proyecto permitiendo que se perfeccione cada vez más. Claramente hay mucho que mejorar, sobre todo en la interfaz de usuario, manejo de varios gráficos con big data, paralelismo, nuevas formas de visualización de tráfico etc. por lo tanto este trabajo no se encuentra cerrado, sino que abre caminos para futuras investigaciones.

10. Datos estadísticos y futuras líneas de investigación

10. a. Pruebas y datos estadísticos:

Se realizaron pruebas de stress al sistema solicitando la creación de gráficos ad-hoc para el día que registra más accesos (14-11-2011 con un archivo de datos de 434 megabytes) y los tiempos de respuesta (despliegue del gráfico solicitado) considerando el tráfico de todo el día fueron:

- Accesos por puerto (todos los puertos seleccionados): Pie 8 segundos.
- Accesos por protocolo (todos los puertos seleccionados): PieProt 8 segundos, RingProt 11 segundos.
- Accesos por puerto y protocolo (todos los puertos seleccionados): Rad 1 segundo, Rect 9 segundos, Bubble 5 segundos.
- Accesos por protocolo y dirección IP (todos los puertos seleccionados): Column 5 segundos, Area 12 segundos.

Para los gráficos de direcciones IP y Puertos en el tiempo se seleccionó el día 28 de Diciembre de 2011 con 256 MegaBytes de tráfico. El cambio de fecha se debe a que la implementación de las listas (de IPs y puertos), necesaria para la creación de los gráficos, se finalizó a mediados de Diciembre. Los resultados fueron los siguientes:

- Accesos de IP en el tiempo (seleccionando las 3 IPs más demandadas): 132 segundos.
- Accesos de puertos en el tiempo (seleccionando los 5 puertos más accedidos, dejando de lado el primero (puerto 445) ya que tenía 2681603 accesos, lo que estropearía la escala y probablemente generaría errores de Flash): 100 segundos.

Cabe destacar que estos dos últimos gráficos tomaron mucho más tiempo en generarse que los demás. Este resultado es de esperar ya que en éstos se calculan y despliegan muchos más datos que en los demás (un punto por cada minuto del día). Si en un determinado minuto del día existen cero accesos de una determinada IP a la red, es necesario crear un tag en el XML con esta información para que luego Maani Charts pueda interpretarlo. Entonces, por ejemplo, si se están calculando la cantidad de paquetes que llegan a la red desde tres direcciones IP durante 1 hora, si tendrán 180 tags (60 por cada dirección IP, un tag por minuto). Es por esto que los archivos XMLs resultan largos y su lectura no es sencilla (muchos tags con "0 accesos").

Dado lo anterior, se optó por hacer una prueba con otra herramienta gráfica que tome por defecto el valor cero y no sea necesario explicitarlo (solo se especifican la cantidad de solicitudes o paquetes que sean distintas de cero), tomar el tiempo y compararlo con el de Maani Charts. Se eligió el software High

Charts [HC12] para generar el gráfico de IPs en el tiempo del día 28-12-2012 (257 mb). El proceso de map - reduce utilizado en ambas herramientas gráficas fue el mismo ya que de todas maneras hay que calcular cuántos accesos existen por minuto. La diferencia radicaba en el tiempo necesario para preparar el XML para Maani Charts o el Json para High Charts, pero los tiempos no mejoraron, el proceso total tardó 3 minutos en ambos casos.

De todas maneras existen beneficios asociados a la utilización de High Charts, el peso de los archivos resultantes (Json) es menor a los de Maani Charts (11 kb v/s 180 kb), ya que no se generan "tags" por los accesos iguales a cero y por lo tanto el envío desde el computador del laboratorio al servidor web tarda menos tiempo (de todas maneras es casi imperceptible la diferencia). Además la lectura es mucho más sencilla que la de los XMLs de Maani Charts (dado su largo). Por razones de tiempo no se implementaron todos los gráficos con High Charts pero es una alternativa viable, tomando los datos que el map - reduce implementado entrega. Se propone seguir esta línea de investigación como trabajo futuro.

La cantidad de paquetes que llegan a la darknet ha variado con el tiempo, esto se refleja en el peso de los archivos contenedores de tráfico. En un inicio éstos eran relativamente pequeños (6 MegaBytes en promedio por día) pero a medida que han pasado los meses este número ha crecido alcanzando incluso archivos de 434 megabytes como ocurrió el día 14 de Noviembre del año 2011. La Figura 55 muestra el crecimiento de los archivos pcap arrojados por la darknet (valores de promedios mensuales en megabytes).



Figura 55: Peso de archivos de tráfico en el tiempo. Original creado con Google Charts. 2012.

10. b. Futuras líneas de investigación:

Tal como se mencionó anteriormente hay algunos aspectos en este trabajo de tesis que se pueden mejorar, lo cuales se proponen como trabajo futuro:

- Creación de otros gráficos en “Amazon” en caso de big data.
- Mejoramiento de la interfaz del usuario (sitio web): selección más sencilla de la hora, distribución de manera más ordenada de la información, mostrar una línea de tiempo en videos otorgados por EtherApe, mejorar escala de gráficos TCPstat, etc.
- Respecto a las alertas desplegadas en el sitio: mostrarlas de manera gráfica y atractiva (no sólo con texto). Además sería interesante la implementación de un sistema de autenticación para que cada usuario del sitio pueda definir sus propias alertas y éstas no se sobre escriban (tema ya está cubierto para la creación de gráficos, distintos usuarios pueden solicitarlos y éstos no se sobre escribirán dada la existencia de un número random asociado a cada solicitud).
- Manejar de mejor manera el envío de los archivos XML entre el computador del laboratorio y el servidor web para evitar que los datos no lleguen por un error en la red. Una alternativa a esto sería utilizar algún sistema de sincronización entre los computadores.
- Poder buscar más detalle en un gráfico haciendo click o zoom sobre él (Maani Charts debería ser descartado, ya que al hacer click permite la interacción con el gráfico pero no la profundización de éstos). De esta manera el usuario podría indagar en posibles ataques que detecte sin necesidad de crear nuevos gráficos.
- Generar gráficos en torno a los puertos de origen. Este dato no se ha explotado lo suficiente en esta tesis.
- Creación de gráficos y videos a pedido con los cuatros software utilizados en esta tesis (Afterglow, Etherape, Moncube, TCPstat), realizando cortes en los archivos de tráfico con tcpslice (tal como se hizo con los gráficos creados a partir de XMLs).
- Incluir nuevas darknets en el sistema para tener parámetros de comparación (generación de gráficos por separado).
- Graficar utilizando otras herramientas tales como High Charts para ahorrar tiempo en el envío de datos desde el computador del laboratorio hasta el servidor web y simplificar la lectura de éstos, por ejemplo.
- Agregar OpenGL al servidor virtual X instalado (VirtualGL por ejemplo [VIRGL11]) o buscar una solución análoga para que, en caso de un corte de luz prolongado, los gráficos de Moncube se sigan generando automáticamente a pesar de que nadie haya iniciado sesión en el computador del laboratorio.

- Separar los siguientes procesos en distintos servidores: creación de los archivos XML pre-calculados (paralelizarlos) y alertas, uso de las 4 herramientas seleccionadas, solicitudes provenientes del sitio web (gráficos ad - hoc). Esto con el fin de no sobrecargar ningún computador.
- Creación de un parser que procese los XMLs utilizados en la creación de los gráficos y genere unos nuevos en el formato especificado anteriormente (IODEF) para compartir la información con otros CERTS a nivel mundial.
- Crear nuevos gráficos útiles como por ejemplo: gráficos que muestren el número de requests DNS (puerto 53, protocolo UDP), gráficos que den información sobre los flgas activados por paquete (RST, PUSH, URG, ACK, FIN), cantidad de accesos por subred a la darknet (por ejemplo: Accesos TCP desde IPs en Chile en la últimas 24 horas), entre otros.
- Creación de filtros diferentes a los ya realizados (IPs y puertos con más accesos), pero teniendo siempre presente la importancia de que éstos no deben dejar de lado información que podría ser relevante en términos de seguridad.

Estas ideas pueden guiar a futuros investigadores a perfeccionar el trabajo realizado para esta tesis.

11. Referencias

[AEC212] Amazon Elastic Cloud Computing (Amazon EC2) <http://aws.amazon.com/es/ec2/>. Febrero 2012

[AEMR12] Amazon Elastic MapReduce (Amazon EMR). <http://aws.amazon.com/elasticmapreduce/>. Febrero 2012.

[AF10] Raffael Marty. Software AfterGlow. Software disponible en <http://afterglow.sourceforge.net/>. Agosto 2010.

[AGO12] VSantivirus. Información sobre el virus Agobot y su comportamiento disponible en <http://www.vsantivirus.com/agobot-oab.htm>. Enero 2012.

[AMTU12] Amazon Documentation: Elastic MapReduce Tutorial. Disponible en <http://docs.amazonwebservices.com/ElasticMapReduce/latest/GettingStartedGuide/Welcome.html?r=7350>. Febrero 2012.

[AS10] Ask Student. Disponible en <http://www.askstudent.com/security/sinkholes-in-network-security-5-easy-steps-to-deploy-a-darknet/>. Abril 2010.

[AS312] Amazon Simple Storage Service (Amazon S3). <http://aws.amazon.com/s3/>. Febrero 2012.

[AW05] Martin Arlitt, Carey Williamson. "An Analysis of TCP Reset Behaviour on the Internet". Publicado en ACM SIGCOMM Computer Communication Review. Volumen 35 edición 1 ACM Press. 2005.

[AWS12] Amazon Web Services (AWS). Amazon Elastic MapReduce (Amazon EMR). Febrero 2012.

[BCJMS06] Michael Bailey, Evan Cooke, Farnam Jahanian, Andrew Myrick, Sushant Sinha. "Practical Darknet Measurement". Publicado en Information Sciences and Systems, 2006 40th Annual Conference. Páginas 1496 -1501. University of Michigan. 2006.

[BCJPR05] Michael Bailey, Evan Cooke, Farnam Jahanian, Niels Provos, Karl Rosaen, David Watson. "Data Reduction for the Scalable Automated Analysis of Distributed Darknet Traffic". Technical Report. University of Michigan. 2005. Disponible en <http://www.eecs.umich.edu/fjgroup/pubs/filtering-imc05.pdf>. Revisado en Julio 2012.

[BD03] Y.N. Bardachev, A.A. Didyk. "Revealing of informative multifractal properties of network traffic for anomalies detection". Technical Report. Kherson National Technical University, Ucrania. 2003. Disponible para su descarga en <http://opengmdh.org/raw-attachment/wiki/ICIM08-51/Revealing%20of%20Informative%20Multifractal%20Properties%20of%20Network%20Traffic%20for%20Anomalies%20Detection.PDF>. Revisado en Julio 2012.

[CA110] Publicaciones que utilizan datos de CAIDA. Disponibles en <http://www.caida.org/data/publications/bydataset/index.xml>. Mayo 2010.

[CA210] Investigaciones sobre el Network Telescope. Disponibles en <http://www.caida.org/research/security/telescope/>. Mayo 2010.

- [CA310] Fuentes de datos de CAIDA asociados al Network Telescope. Disponibles en http://www.caida.org/data/passive/network_telescope.xml. Mayo 2010.
- [CB07] Sébastien Chainay, Karima Boudaoud. "A Model for Automatic generation of behaviourbased worm signatures". Technical Report. University of Nice Sophia Antipolis. 2007. Disponible en <http://spiderman-2.laas.fr/METROSEC/MonAM2006.pdf>. Revisado en Julio 2012.
- [CJM07] Evan Cooke, Farnam Jahanian, Danny McPherson. "The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets". Technical Report. Electrical Engineering and Computer Science Department. 2007. Disponible en <http://www.eecs.umich.edu/fjgroup/pubs/botnets-sruti05.pdf>. Revisado en Julio 2012.
- [CONV10] Comando convert Detalles del comando disponibles en el sitio: http://linux.about.com/od/commands/l/blcmdl1_convert.htm. Enero 2010.
- [CS89] Cliff Stoll. "The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage". Editorial New York. 1989.
- [CY08] Team Cymru . "Who is looking for your SCADA infrastructure?". Technical Report. 2008. Disponible en <http://www.team-cymru.com/ReadingRoom/Whitepapers/2009/scada.pdf>. Revisado en Julio 2012.
- [CY10] Team CYMRU (Darknet Project). Disponible en <http://www.team-cymru.org/Services/darknets.html>. Abril 2011.
- [DG04] Jeffrey Dean, Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. OSDI '04 Proceedings. Páginas 137-150. Google, Inc. 2004.
- [EA10] Juan Toledo. Software EtherApe. Disponible online en <http://etherape.sourceforge.net/>. Agosto 2010.
- [F12] Flash Player, Adobe. Software disponible en <http://www.adobe.com/es/products/flashplayer.html>. Febrero 2012.
- [FJ10] Farman Jahanian. "Networking and Security Research Group". Disponible en <http://www.eecs.umich.edu/fjgroup/>. University of Michigan. Abril 2010.
- [G03] Darren Grabowski, Manager NTT America Global IP Network Security & Abuse Team. "Global Network Pandemic – The Silent Threat". Technical Report. 2003. Disponible en http://www.us.ntt.net/downloads/papers/Grabowski_Global_Network_Pandemic.pdf. Revisado en Julio 2012.
- [HC12] Highsoft Solutions. Herramienta gráfica High Charts. Software disponible en <http://www.highcharts.com/>. Febrero 2012.
- [HB11] Phillip M. Hallam-Baker, Brian Behlendorf. The common log file format. Disponible en http://www.w3.org/Daemon/User/Config/Logging.html#common_logfile_format. Agosto 2011.

- [I11] Alfon. Seguridad y redes: “Software Inetvis”. Disponible en <http://seguridadyredes.wordpress.com/2010/11/04/inetvis-representacion-tridimensional-de-capturas-de-red-a-partir-de-archivos-pcap-parte-i/>. Junio 2011.
- [IMCA11] CAIDA Researchers. Imagen de Ataque de Denegación de Servicios. Disponible en http://www.caida.org/data/passive/network_telescope.xml. Julio 2011.
- [IODEF12] Cover Pages. Incident Object Description and Exchange Format IODEF. Detalles y studios disponibles en <http://xml.coverpages.org/iodef.html>. Febrero 2012.
- [JB05] Jaime Blasco, Aitsec. “An approach to malware collection log visualization”. Technical Report. 2005. Disponible en <http://dl.packetstormsecurity.net/papers/evaluation/An-approach-to-malware-collection-log-visualization.pdf>. Revisado en Julio 2012.
- [JDV09] Gideon Juve, Ewa Deelman, Karan Vahi, Gaurang Mehta, Bruce Berriman, Benjamin P. Berman, Phil Maechling. Scientific Workflow Applications on Amazon EC2. Publicado en el Workshop on Cloud-based Services and Applications y en la Quinta Conferencia Internacional IEEE en Ciencias (e-Science 2009). Páginas 59 – 66. Oxford UK. Diciembre 2009.
- [JR04] Jeremiah K. Jones, Gordon W. Romney. “Honeynets: an educational resource for IT security”. Publicado en “CITC5 '04 Proceedings of the 5th conference on Information technology education”. ACM Press. 2004.
- [L06] Tom Liston. “Follow the bouncing malware VII: Afterglow”. 2006. Disponible en <http://isc.sans.edu/diary.html?date=2005-07-20>. Revisado en Julio 2012.
- [M10] Carlos M. Martinez, “Presentación Aprendiendo del enemigo”. Anteldata CSIRT ANTEL . Disponible en <http://www.csirt-antel.com.uy/main/public/aprendiendo-del-enemigo-01.pdf>. Abril 2010.
- [MA11] Maani Charts , “XML/SWF Charts”. Software disponible en http://www.maani.us/xml_charts/. Mayo 2011.
- [MAPI12] Documentación de Maani Charts y ejemplos, “XML/SWF Charts”. Documentación disponible en http://www.maani.us/xml_charts/index.php?menu=Reference.
- [MG08] Marcos Martínez García. “Construcción de Laboratorios virtuales para la administración de sistemas y servidores”. Universidad Politécnica de Valencia. España. Memoria de título. 2008. Disponible en http://riunet.upv.es/bitstream/handle/10251/9135/PFC_Completo_PDF.pdf?sequence=1. Revisado en Julio 2012.
- [MHL02] B. Mukherjee, L. Heberlein, K. Levitt. “Network intrusion detection”. IEEE Network, páginas 26-41. University of California, Davis. 2002.
- [MI11] Microsoft. “Microsoft Security Intelligence Report, Volume 10”. Disponible en http://www.microsoft.com/security/sir/keyfindings/default.aspx#!section_4_1. Julio 2011.
- [MO10] David Moore CAIDA. “Presentación Network Telescopes, David”. Disponible en <http://www.caida.org/publications/presentations/2003/dimacs0309/>. Abril 2010.

[MON10] The Security Team of the French Atomic Energy and Alternative Energies Commission. "Software Moncube". Software disponible en <http://www-moncube.cea.fr/doku.php/en:cube:cube>. Agosto 2010.

[MPSW03] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuard Standford, Nicholas Weaver. "Inside the Slammer Worm". En Security & Privacy, IEEE. Páginas 33 – 39. University of California. 2003.

[MSB02] David Moore, Colleen Shannon, Jeffery Brown. "Code-Red: a case study on the spread and victims of an Internet worm". En proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement. ACM Press. 2002.

[MSBV06] David Moore, Colleen Shannon, Doug Brown, Geoffrey M. Voelker, Stefan Savage. "Inferring Internet Denial-of-Service Activity". Publicado en ACM Transactions on Computer Systems. Volumen 24 edición 2. ACM Press. 2006.

[MVS01] David Moore, Geoffrey M. Voelker, Stefan Savage. "Inferring Internet Denial-of-Service Activity". Publicado en SSYM'01 Proceedings of the 10th conference on USENIX Security Symposium . USENIX Association Berkeley, CA, USA. 2001.

[NEAFT10] Uso de Neato para AfterGlow. Alfon. Seguridad y redes: "Software Inetvis". Disponible en <http://seguridadyredes.wordpress.com/2010/06/28/visualizacian-grafica-de-ficheros-pcap-con-afterglow>. Julio 2010.

[OGL12] OpenGL Software, Khronos. Disponible en <http://www.opengl.org/>. Febrero 2012.

[PA11] Lacnic. Proyecto Amparo. Descripción disponible en <http://www.proyectoamparo.net/>. Junio 2011.

[PYBPP04] Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson, Larry Peterson. "Characteristics of Internet Background Radiation". Publicado en "Proceedings of the 4th ACM SIGCOMM conference on Internet measurement". ACM Press. 2004.

[RM10] Raffael Marty. "Applied Security Visualization". Páginas 3, 5, 7, 13, 23, 78, 87, 93, 101, 105, 115, 451, 483 y 491. Editorial Addison-Wesley. 2010.

[S11] Meshuggeneh. Software Shoki. Disponible en <http://shoki.sourceforge.net/>. Junio 2011.

[SASS12] Hispasec sistemas, Gusano Sasser. Información disponible en <http://unaaldia.hispasec.com/2004/05/gusano-sasser-infecta-automaticamente.html>. Enero 2012.

[SCMD12] S3 Tools: command line S3 client. Descripción del software y pasos para instalación disponible en <http://s3tools.org/s3cmd>. Febrero 2012.

[SED10] Linux / Unix command: sed Detalles del comando disponibles en el sitio: http://linux.about.com/od/commands/l/blcmdl1_sed.htm. Enero 2010.

[SMS01] Dug Song, Rob Malan, Robert Stone. "A Snapshot of Global Internet Worm Activity". ArborNetworks. Technical Report. 2001. Disponible en

http://www.lasr.cs.ucla.edu/classes/239_3.winter03/papers/snapshot_worm_activity.pdf. Revisado en Julio 2012.

[SNT12] Sourcefire. Sistema de detección de intrusos Snort. Software disponible en <http://www.snort.org/>. Febrero 2012.

[SPL12] Comando Split en Linux, descripción disponible en el sitio: <http://unixhelp.ed.ac.uk/CGI/man-cgi?split>. Febrero 2012.

[SY11] Symantec. "Symantec Internet Security Threat Report" Volumen 16. Disponible en https://www4.symantec.com/mktginfo/downloads/21182883_GA_REPORT_ISTR_Main-Report_04-11_HI-RES.pdf. Julio 2011.

[TCPDU10] Software Tcpcdump y Libpcap. Disponible en <http://www.tcpdump.org/>. Enero 2010.

[TCPRW10] Comando tcprewrite Detalles del comando disponibles en el sitio: <http://tcpreplay.synfin.net/tcprewrite.html>. Enero 2010.

[TCPS10] Linux / Unix command: tcpslice. Detalles del comando disponibles en el sitio: http://linux.about.com/library/cmd/blcmdl8_tcpslice.htm. Enero 2010.

[TDT03] Quang-Anh Tran, Haixin Duan, Xing Li. Tsinghua University . "One-class Support Vector Machine for Anomaly Network Traffic Detection". Journal of Software. 2003.

[TPP09] Roberto Tamassia, Bernardo Palazzi, Charalampos Papamanthou . "Graph Drawing for Security Visualization". Publicado en el libro Graph Drawing. 2009.

[TS10] Paul Herman. Software TCPstat. Software disponible en <http://www.frenchfries.net/paul/tcpstat/>. Agosto 2010.

[TSHA10] Comando tshark. Detalles del comando disponibles en <http://www.wireshark.org/docs/man-pages/tshark.html>. Enero 2010.

[U10] Dirección General de Servicios de Cómputo Académico-UNAM . "Revista digital UNAM". Disponible en <http://www.revista.unam.mx/vol.9/num4/art21/int21.htm>. Mayo 2010.

[V09] Craig Valli, Edith Cowan University. "An Analysis of Malfeasant Activity Directed at a VoIP HoneyPot". Conference Proceeding. 2009.

[VIRGL11] VirtualGL. Software disponible en <http://www.virtualgl.org/>. Octubre 2011.

[W04] Colin Ware. "Information Visualization: Perception for Design". Editorial Morgan Kaufmann. Abril 2004.

[WHO11] TeamCymru. Comando Whois del TeamCymru: Detalles disponibles en <http://www.team-cymru.org/Services/ip-to-asn.html#whois>. Mayo 2011.

[XSERV11] Real VNC, Servidor Virtual X. Información del software disponible en <http://www.realvnc.com/products/enterprise/4.1/man/vncserver.html>. Octubre 2011.

11. Anexos

A. Código fuente del script *afterglowMin.sh*

```
#!/bin/bash
ayer=$(date --date='1 day ago')
echo "ayer era: " $ayer
year=$(date --date='1 day ago' +%Y)
month=$(date --date='1 day ago' +%m)
day=$(date --date='1 day ago' +%d)

archivo="salida-"$year-"-$month"-"$day
salida=/home/rfaccilo/Desktop/capturas/$archivo
cp $salida /home/rfaccilo/Desktop/tesis/afterglow/src/perl/graph/$archivo
for H in "00" "01" "02" "03" "04" "05" "06" "07" "08" "09" "10" "11" "12" "13"
"14" "15" "16" "17" "18" "19" "20" "21" "22" "23";

do
/home/rfaccilo/Desktop/tesis/afterglow/src/perl/graph/afterglow.sh $archivo
${H}h00m00s ${H}h10m00s
/home/rfaccilo/Desktop/tesis/afterglow/src/perl/graph/afterglow.sh $archivo
${H}h00m00s ${H}h20m00s
/home/rfaccilo/Desktop/tesis/afterglow/src/perl/graph/afterglow.sh $archivo
${H}h00m00s ${H}h30m00s
/home/rfaccilo/Desktop/tesis/afterglow/src/perl/graph/afterglow.sh $archivo
${H}h30m00s ${H}h40m00s
/home/rfaccilo/Desktop/tesis/afterglow/src/perl/graph/afterglow.sh $archivo
${H}h30m00s ${H}h50m00s
/home/rfaccilo/Desktop/tesis/afterglow/src/perl/graph/afterglow.sh $archivo
${H}h30m00s ${H}h59m59s

done
cd /home/rfaccilo/Desktop/tesis/afterglow/src/perl/graph
rm $archivo
```

B. Código fuente del script *afterglow.sh*

```
#!/bin/bash
year=$(date --date='1 day ago' +%Y)
month=$(date --date='1 day ago' +%m)
day=$(date --date='1 day ago' +%d)

dia=$year-"-$month"-"$day
cd ~/Desktop/tesis/afterglow/src/perl/graph/
rm -rf graph_afterglow*
rm -rf tcpslice*
rm -rf grafica*

/usr/sbin/tcpslice -w tcpslice_ $1_ $2_ $3.pcap $2 $3 $1
```

```

/usr/bin/tcprewrite --pnat=X.X.X.X/24:1.1.1.0/24 --
infile=tcpslice_$1_$2_$3.pcap --outfile=salidaNUEVA.pcap
/usr/bin/tshark -r salidaNUEVA.pcap -T fields -E separator=, -e ip.dst -e
ip.src | perl afterglow.pl -c color.properties -t > grafica_$1_$2_$3.dot
/usr/bin/neato -Tgif -o graph_afterglow_$1_$2_$3.GIF ./grafica_$1_$2_$3.dot
mkdir -p ~/Desktop/AFTERGLOW/$dia/TCPSLICE/
mv tcpslice_$1_$2_$3.pcap
~/Desktop/AFTERGLOW/$dia/TCPSLICE/tcpslice_$1_$2_$3.pcap
mkdir -p ~/Desktop/AFTERGLOW/$dia/DOT/
mv grafica_$1_$2_$3.dot ~/Desktop/AFTERGLOW/$dia/DOT/grafica_$1_$2_$3.dot
mkdir -p ~/Desktop/AFTERGLOW/$dia/GRAHP/
mv graph_afterglow_$1_$2_$3.gif
~/Desktop/AFTERGLOW/$dia/GRAHP/graph_afterglow_$1_$2_$3.gif
convert
/home/rfaccilo/Desktop/AFTERGLOW/$dia/GRAHP/graph_afterglow_$1_$2_$3.GIF
/home/rfaccilo/Desktop/AFTERGLOW/$dia/GRAHP/graph_afterglow_$1_$2_$3.PNG
rm /home/rfaccilo/Desktop/AFTERGLOW/$dia/GRAHP/graph_afterglow_$1_$2_$3.GIF

```

C. Código fuente del script *moncubeMin.sh*

```

#!/bin/bash
ayer=$(date --date='1 day ago')
echo "ayer era: " $ayer
year=$(date --date='1 day ago' +%Y)
month=$(date --date='1 day ago' +%m)
day=$(date --date='1 day ago' +%d)
archivo="salida-"$year-"$month-"$day
salida=$archivo
/home/rfaccilo/Desktop/tesis/moncube-1.2.3/bin/moncube.sh
~/Desktop/capturas/$salida

```

D. Código fuente de script *moncube.sh*

```

#!/bin/bash
year=$(date --date='1 day ago' +%Y)
month=$(date --date='1 day ago' +%m)
day=$(date --date='1 day ago' +%d)
dia=$year-"$month-"$day

waitNow(){
sleep 15s
xdotool key w
xdotool key r
sleep 3s
xdotool key w
sleep 7s
xdotool key w
xdotool key r
sleep 2s
killall parsecube

```

```

}

cd /home/rfaccilo
rm salida*
rm -rf cubeshot-*
rm moncube_A_$dia.bmp
rm moncube_B_$dia.bmp
rm moncube_C_$dia.bmp
/usr/bin/tcprewrite --pnat=xxx.xx.xx.x/x:10.10.10.0/24 --infile=$1 --
outfile=salidaMON
/usr/sbin/tcpdump -r salidaMON -nt > salida

waitNow |./parsecube

mkdir -p ~/Desktop/MONCUBE/$dia/
mv cubeshot-001.bmp ~/Desktop/MONCUBE/$dia/moncube_A_$dia.bmp
mv cubeshot-002.bmp ~/Desktop/MONCUBE/$dia/moncube_B_$dia.bmp
mv cubeshot-003.bmp ~/Desktop/MONCUBE/$dia/moncube_C_$dia.bmp
mv salida ~/Desktop/MONCUBE/$dia

```

E. Código fuente del script *TCPstats_protocolMin.sh*.

```

#!/bin/bash
ayer=$(date --date='1 day ago')
echo "ayer era: " $ayer
year=$(date --date='1 day ago' +%Y)
month=$(date --date='1 day ago' +%m)
day=$(date --date='1 day ago' +%d)
archivo="salida-"$year-"$month-"$day

salida=/home/rfaccilo/Desktop/capturas/$archivo
cp $salida /home/rfaccilo/Desktop/tesis/tcpstat-1.5/$archivo
cd /home/rfaccilo/Desktop/tesis/tcpstat-1.5
for H in "00" "01" "02" "03" "04" "05" "06" "07" "08" "09" "10" "11" "12" "13"
"14" "15" "16" "17" "18" "19" "20" "21" "22" "23";

do /usr/sbin/tcpslice -w ${H}h00m00s_${H}h59m59s.pcap ${H}h00m00s ${H}h59m59s
$archivo
/home/rfaccilo/Desktop/tesis/tcpstat-1.5/TCPstats_protocol.sh
${H}h00m00s_${H}h59m59s.pcap
done

/home/rfaccilo/Desktop/tesis/tcpstat-1.5/TCPstats_protocol.sh $archivo

```

F. Código fuente del script *TCPstats_protocol.sh*.

```

#!/bin/bash
year=$(date --date='1 day ago' +%Y)
month=$(date --date='1 day ago' +%m)
day=$(date --date='1 day ago' +%d)

```



```

dia=$year-"-$month"-"$day
cd /home/rfaccilo/Desktop/tesis/tcpstat-1.5/
rm -rf graph_protocol_*
rm -rf arp.data
rm -rf icmp.data
rm -rf tcp.data
rm -rf udp.data

/usr/bin/tcpstat -r $1 -o "%R\t%A\n" 60 > arp.data
/usr/bin/tcpstat -r $1 -o "%R\t%C\n" 60 > icmp.data
/usr/bin/tcpstat -r $1 -o "%R\t%T\n" 60 > tcp.data
/usr/bin/tcpstat -r $1 -o "%R\t%U\n" 60 > udp.data
/usr/bin/gnuplot
/home/rfaccilo/Desktop/tesis/tcpstat-1.5/gnuplot.script >
graph_protocol_$1.png

mkdir -p ~/Desktop/TCP_PROTOCOL/$dia/PNG/
mv graph_protocol_* ~/Desktop/TCP_PROTOCOL/$dia/PNG/
mkdir -p ~/Desktop/TCP_PROTOCOL/$dia/DATA/
mv arp.data ~/Desktop/TCP_PROTOCOL/$dia/DATA
mv icmp.data ~/Desktop/TCP_PROTOCOL/$dia/DATA
mv tcp.data ~/Desktop/TCP_PROTOCOL/$dia/DATA
mv udp.data ~/Desktop/TCP_PROTOCOL/$dia/DATA
mkdir -p ~/Desktop/TCP_PROTOCOL/$dia/PCAP/
mv *.pcap ~/Desktop/TCP_PROTOCOL/$dia/PCAP/
cp salida* ~/Desktop/TCP_PROTOCOL/$dia/PCAP/

```

G. Código fuente del script *gnuplot.script*.

```

set term png small #FFFFFF
set data style lines
set grid
set yrange [ 0 : ]
set title "Paquetes por protocolo"
set xlabel "minutos"
set ylabel "paquetes/min"
plot "arp.data" using ($1/60):2 smooth csplines title "ARP" \
, "icmp.data" using ($1/60):2 smooth csplines title "ICMP" \
, "tcp.data" using ($1/60):2 smooth csplines title "TCP" \
, "udp.data" using ($1/60):2 smooth csplines title "UDP"

```

H. Código fuente del script *TCPstats_SYNACKMin.sh*

```

#!/bin/bash
ayer=$(date --date='1 day ago')
echo "ayer era: " $ayer
year=$(date --date='1 day ago' +%Y)
month=$(date --date='1 day ago' +%m)
day=$(date --date='1 day ago' +%d)
archivo="salida-"$year-"-$month"-"$day

```

```

salida=/home/rfaccilo/Desktop/capturas/$archivo
cp $salida /home/rfaccilo/Desktop/tesis/tcpstat-1.5/$archivo
cd /home/rfaccilo/Desktop/tesis/tcpstat-1.5
for H in "00" "01" "02" "03" "04" "05" "06" "07" "08" "09" "10" "11" "12" "13"
"14" "15" "16" "17" "18" "19" "20" "21" "22" "23";

do /usr/sbin/tcpslice -w ${H}h00m00s_${H}h59m59s.pcap ${H}h00m00s ${H}h59m59s
$archivo
/home/rfaccilo/Desktop/tesis/tcpstat-1.5/TCPstats_SYNACK.sh
${H}h00m00s_${H}h59m59s.pcap
done

/home/rfaccilo/Desktop/tesis/tcpstat-1.5/TCPstats_SYNACK.sh $archivo

```

I. Código fuente del script *TCPstats_SYNACKMin.sh*

```

#!/bin/bash
year=$(date --date='1 day ago' +%Y)
month=$(date --date='1 day ago' +%m)
day=$(date --date='1 day ago' +%d)

dia=$year-"-$month"-"$day
cd /home/rfaccilo/Desktop/tesis/tcpstat-1.5/

rm -rf graph_synack_*
rm -rf total.data
rm -rf synack.data
/usr/bin/tcpstat -r $1 -o "%R\t%n\n" 60 > total.data
/usr/sbin/tcpdump -r $1 'tcp[13] = 18' -w synack.pcap
/usr/bin/tcpstat -r synack.pcap -o "%R\t%n\n" 60 > synack.data
/usr/bin/gnuplot /home/rfaccilo/Desktop/tesis/tcpstat-1.5/gnuplotACK.script >
graph_synack_${1}.png

/usr/bin/tcpstat -r archivo_nuevo.pcap -o "%R\t%T\n" 60 > tcp.data
/usr/bin/gnuplot /home/rfaccilo/Desktop/tesis/tcpstat-1.5/gnuplotACK.script >
graph_synack_${1}.png

mkdir -p ~/Desktop/TCP_SYNACK/$dia/PNG/
mv graph_synack* ~/Desktop/TCP_SYNACK/$dia/PNG
mkdir -p ~/Desktop/TCP_SYNACK/$dia/DATA/
mv total.data ~/Desktop/TCP_SYNACK/$dia/DATA
mv synack.data ~/Desktop/TCP_SYNACK/$dia/DATA
mkdir -p ~/Desktop/TCP_SYNACK/$dia/PCAP
mv *.pcap ~/Desktop/TCP_SYNACK/$dia/PCAP
cp salida* ~/Desktop/TCP_SYNACK/$dia/PCAP

```

J. Código fuente del script *gnuplotACK.script*

```

set term png small #FFFFFF
set data style lines
set grid

```

```

set yrange [ 0 : ]
set title "Paquetes totales y SYN - ACK"
set xlabel "minutos"
set ylabel "paquetes/min"
plot "total.data" using ($1/60):2 smooth csplines title "Paquetes totales" \
, "synack.data" using ($1/60):2 smooth csplines title "Paquetes SYN-ACK"

```

K. Código fuente del script *ether.sh*

```

#!/bin/bash
year=$(date +%Y)
month=$(date +%m)
day=$(date +%d)
archivo="salida-"$year-"-$month"-"$day

salida=/home/rfaccilo/Desktop/capturas/$archivo

cp $salida /home/rfaccilo/Desktop/tesis/etherape-0.9.9/$archivo
hora_inicio=$(date --date='15 minutes ago' +%H)
minuto_inicio=$(date --date='15 minutes ago' +%M)
hora_fin=$(date --date='5 minutes ago' +%H)
minuto_fin=$(date --date='5 minutes ago' +%M)
/usr/sbin/tcplice -w
${hora_inicio}h${minuto_inicio}m00s_${hora_fin}h${minuto_fin}m00s.pcap
${hora_inicio}h${minuto_inicio}m00s_${hora_fin}h${minuto_fin}m00s
/home/rfaccilo/Desktop/tesis/etherape-0.9.9/$archivo

mv ${hora_inicio}h${minuto_inicio}m00s_${hora_fin}h${minuto_fin}m00s.pcap
/home/rfaccilo/Desktop/tesis/etherape-0.9.9
/home/rfaccilo/Desktop/tesis/etherape-0.9.9/ethMin.sh
${hora_inicio}h${minuto_inicio}m00s_${hora_fin}h${minuto_fin}m00s.pcap

```

L. Código fuente del script *etherMin.sh*

```

#!/bin/bash
year=$(date +%Y)
month=$(date +%m)
day=$(date +%d)
dia=$year-"-$month"-"$day
CUENTA=`ps -efa | grep '[/]etherape' > /dev/null`

waitNow(){
xmin=$(xwininfo -name EtherApe |grep "Absolute upper-left X: " | sed -e "s/
Absolute upper-left X: //" )
ymin=$(xwininfo -name EtherApe |grep "Absolute upper-left Y: " | sed -e "s/
Absolute upper-left Y: //" )
ffmpeg -f x11grab -r 25 -s 530x328 -i :1+${(xmin+90)},{(ymin+90)} -t 600
/home/rfaccilo/Desktop/tesis/etherape-0.9.9/video.mpeg
killall -9 ffmpeg
killall -9 etherape
}

```

```

mv $1 /home/rfaccilo/Desktop/tesis/etherape-0.9.9
cd /home/rfaccilo/Desktop/tesis/etherape-0.9.9
rm video*
rm salidaDESTINO*
/usr/bin/tcprewrite --pnat= X.X.X.X/X:10.10.10.1 --infile=$1 --
outfile=salidaDESTINO

etherape -r salidaDESTINO_${1} &
sleep 5
waitNow

mkdir -p /home/rfaccilo/Desktop/ETHERAPEminuto/$dia/VIDEO/
mv video* /home/rfaccilo/Desktop/ETHERAPEminuto/$dia/VIDEO/video_${1}.mpeg
mkdir -p /home/rfaccilo/Desktop/ETHERAPEminuto/$dia/PCAP/
mv salida* /home/rfaccilo/Desktop/ETHERAPEminuto/$dia/PCAP
mv *.pcap /home/rfaccilo/Desktop/ETHERAPEminuto/$dia/PCAP
rm -r /home/rfaccilo/Desktop/ETHERAPEminuto/$dia/PCAP

```

M. Código del script *tiraListas.sh*

```

#!/bin/bash
year=$(date --date='1 day ago' +%Y)
month=$(date --date='1 day ago' +%m)
day=$(date --date='1 day ago' +%d)
archivo="salida-"$year-"-"$month-"-"$day

salida=/home/rfaccilo/Desktop/capturas/$archivo
cd /home/rfaccilo/Desktop
./generadorListas.sh 00h00m00s 23h59m59s $archivo "all"

cp /home/rfaccilo/Desktop/capturas/IP.txt
/home/rfaccilo/Desktop/listas/IP${archivo}.txt
mv /home/rfaccilo/Desktop/capturas/Puertos.txt
/home/rfaccilo/Desktop/listas/Puertos${archivo}.txt
mv /home/rfaccilo/Desktop/capturas/IP.txt
/home/rfaccilo/Desktop/listas/IP2${archivo}.txt

echo -e "begin\ncountrycode" > temp;
cat /home/rfaccilo/Desktop/listas/IP${archivo}.txt >>temp;mv temp
/home/rfaccilo/Desktop/listas/IP${archivo}.txt
echo 'end' >> /home/rfaccilo/Desktop/listas/IP${archivo}.txt

netcat whois.cymru.com 43 < /home/rfaccilo/Desktop/listas/IP${archivo}.txt >
/home/rfaccilo/Desktop/listas/IPdatos${archivo}.txt
sed '1d' /home/rfaccilo/Desktop/listas/IPdatos${archivo}.txt > temp
sed '/NA /d' temp > temp2
sed '$d' temp2 > tempBueno

cp tempBueno /home/rfaccilo/Desktop/listas/paises${archivo}.txt

sed '1001,999999d' tempBueno > temp3

```

```

mv temp3 /home/rfaccilo/Desktop/listas/IPdatos${archivo}.txt
sed '/None/d' /home/rfaccilo/Desktop/listas/Puertos${archivo}.txt > temp4
sed '1001,999999d' temp4 > temp5

mv temp5 /home/rfaccilo/Desktop/listas/Puertosdatos${archivo}.txt
rm /home/rfaccilo/Desktop/listas/IP${archivo}.txt
rm /home/rfaccilo/Desktop/listas/Puertos${archivo}.txt
rm temp*
cd /home/rfaccilo/Desktop/capturas
rm cortado*
rm fileLimpio.txt

cd /home/rfaccilo/Desktop
./graficosFlash.sh

cd /home/rfaccilo/Desktop/listas
./paises.sh

```

N. Código del script *generadorListas.sh*

```

#!/bin/bash
HOME_DIR=/home/rfaccilo/Desktop
cd $HOME_DIR/capturas
/usr/sbin/tcplice -w cortadoLi_${1}_${2}_${3} $1 $2 $3
/usr/sbin/tcpdump -nr cortadoLi_${1}_${2}_${3} &> unoLi_${1}_${2}_${3}.txt
sed '1d' unoLi_${1}_${2}_${3}.txt > tempLi_${1}_${2}_${3}.txt
cat tempLi_${1}_${2}_${3}.txt | sed 's/\./ /g' | sed 's/:/ /' | sed 's/:/ /' |
sed 's/:/ /' | sed 's/:/ /' > cortadoLi_${1}_${2}_${3}.txt

inicio=$1
fin=$2
archivo=$3
tipo=$4
shift 3

python2.6 $HOME_DIR/Parser/src/listaIP.py
$HOME_DIR/capturas/cortadoLi_${inicio}_${fin}_${archivo}.txt $@ >&
$HOME_DIR/errauto/errores_listaIP.log
sed '/ICMP /d' cortadoLi_${inicio}_${fin}_${archivo}.txt > temp
rm cortadoLi_${inicio}_${fin}_${archivo}.txt
mv temp cortadoLi_${inicio}_${fin}_${archivo}.txt

python2.6 $HOME_DIR/Parser/src/listaPuertos.py
$HOME_DIR/capturas/cortadoLi_${inicio}_${fin}_${archivo}.txt $@ >&
$HOME_DIR/errauto/errores_listaPuertos.log

```

O. Código del script *graficosFlash.sh*

```
#!/bin/bash

year=$(date --date='1 day ago' +%Y)
month=$(date --date='1 day ago' +%m)
day=$(date --date='1 day ago' +%d)

ayer=$year-"-$month"-"$day

year=$(date --date='2 day ago' +%Y)
month=$(date --date='2 day ago' +%m)
day=$(date --date='2 day ago' +%d)

ayer2=$year-"-$month"-"$day

year=$(date --date='3 day ago' +%Y)
month=$(date --date='3 day ago' +%m)
day=$(date --date='3 day ago' +%d)

ayer3=$year-"-$month"-"$day

year=$(date --date='4 day ago' +%Y)
month=$(date --date='4 day ago' +%m)
day=$(date --date='4 day ago' +%d)

ayer4=$year-"-$month"-"$day

year=$(date --date='5 day ago' +%Y)
month=$(date --date='5 day ago' +%m)
day=$(date --date='5 day ago' +%d)

ayer5=$year-"-$month"-"$day

year=$(date --date='6 day ago' +%Y)
month=$(date --date='6 day ago' +%m)
day=$(date --date='6 day ago' +%d)

ayer6=$year-"-$month"-"$day

year=$(date --date='7 day ago' +%Y)
month=$(date --date='7 day ago' +%m)
day=$(date --date='7 day ago' +%d)
ayer7=$year-"-$month"-"$day

cd /home/rfaccilo/Desktop/listas/

fileIP2=IP2salida-${ayer}.txt
fileIP=IPdatossalida-${ayer}.txt
filePuertos=Puertosdatossalida-${ayer}.txt
filePuertosUDP=PuertosUDPdatossalida-${ayer}.txt
filePuertosTCP=PuertosTCPdatossalida-${ayer}.txt

fileAtaIP=IPAtasalida-${ayer}.txt
#IPS
```

```

ips=$(egrep -m 3 -o '\b[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\b'
$fileIP)

ipsata=$(egrep -m 3 -o '\b[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\b'
$fileAtaIP)

if [ "$ips" == "" ]; then
    echo "entre al if"
    fileIP=$fileIP2
    ips=$(egrep -m 3 -o '\b[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-
9]{1,3}\b' $fileIP)
    #rm IPdatossalida-${ayer}.txt
    mv IP2salida-${ayer}.txt IPdatossalida-${ayer}.txt
    echo "sali del if"
fi

echo "archivo: " $fileIP
echo "-----ips " $ips

#PUERTOS NORMALES
linea1=$(sed -n '1 p' $filePuertos)
pospipe=$(awk -v a="$linea1" -v b="|" 'BEGIN{print index(a,b)}')
p1=${linea1:0:$pospipe-1}
echo "p1 " $p1

accesos1=${linea1:$pospipe}

linea2=$(sed -n '2 p' $filePuertos)
pospipe=$(awk -v a="$linea2" -v b="|" 'BEGIN{print index(a,b)}')
p2=${linea2:0:$pospipe-1}
echo "p2 " $p2

accesos2=${linea2:$pospipe}
linea3=$(sed -n '3 p' $filePuertos)
pospipe=$(awk -v a="$linea3" -v b="|" 'BEGIN{print index(a,b)}')
p3=${linea3:0:$pospipe-1}
echo "p3 " $p3

accesos3=${linea3:$pospipe}

linea4=$(sed -n '4 p' $filePuertos)
pospipe=$(awk -v a="$linea4" -v b="|" 'BEGIN{print index(a,b)}')
p4=${linea4:0:$pospipe-1}
echo "p4 " $p4

accesos4=${linea4:$pospipe}

linea5=$(sed -n '5 p' $filePuertos)
pospipe=$(awk -v a="$linea5" -v b="|" 'BEGIN{print index(a,b)}')
p5=${linea5:0:$pospipe-1}
echo "p5 " $p5

accesos5=${linea5:$pospipe}

#PUERTOS udp
linea1=$(sed -n '1 p' $filePuertosUDP)
pospipe=$(awk -v a="$linea1" -v b="|" 'BEGIN{print index(a,b)}')

```

```

up1=${linea1:0:$pospipe-1}

linea2=$(sed -n '2 p' $filePuertosUDP)
pospipe=$(awk -v a="$linea2" -v b="|" 'BEGIN{print index(a,b)}')
up2=${linea2:0:$pospipe-1}

linea3=$(sed -n '3 p' $filePuertosUDP)
pospipe=$(awk -v a="$linea3" -v b="|" 'BEGIN{print index(a,b)}')
up3=${linea3:0:$pospipe-1}

linea4=$(sed -n '4 p' $filePuertosUDP)
pospipe=$(awk -v a="$linea4" -v b="|" 'BEGIN{print index(a,b)}')
up4=${linea4:0:$pospipe-1}

linea5=$(sed -n '5 p' $filePuertosUDP)
pospipe=$(awk -v a="$linea5" -v b="|" 'BEGIN{print index(a,b)}')
up5=${linea5:0:$pospipe-1}

linea6=$(sed -n '6 p' $filePuertosUDP)
pospipe=$(awk -v a="$linea6" -v b="|" 'BEGIN{print index(a,b)}')
up6=${linea6:0:$pospipe-1}

linea7=$(sed -n '7 p' $filePuertosUDP)
pospipe=$(awk -v a="$linea7" -v b="|" 'BEGIN{print index(a,b)}')
up7=${linea7:0:$pospipe-1}

linea8=$(sed -n '8 p' $filePuertosUDP)
pospipe=$(awk -v a="$linea8" -v b="|" 'BEGIN{print index(a,b)}')
up8=${linea8:0:$pospipe-1}

linea9=$(sed -n '9 p' $filePuertosUDP)
pospipe=$(awk -v a="$linea9" -v b="|" 'BEGIN{print index(a,b)}')
up9=${linea9:0:$pospipe-1}

linea10=$(sed -n '10 p' $filePuertosUDP)
pospipe=$(awk -v a="$linea10" -v b="|" 'BEGIN{print index(a,b)}')
up10=${linea10:0:$pospipe-1}

#PUERTOS tcp
linea1=$(sed -n '1 p' $filePuertosTCP)
pospipe=$(awk -v a="$linea1" -v b="|" 'BEGIN{print index(a,b)}')
tp1=${linea1:0:$pospipe-1}

linea2=$(sed -n '2 p' $filePuertosTCP)
pospipe=$(awk -v a="$linea2" -v b="|" 'BEGIN{print index(a,b)}')
tp2=${linea2:0:$pospipe-1}

linea3=$(sed -n '3 p' $filePuertosTCP)
pospipe=$(awk -v a="$linea3" -v b="|" 'BEGIN{print index(a,b)}')
tp3=${linea3:0:$pospipe-1}

linea4=$(sed -n '4 p' $filePuertosTCP)
pospipe=$(awk -v a="$linea4" -v b="|" 'BEGIN{print index(a,b)}')
tp4=${linea4:0:$pospipe-1}

linea5=$(sed -n '5 p' $filePuertosTCP)
pospipe=$(awk -v a="$linea5" -v b="|" 'BEGIN{print index(a,b)}')

```



```

tp5=${linea5:0:$pospipe-1}

linea6=$(sed -n '6 p' $filePuertosTCP)
pospipe=$(awk -v a="$linea6" -v b="|" 'BEGIN{print index(a,b)}')
tp6=${linea6:0:$pospipe-1}

linea7=$(sed -n '7 p' $filePuertosTCP)
pospipe=$(awk -v a="$linea7" -v b="|" 'BEGIN{print index(a,b)}')
tp7=${linea7:0:$pospipe-1}

linea8=$(sed -n '8 p' $filePuertosTCP)
pospipe=$(awk -v a="$linea8" -v b="|" 'BEGIN{print index(a,b)}')
tp8=${linea8:0:$pospipe-1}

linea9=$(sed -n '9 p' $filePuertosTCP)
pospipe=$(awk -v a="$linea9" -v b="|" 'BEGIN{print index(a,b)}')
tp9=${linea9:0:$pospipe-1}

linea10=$(sed -n '10 p' $filePuertosTCP)
pospipe=$(awk -v a="$linea10" -v b="|" 'BEGIN{print index(a,b)}')
tp10=${linea10:0:$pospipe-1}

echo "archivo" $ayer
cd /home/rfaccilo/Desktop
#21 23 25 53 80 443 110 column area time

echo "El puerto " $p1 " fue accesado " $accesos1 " veces." >>
/home/rfaccilo/Desktop/graficosFlash/accesos-${ayer}.txt
echo "El puerto " $p2 " fue accesado " $accesos2 " veces." >>
/home/rfaccilo/Desktop/graficosFlash/accesos-${ayer}.txt
echo "El puerto " $p3 " fue accesado " $accesos3 " veces." >>
/home/rfaccilo/Desktop/graficosFlash/accesos-${ayer}.txt
echo "El puerto " $p4 " fue accesado " $accesos4 " veces." >>
/home/rfaccilo/Desktop/graficosFlash/accesos-${ayer}.txt
echo "El puerto " $p5 " fue accesado " $accesos5 " veces." >>
/home/rfaccilo/Desktop/graficosFlash/accesos-${ayer}.txt

./generador.sh 00h00m00s 23h59m59s salida-${ayer} pie $p2 $p3 $p4 $p5
./generadorRing.sh 00h00m00s 23h59m59s salida-${ayer} salida-${ayer2} salida-
${ayer3} salida-${ayer4} salida-${ayer5} salida-${ayer6} salida-${ayer7} $p2
$p3 $p4 $p5
./generador.sh 00h00m00s 23h59m59s salida-${ayer} pieProt $p2 $p3 $p4 $p5
./generador.sh 00h00m00s 23h59m59s salida-${ayer} rad $p2 $p3 $p4 $p5
./generador.sh 00h00m00s 23h59m59s salida-${ayer} bubble 21 23 25 53 80 443
110
./generador.sh 00h00m00s 23h59m59s salida-${ayer} rect $p2 $p3 $p4 $p5
./generador.sh 00h00m00s 23h59m59s salida-${ayer} column 21 23 25 53 80 443
110
./generador.sh 00h00m00s 23h59m59s salida-${ayer} area 21 23 25 53 80 443 110
./generadorTime.sh 00h00m00s 23h59m59s salida-${ayer} time $p2 $p3 $p4 $p5 .
./generadorTime.sh 00h00m00s 23h59m59s salida-${ayer} timeIP $sips . .
./generadorTime.sh 00h00m00s 23h59m59s salida-${ayer} timeAtaIP $sipsata . .
./generador.sh 00h00m00s 23h59m59s salida-${ayer} columnAta 21 23 25 53 80 443
110
./generador.sh 00h00m00s 23h59m59s salida-${ayer} areaAta 21 23 25 53 80 443
110

```

```

echo "UDP: " $sup1 $sup2 $sup3 $sup4 $sup5 $sup6 $sup7 $sup8 $sup9 $sup10
echo "TCP: " $tp1 $tp2 $tp3 $tp4 $tp5 $tp6 $tp7 $tp8 $tp9 $tp10

./generador.sh 00h00m00s 23h59m59s salida-${ayer} rectUDP $sup1 $sup2 $sup3 $sup4
$sup5 $sup6 $sup7 $sup8 $sup9 $sup10
./generador.sh 00h00m00s 23h59m59s salida-${ayer} rectTCP $tp1 $tp2 $tp3 $tp4
$tp5 $tp6 $tp7 $tp8 $tp9 $tp10

mv /home/rfaccilo/Desktop/graficosFlash/*.xml
/home/rfaccilo/Desktop/intermedia/
mkdir /home/rfaccilo/Desktop/graficosFlash/${ayer}
mv /home/rfaccilo/Desktop/intermedia/*
/home/rfaccilo/Desktop/graficosFlash/${ayer}
mv /home/rfaccilo/Desktop/graficosFlash/accesos-${ayer}.txt
/home/rfaccilo/Desktop/graficosFlash/${ayer}/accesos-${ayer}.txt

cd /home/rfaccilo/Desktop/listas/
rm IPAtasalida-${ayer}.txt

```

P. Código del método *printListaIP*

```

def printListaIP(a):
    b=all(a)
    salida=[]
    for (key, val) in b:
        salida.append(str(key)+"      "+str(val)+"\n")
    return salida

```

Q. Código del método *printListaIP*

```

def printListaPuertos(a):
    b=all(a)
    salida=[]
    for (key, val) in b:
        salida.append(str(key)+"|"+str(val)+"\n")
    return salida

```

R. Código del script *generador.sh*

```

#!/bin/bash
HOME_DIR=/home/rfaccilo/Desktop/
cd $HOME_DIR/capturas
/usr/sbin/tcplice -w cortado_${1}_${2}_${3} $1 $2 $3 &>
$HOME_DIR/errauto/errores_tcplice_${4}.log
/usr/sbin/tcpdump -ntr cortado_${1}_${2}_${3} &> uno_${1}_${2}_${3}.txt
sed '1d' uno_${1}_${2}_${3}.txt > temp_${1}_${2}_${3}.txt
cat temp_${1}_${2}_${3}.txt | sed 's/\. /g' | sed 's/:/ /' >
cortado_${1}_${2}_${3}.txt

```

```

inicio=$1
fin=$2
archivo=$3
tipo=$4
shift 4
case $tipo in

"column" )
    python2.6 $HOME_DIR/Parser/src/column.py
$HOME_DIR/capturas/cortado_${inicio}_${fin}_${archivo}.txt $@ >&
$HOME_DIR/errauto/errores_column.log
    ;;

"area" )
    python2.6 $HOME_DIR/Parser/src/area.py
$HOME_DIR/capturas/cortado_${inicio}_${fin}_${archivo}.txt $@ >&
$HOME_DIR/errauto/errores_area.log
    ;;

"pie" )
    python2.6 $HOME_DIR/Parser/src/pie.py
$HOME_DIR/capturas/cortado_${inicio}_${fin}_${archivo}.txt $@ >&
$HOME_DIR/errauto/errores_pie.log
    ;;

"pieProt" )
    python2.6 $HOME_DIR/Parser/src/pieProt.py
$HOME_DIR/capturas/cortado_${inicio}_${fin}_${archivo}.txt $@ >&
$HOME_DIR/errauto/errores_pieProt.log
    ;;

"rad" )
    python2.6 $HOME_DIR/Parser/src/rad.py
$HOME_DIR/capturas/cortado_${inicio}_${fin}_${archivo}.txt $@ >&
$HOME_DIR/errauto/errores_rad.log
    ;;

"rect" )
    python2.6 $HOME_DIR/Parser/src/rect.py
$HOME_DIR/capturas/cortado_${inicio}_${fin}_${archivo}.txt $@ >&
$HOME_DIR/errauto/errores_rect.log
    ;;

"bubble" )
    python2.6 $HOME_DIR/Parser/src/bubble.py
$HOME_DIR/capturas/cortado_${inicio}_${fin}_${archivo}.txt $@ >&
$HOME_DIR/errauto/errores_bubble.log
    ;;

esac

```

```
cd $HOME_DIR/capturas
rm cortado*
rm file*
rm uno*
rm temp*
```

S. Código del script *generadorTime.sh*

```
#!/bin/bash
HOME_DIR=/home/rfaccilo/Desktop
cd $HOME_DIR/capturas
/usr/sbin/tcplice -w cortado_${1}_${2}_${3} $1 $2 $3 &>
$HOME_DIR/errores/errores_tcplice_${4}.log
/usr/sbin/tcpdump -nr cortado_${1}_${2}_${3} &> uno_${1}_${2}_${3}.txt
sed '1d' uno_${1}_${2}_${3}.txt > temp_${1}_${2}_${3}.txt
cat temp_${1}_${2}_${3}.txt | sed 's/\. /g' | sed 's/:/ /' | sed 's/:/ /' |
sed 's/:/ /' > cortado_${1}_${2}_${3}.txt
inicio=$1
fin=$2
archivo=$3
tipo=$4
shift 4
case $tipo in
"time" )
    python2.6 $HOME_DIR/Parser/src/time.py
    $HOME_DIR/capturas/cortado_${inicio}_${fin}_${archivo}.txt $@ >&
    $HOME_DIR/errauto/errores_time.log
    ;;
"timeIP" )
    python2.6 $HOME_DIR/Parser/src/timeIP.py
    $HOME_DIR/capturas/cortado_${inicio}_${fin}_${archivo}.txt $@ >&
    $HOME_DIR/errauto/errores_timeIP.log
    ;;
esac
rm cortado*
cd $HOME_DIR/capturas
rm cortado*
rm file*
rm uno*
rm temp*
```

T. Código del script *generadorRing.sh*

```
#!/bin/bash
HOME_DIR=/home/rfaccilo/Desktop/
cd $HOME_DIR/capturas
/usr/sbin/tcplice -w cortado_${1}_${2}_${3} $1 $2 $3 &>
$HOME_DIR/errores/errores_tcplice_ring1.log
/usr/sbin/tcpdump -ntr cortado_${1}_${2}_${3} &> uno_${1}_${2}_${3}.txt
sed '1d' uno_${1}_${2}_${3}.txt > temp_${1}_${2}_${3}.txt
cat temp_${1}_${2}_${3}.txt | sed 's/\. /g' | sed 's/:/ /' >
cortado_${1}_${2}_${3}.txt

/usr/sbin/tcplice -w cortado_${1}_${2}_${4} $1 $2 $4 &>
$HOME_DIR/errores/errores_tcplice_ring2.log
/usr/sbin/tcpdump -ntr cortado_${1}_${2}_${4} &> uno_${1}_${2}_${4}.txt
sed '1d' uno_${1}_${2}_${4}.txt > temp_${1}_${2}_${4}.txt
cat temp_${1}_${2}_${4}.txt | sed 's/\. /g' | sed 's/:/ /' >
cortado_${1}_${2}_${4}.txt

/usr/sbin/tcplice -w cortado_${1}_${2}_${5} $1 $2 $5 &>
$HOME_DIR/errores/errores_tcplice_ring3.log
/usr/sbin/tcpdump -ntr cortado_${1}_${2}_${5} &> uno_${1}_${2}_${5}.txt
sed '1d' uno_${1}_${2}_${5}.txt > temp_${1}_${2}_${5}.txt
cat temp_${1}_${2}_${5}.txt | sed 's/\. /g' | sed 's/:/ /' >
cortado_${1}_${2}_${5}.txt

/usr/sbin/tcplice -w cortado_${1}_${2}_${6} $1 $2 $6 &>
$HOME_DIR/errores/errores_tcplice_ring4.log
/usr/sbin/tcpdump -ntr cortado_${1}_${2}_${6} &> uno_${1}_${2}_${6}.txt
sed '1d' uno_${1}_${2}_${6}.txt > temp_${1}_${2}_${6}.txt
cat temp_${1}_${2}_${6}.txt | sed 's/\. /g' | sed 's/:/ /' >
cortado_${1}_${2}_${6}.txt

/usr/sbin/tcplice -w cortado_${1}_${2}_${7} $1 $2 $7 &>
$HOME_DIR/errores/errores_tcplice_ring5.log
/usr/sbin/tcpdump -ntr cortado_${1}_${2}_${7} &> uno_${1}_${2}_${7}.txt
sed '1d' uno_${1}_${2}_${7}.txt > temp_${1}_${2}_${7}.txt
cat temp_${1}_${2}_${7}.txt | sed 's/\. /g' | sed 's/:/ /' >
cortado_${1}_${2}_${7}.txt

/usr/sbin/tcplice -w cortado_${1}_${2}_${8} $1 $2 $8 &>
$HOME_DIR/errores/errores_tcplice_ring6.log
/usr/sbin/tcpdump -ntr cortado_${1}_${2}_${8} &> uno_${1}_${2}_${8}.txt
sed '1d' uno_${1}_${2}_${8}.txt > temp_${1}_${2}_${8}.txt
cat temp_${1}_${2}_${8}.txt | sed 's/\. /g' | sed 's/:/ /' >
cortado_${1}_${2}_${8}.txt

/usr/sbin/tcplice -w cortado_${1}_${2}_${9} $1 $2 $9 &>
$HOME_DIR/errores/errores_tcplice_ring7.log
```

```

/usr/sbin/tcpdump -ntr cortado_${1}_${2}_${9} &> uno_${1}_${2}_${9}.txt
sed '1d' uno_${1}_${2}_${9}.txt > temp_${1}_${2}_${9}.txt
cat temp_${1}_${2}_${9}.txt | sed 's/\./ /g' | sed 's/:/ /' >
cortado_${1}_${2}_${9}.txt

inicio=$1
fin=$2
a1=$3
a2=$4
a3=$5
a4=$6
a5=$7
a6=$8
a7=$9

shift 9

python2.6 $HOME_DIR/Parser/src/ringProto.py
$HOME_DIR/capturas/cortado_${inicio}_${fin}_${a1}.txt
$HOME_DIR/capturas/cortado_${inicio}_${fin}_${a2}.txt
$HOME_DIR/capturas/cortado_${inicio}_${fin}_${a3}.txt
$HOME_DIR/capturas/cortado_${inicio}_${fin}_${a4}.txt
$HOME_DIR/capturas/cortado_${inicio}_${fin}_${a5}.txt
$HOME_DIR/capturas/cortado_${inicio}_${fin}_${a6}.txt
$HOME_DIR/capturas/cortado_${inicio}_${fin}_${a7}.txt $@ >&
$HOME_DIR/errauto/errores_ring.log

cd $HOME_DIR/capturas
rm cortado_*
rm file*
rm uno*

#cd $HOME_DIR
#rm *.xml

```

U. Tabla de llamadas python

Llamada	Código
column.py	<pre> from Main import * limpia(sys.argv[1], "fileLimpio.txt") column() </pre>

columnAta.py	<pre> from Main import * limpia(sys.argv[1],"fileLimpio.txt") columnAta() </pre>
area.py	<pre> from Main import * limpia(sys.argv[1],"fileLimpio.txt") Area() </pre>
areaAta.py	<pre> from Main import * limpia(sys.argv[1],"fileLimpio.txt") AreaAta() </pre>
pie.py	<pre> from Main import * limpia(sys.argv[1],"fileLimpio.txt") pie() </pre>
pieProt.py	<pre> from Main import * limpia(sys.argv[1],"fileLimpio.txt") pieProt() </pre>
rad.py	<pre> from Main import * limpia(sys.argv[1],"fileLimpio.txt") Rad() </pre>
rect.py	<pre> from Main import * limpia(sys.argv[1],"fileLimpio.txt") Rect() </pre>
rectUDP.py	<pre> from Main import * limpiaUDP(sys.argv[1],"fileLimpio.txt") RectUDP() </pre>
rectTCP.py	<pre> from Main import * limpiaTCP(sys.argv[1],"fileLimpio.txt") </pre>

	RectTCP()
bubble.py	<pre> from Main import * limpia(sys.argv[1], "fileLimpio.txt") bubble() </pre>
time.py	<pre> from Main import * limpiatime(sys.argv[1], "fileLimpio.txt") time() </pre>
timeIP.py	<pre> from Main import * limpiaIP(sys.argv[1], "fileLimpio.txt") timeIP() </pre>
timeAtaIP.py	<pre> from Main import * limpiaAtaIP(sys.argv[1], "fileLimpio.txt") timeAtaIP() </pre>
ringProto.py	<pre> from MainRing import * limpia(sys.argv[1], "fileLimpio1.txt") limpia(sys.argv[2], "fileLimpio2.txt") limpia(sys.argv[3], "fileLimpio3.txt") limpia(sys.argv[4], "fileLimpio4.txt") limpia(sys.argv[5], "fileLimpio5.txt") limpia(sys.argv[6], "fileLimpio6.txt") limpia(sys.argv[7], "fileLimpio7.txt") ringProto() </pre>

V. Código del método *limpia*

```

def limpia(File, FileClean):

    params = sys.argv[2:]

```



```

try:
    E = open(File, "r")
    S = open(FileClean, "w")

    for line in E:
        data = re.split(" ", line)

        if(data[5].isdigit() and (data[13] == "UDP," or
data[13] == "tcp")):
            if(data[11].isdigit()):

                for p in params:
                    if(p == "known"):
                        if(int(data[11]) <= 1024):
                            S.writelines(line)

                    if(p == "all"):
                        S.writelines(line)

                    if(p == str(data[11])):
                        S.writelines(line)

except:
    print "Unexpected error:", sys.exc_info()[0]
    raise
    E.close()
    S.close()
    sys.exit(1)
E.close()
S.close()

```

W. Código del método limpiaUDP

```

def limpiaUDP(File,FileClean):

    params = sys.argv[2:]

    try:

        E = open(File, "r")

        S = open(FileClean, "w")

        for line in E:

            data = re.split(" ", line)

```

```

        if(data[5].isdigit() and data[13] == "UDP,"):
            if(data[11].isdigit()):
                for p in params:
                    if(p == "known"):
                        if(int(data[11]) <= 1024):
                            S.writelines(line)
                    if(p == "all"):
                        S.writelines(line)
                    if(p == str(data[11])):
                        S.writelines(line)

except:
    print "Unexpected error:", sys.exc_info()[0]
    raise
    E.close()
    S.close()
    sys.exit(1)

E.close()
S.close()

```

X. Código del método limpiaTCP

```

def limpiaTCP(File,FileClean):
    params = sys.argv[2:]
    try:
        E = open(File, "r")
        S = open(FileClean, "w")
        for line in E:
            data = re.split(" ", line)
            if(data[5].isdigit() and data[13] == "tcp"):
                if(data[11].isdigit()):

```

```

        for p in params:
            if(p == "known"):
                if(int(data[11]) <= 1024):
                    S.writelines(line)
            if(p == "all"):
                S.writelines(line)
            if(p == str(data[11])):
                S.writelines(line)

except:
    print "Unexpected error:", sys.exc_info()[0]
    raise
    E.close()
    S.close()
    sys.exit(1)

E.close()
S.close()

```

Y. Código del método *limpiatime*

```

def limpiatime(File,FileClean):

    params = sys.argv[2:]
    try:
        E = open(File, "r")
        S = open(FileClean, "w")
        for line in E:
            data = re.split(" ", line)

            if data[9].isdigit() and data[15].isdigit():
                for p in params:
                    if(p=="known"):
                        if(int(data[15]) < 1024):
                            S.writelines(line)
                    if(p=="all"):
                        S.writelines(line)
                    if(p == str(data[15])):
                        S.writelines(line)

    except:

```

```

        print "Unexpected error:", sys.exc_info()[0]
        raise
    E.close()
    S.close()
    sys.exit(1)
E.close()
S.close()

```

Z. Código del método *limpiaIP*

```

def limpiaIP(File,FileClean):
    params = sys.argv[2:]
    try:
        E = open(File, "r")
        S = open(FileClean, "w")
        for line in E:
            data = re.split(" ", line)

            if data[9].isdigit() and data[5].isdigit() and data[6].isdigit() and
            data[7].isdigit() and data[8].isdigit():
                ip = data[5] + "." + data[6] + "." + data[7] + "." + data[8]
                for ip_param in params:
                    if(ip_param == ip):
                        S.writelines(line)

    except:
        print "Unexpected error:", sys.exc_info()[0]
        raise
    E.close()
    S.close()
    sys.exit(1)
E.close()
S.close()

```

AA. Código del método *limpiaAtaIP*

```

def limpiaAtaIP(File,FileClean):

    params = sys.argv[2:]

    try:

        E = open(File, "r")

        S = open(FileClean, "w")

        #print "Escribiendo datos :)"

```

```

        #print 'Me meti a limpiaIP2'

        for line in E:

            data = re.split(" ", line)

            if data[9].isdigit() and data[11].isdigit() and
data[12].isdigit() and data[13].isdigit() and data[14].isdigit():

                ip = data[11] + "." + data[12] + "." +
data[13] + "." + data[14]

                for ip_param in params:

                    if(ip_param == ip):

                        S.writelines(line)

except:

    print "Unexpected error:", sys.exc_info()[0]

    raise

    E.close()

    S.close()

    sys.exit(1)

E.close()

S.close()

```

BB. Ejemplos de XMLs por función asociada a tipo de gráfico

Gráfico	XML representativo
column()	<pre> ... <row> <string>TCP</string> <number tooltip='216.105.40.52: 22 (TCP) '>22</number> </row> <row> <string>UDP</string> <number tooltip='216.105.40.52: 0 (UDP) '>0</number> <number tooltip='92.240.68.152: 0 (UDP) '>0</number> </pre>

	<pre> </row> <row> <string>ICMP</string> <number tooltip='216.105.40.52: 0 (ICMP) '>0</number> </row> ... </pre> <p>En este XML se separan las IPs atacantes por protocolo destacando la cantidad de accesos de cada una de ellas (22, 0, 0, 0 respectivamente).</p>
columnAta()	<pre> ... <row> <string>TCP</string> <number tooltip='1.1.1.52: 22 (TCP) '>22</number> </row> <row> <string>UDP</string> <number tooltip='1.1.1.52: 0 (UDP) '>0</number> <number tooltip='1.1.1.152: 0 (UDP) '>0</number> </row> <row> <string>ICMP</string> <number tooltip='1.1.1.52: 0 (ICMP) '>0</number> </row> ... </pre> <p>En este XML se separan las IPs atacadas por protocolo destacando la cantidad de accesos a cada una de ellas (22, 0, 0, 0 respectivamente).</p>
pie()	<pre> ... <row> <null/> <string>443</string> <string>63938</string> <string>53</string> <string>3389</string> </row> <row> <tring></string> <number bevel='data '>114</number> </pre>

	<pre> <number bevel='data'>119</number> <number bevel='data'>203</number> <number bevel='data'>272</number> </row> ... </pre> <p>El bloque (row) superior corresponde a los puertos y el inferior a los accesos por puerto (en orden).</p>
<p>pieProt()</p>	<pre> ... <row> <null/> <string>'TCP'</string> <string>'UDP'</string> </row> <row> <string></string> <number bevel='data'>415</number> <number bevel='data'>293</number> </row> ... </pre> <p>El bloque (row) superior corresponde a los protocolos y el inferior a los accesos por protocolo (en orden).</p>
<p>Rad()</p>	<pre> ... <row> <null/> <string>443</string> <string>63938</string> </row> <row> <string>TCP</string> <number>114</number> <number>0</number> </row> <row> <string>UDP</string> <number>0</number> <number>119</number> </row> </pre>

	<pre> <row> <string>ICMP</string> <number>0</number> <number>0</number> </row> ... </pre> <p>En el primer bloque (row) aparecen los puertos accedidos, luego en los siguientes, se describen la cantidad de accesos por protocolo a cada uno de los puertos del inicio en orden.</p>
<p>Area()</p>	<pre> ... <row> <string>TCP</string> <number tooltip='216.105.40.52: 22 (TCP) '>22</number> </row> <row> <string>UDP</string> <number tooltip='216.105.40.52: 0 (UDP) '>0</number> <number tooltip='92.240.68.152: 0 (UDP) '>0</number> </row> <row> <string>ICMP</string> <number tooltip='216.105.40.52: 0 (ICMP) '>0</number> </row> ... </pre> <p>Se separan las IPs de origen por protocolo destacando la cantidad de accesos de cada una de ellas (22, 0, 0, 0 respectivamente).</p>
<p>AreaAta()</p>	<pre> ... <row> <string>TCP</string> <number tooltip='1.1.1.52: 22 (TCP) '>22</number> </row> <row> <string>UDP</string> <number tooltip='1.1.1.52: 0 (UDP) '>0</number> <number tooltip='1.1.1.152: 0 (UDP) '>0</number> </row> <row> </pre>

	<pre> <string>ICMP</string> <number tooltip='1.1.1.52: 0 (ICMP) '>0</number> </row> ... </pre> <p>Se separan las IPs de destino por protocolo destacando la cantidad de accesos a cada una de ellas (22, 0, 0, 0 respectivamente).</p>
Rect()	<pre> ... <row> <null/> <string>443</string> <string>63938</string> </row> <row> <string>TCP</string> <number bevel='data' shadow='high'>114</number> <number bevel='data' shadow='high'>0</number> </row> <row> <string>UDP</string> <number bevel='data' shadow='high'>0</number> <number bevel='data' shadow='high'>119</number> </row> <row> <string>ICMP</string> <number bevel='data' shadow='high'>0</number> <number bevel='data' shadow='high'>0</number> </row> ... </pre> <p>En el primer bloque (row) aparecen los puertos accedidos, luego en los siguientes, se describen la cantidad de accesos por protocolo a cada uno de los puertos del inicio en orden. Similar al XML de Rad, la única diferencia es la propiedad "shadow" (sombra) que tiene relación con el aspecto del gráfico.</p>
bubble()	<pre> ... <string>TCP</string> <number bevel='bevel1'>25</number> <string>25</string> <number tooltip='25 accesos, desde: 208.84.147.187 al puerto: 25'>25</number> </pre>

	<pre> <string>UDP</string> <number bevel='bevel1'>53</number> <number>174</number> <number tooltip='174 accesos, desde: 200.89.70.3 al puerto: 53'>174</number> <string>ICMP</string> <number bevel='bevel1'>0</number> <number>0</number> <number tooltip='0'>0</number> ... </pre> <p>Se despliegan los accesos por IP hacia cada puerto separados por protocolo.</p>
time()	<pre> ... <string>Puerto 3389</string> <number shadow='low' tooltip='hora: 00:03, 21 accesos'>0003</number> <number>21</number> <number shadow='low' tooltip='hora: 00:00, 162 accesos'>0000</number> <number>162</number> </row> <row> <string>Puerto 53</string> <number shadow='low' tooltip='hora: 00:00, 58 accesos'>0000 </number> <number>58</number> </row> ... </pre> <p>Se despliegan los accesos por hora a cada puerto (Ejemplo: el puerto 3389 fue accedido 21 veces a las 00:03 → durante un minuto).</p>
timeIP()	<pre> ... <string>IP 201.186.43.46</string> <number shadow='low' tooltip='hora: 00:03, 177 accesos'>0003</number> <number>177</number> </row> <row> <string>IP 200.89.70.3</string> <number shadow='low' tooltip='hora: 00:02, 29 accesos'>0002</number> </pre>

	<pre><number>29</number> </row> ...</pre> <p>Se despliegan los accesos por hora de cada IP (Ejemplo: la IP 201.186.43.46 accedió 177 veces la darknet a las 00:03 → durante un minuto).</p>
timeAtaIP()	<pre>... <string>IP 1.1.1.43</string> <number shadow='low' tooltip='hora: 00:03, 177 accesos'>0003</number> <number>177</number> </row> <row> <string>IP 1.1.1.3</string> <number shadow='low' tooltip='hora: 00:02, 29 accesos'>0002</number> <number>29</number> </row> ...</pre> <p>Se despliegan los accesos por hora de cada IP (Ejemplo: la IP 1.1.1.43 fue accedida 177 veces a las 00:03 → durante un minuto).</p>
ringProto()	<pre>... <row> <string>TCP</string> <number tooltip='dia: 1, accesos: 272' line_color='eeffee' line_thickness='2' line_alpha='50'>272</number> <number tooltip='dia: 2, accesos: 51455' line_color='eeffee' line_thickness='2' line_alpha='50'>51455</number> <number tooltip='dia: 3, accesos: 17271' line_color='eeffee' line_thickness='2' line_alpha='50'>17271</number> <number tooltip='dia: 4, accesos: 30496' line_color='eeffee' line_thickness='2' line_alpha='50'>30496</number> <number tooltip='dia: 5, accesos: 31161' line_color='eeffee' line_thickness='2' line_alpha='50'>31161</number> <number tooltip='dia: 6, accesos: 1927' line_color='eeffee' line_thickness='2' line_alpha='50'>1927</number> <number tooltip='dia: 7, accesos: 17731' line_color='eeffee' line_thickness='2' line_alpha='50'>17731</number> </row> <row> <string>UDP</string> <number tooltip='dia: 1, accesos: 119' line_color='eeffee' line_thickness='2' line_alpha='50'>119</number></pre>

	<pre> <null/> <null/> <null/> <null/> <number tooltip='dia: 6, accesos: 56' line_color='eeffee' line_thickness='2' line_alpha='50'>56</number> <null/> </row> ... </pre> <p>Este XML muestra los accesos por día separados por protocolo. Aquí se puede apreciar que en caso de que no haya accesos en cierto protocolo durante un día, se genera el tag <null>. Esto es debido a que el “row” de cada protocolo deben tener el mismo largo (una línea por cada día de la semana).</p>
RectUDP()	<pre> ... <row> <null/> <string>443</string> <string>63938</string> </row> <row> <string>TCP</string> </row> <row> <string>UDP</string> <number bevel='data' shadow='high'>0</number> <number bevel='data' shadow='high'>119</number> </row> <row> <string>ICMP</string> </row> ... </pre> <p>En el primer bloque (row) aparecen los puertos accedidos, luego en los siguientes, se describen la cantidad de accesos por protocolo (sólo a UDP en este caso) a cada uno de los puertos del inicio en orden. Similar al XML de Rad, la única diferencia es la propiedad “shadow” (sombra) que tiene relación con el aspecto del gráfico.</p>
RectTCP()	<pre> ... <row> </pre>

```

<null/>
<string>443</string>
<string>63938</string>
</row>
<row>
<string>TCP</string>
<number bevel='data' shadow='high'>114</number>
<number bevel='data' shadow='high'>0</number>
</row>
<row>
<string>UDP</string>
</row>
<row>
<string>ICMP</string>
</row>
...

```

En el primer bloque (row) aparecen los puertos accedidos, luego en los siguientes, se describen la cantidad de accesos por protocolo (sólo a TCP en este caso) a cada uno de los puertos del inicio en orden. Similar al XML de Rad, la única diferencia es la propiedad "shadow" (sombra) que tiene relación con el aspecto del gráfico.

CC. Código del script tiraAlerta.sh

```

#!/bin/bash
year=$(date --date='1 day ago' +%Y)
month=$(date --date='1 day ago' +%m)
day=$(date --date='1 day ago' +%d)
archivo="salida-"$year-"$month-"$day
cd /home/rfaccilo/Desktop
./generadorAlerta.sh 00h00m00s 23h59m59s $archivo "all"
./buscador.sh

```

DD. Código del script generadorAlertas.sh

```

#!/bin/bash
HOME_DIR=/home/rfaccilo/Desktop
cd $HOME_DIR/capturas
/usr/sbin/tcplice -w cortadoAl_${1}_${2}_${3} $1 $2 $3 &>
$HOME_DIR/errorrtcpslice.txt
/usr/sbin/tcpdump -nr cortadoAl_${1}_${2}_${3} &> unoAl_${1}_${2}_${3}.txt
sed '1d' unoAl_${1}_${2}_${3}.txt > tempAl_${1}_${2}_${3}.txt

```

```

cat tempAl_${1}_${2}_${3}.txt | sed 's/\. / /g' | sed 's:/ /' | sed 's:// /' |
sed 's:// /' | sed 's:// /' > cortadoAl_${1}_${2}_${3}.txt

inicio=$1
fin=$2
archivo=$3
tipo=$4
shift 3
python2.6 $HOME_DIR/Parser/src/alertaProtocolo.py
$HOME_DIR/capturas/cortadoAl_${inicio}_${fin}_${archivo}.txt $@ &>
$HOME_DIR/errorprotocolo.txt
python2.6 $HOME_DIR/Parser/src/alertaPuerto.py
$HOME_DIR/capturas/cortadoAl_${inicio}_${fin}_${archivo}.txt $@ &>
$HOME_DIR/errorpuerto.txt
    cd $HOME_DIR/capturas
    rm cortado_*
    rm fileLimpio*

```

EE. Código del script *buscador.sh*

```

#!/bin/bash
year=$(date --date='1 day ago' +%Y)
month=$(date --date='1 day ago' +%m)
day=$(date --date='1 day ago' +%d)

dia=$year-"-$month"-"$day

IFS=$" "
set $(cat /home/rfaccilo/Desktop/datos.txt)

maxTCP=$3
maxUDP=$4
maxICMP=$5
maxpuertos=$6
puerto1=$7
puerto2=$8
puerto3=$9
puerto4=${10}
puerto5=${11}
etiqueta1=$1
etiqueta2=$2

#caso protocolos

    cat /home/rfaccilo/Desktop/datosParser/alertaProtocolo.txt | tr -d '"'
| awk -v etiqueta1="$etiqueta1" -v maxTCP="$maxTCP" -v dia="$dia" -F " "
'$1=="TCP" && $2>=maxTCP {print dia ": TCP usado " $2 " veces. (" etiqueta1
")" }' >> /home/rfaccilo/Desktop/resumen.txt

    cat /home/rfaccilo/Desktop/datosParser/alertaProtocolo.txt | tr -d '"'
| awk -v etiqueta1="$etiqueta1" -v maxUDP="$maxUDP" -v dia="$dia" -F " "
'$1=="UDP" && $2>=maxUDP {print dia ": UDP usado " $2 " veces. (" etiqueta1
")" }' >> /home/rfaccilo/Desktop/resumen.txt

```

```

cat /home/rfaccilo/Desktop/datosParser/alertaProtocolo.txt | tr -d '"' | awk -
v etiquetal="$etiquetal" -v maxICMP="$maxICMP" -v dia="$dia" -F " "
'$1=="ICMP" && $2>=maxICMP {print dia ": ICMP usado " $2 " veces. (" etiquetal
")" }' >> /home/rfaccilo/Desktop/resumen.txt

```

#caso 5 puertos

```

cat /home/rfaccilo/Desktop/datosParser/alertaPuerto.txt | tr -d '"' |
awk -v etiqueta2="$etiqueta2" -v puerto1="$puerto1" -v dia="$dia" -v
maxpuertos="$maxpuertos" -F " " '$1==puerto1 && $2>=maxpuertos {print dia ":
Puerto " $1 " accedido " $2 " veces. (" etiqueta2 ")"}' >>
/home/rfaccilo/Desktop/resumen.txt

```

```

cat /home/rfaccilo/Desktop/datosParser/alertaPuerto.txt | tr -d '"' |
awk -v etiqueta2="$etiqueta2" -v puerto2="$puerto2" -v dia="$dia" -v
maxpuertos="$maxpuertos" -F " " '$1==puerto2 && $2>=maxpuertos {print dia ":
Puerto " $1 " accedido " $2 " veces. (" etiqueta2 ")"}' >>
/home/rfaccilo/Desktop/resumen.txt

```

```

cat /home/rfaccilo/Desktop/datosParser/alertaPuerto.txt | tr -d '"' |
awk -v etiqueta2="$etiqueta2" -v puerto3="$puerto3" -v dia="$dia" -v
maxpuertos="$maxpuertos" -F " " '$1==puerto3 && $2>=maxpuertos {print dia ":
Puerto " $1 " accedido " $2 " veces. (" etiqueta2 ")"}' >>
/home/rfaccilo/Desktop/resumen.txt

```

```

cat /home/rfaccilo/Desktop/datosParser/alertaPuerto.txt | tr -d '"' |
awk -v etiqueta2="$etiqueta2" -v puerto4="$puerto4" -v dia="$dia" -v
maxpuertos="$maxpuertos" -F " " '$1==puerto4 && $2>=maxpuertos {print dia ":
Puerto " $1 " accedido " $2 " veces. (" etiqueta2 ")"}' >>
/home/rfaccilo/Desktop/resumen.txt

```

```

cat /home/rfaccilo/Desktop/datosParser/alertaPuerto.txt | tr -d '"' |
awk -v etiqueta2="$etiqueta2" -v puerto5="$puerto5" -v dia="$dia" -v
maxpuertos="$maxpuertos" -F " " '$1==puerto5 && $2>=maxpuertos {print dia ":
Puerto " $1 " accedido " $2 " veces. (" etiqueta2 ")"}' >>
/home/rfaccilo/Desktop/resumen.txt

```

FF. Código del método nadaUDP

```

def nadaUDP(File,FileClean):

    try:

        E = open(File, "r")

        S = open(FileClean, "w")

    except:

        print"Error opening %s." % (File)

        sys.exit(1)

```

```

for line in E:

    data = re.split(" ", line)

    if(data[9].isdigit() and data[17] == "UDP,"):

        S.writelines(line)

E.close()

S.close()

```

GG. Código del método nadaTCP

```

def nadaTCP(File,FileClean):

    try:

        E = open(File, "r")

        S = open(FileClean, "w")

    except:

        print"Error opening %s." % (File)

        sys.exit(1)

    for line in E:

        data = re.split(" ", line)

        if(data[9].isdigit() and data[17] == "tcp"):

            S.writelines(line)

    E.close()

    S.close()

```

HH. Código del método listaPuertosUDP

```

def listaPuertosUDP():

    map = mapGroup("fileLimpio_listaPuertosUDP.txt",
"mycount", "notChange", "listaPu")

    reduce14 = reduce(map, "sum")

    txt = printListaPuertos(reduce14)

```



```
f=open("/home/rfaccilo/Desktop/capturas/PuertosUDP.txt","w")

f.writelines(txt)

f.close()
```

II. Código del método listaPuertosTCP

```
def listaPuertosTCP():

    map = mapGroup("fileLimpio_listaPuertosTCP.txt",
"mycount","notChange","listaPu")

    reduce14 = reduce(map,"sum")

    txt = printListaPuertos(reduce14)

    f=open("/home/rfaccilo/Desktop/capturas/PuertosTCP.txt","w")

    f.writelines(txt)

    f.close()
```

JJ. Código del método listaAtaIP

```
def listaAtaIP():

    map = mapGroup("fileLimpio_listaAtaIP.txt",
"mycount","notChange","listaAtaIP")

    reduce13 = reduce(map,"sum")

    txt = printListaIP(reduce13)

    f=open("/home/rfaccilo/Desktop/capturas/IPAta.txt","w")

    f.writelines(txt)

    f.close()
```

KK. Código del método listaAtaIP(map)

```
def listaAtaIP(data):

    (key, val) = (data[11]+ "." +data[12]+ "." +data[13]+ "." +data[14],None)
    return (key, val)
```

LL. Código del script `países.sh`

```
#!/bin/bash

year=$(date --date='1 day ago' +%Y)
month=$(date --date='1 day ago' +%m)
day=$(date --date='1 day ago' +%d)

archivo="salida-"$year-"-$month"-"$day
directorio=$year-"-$month"-"$day

echo "country access" >> datosPaises${archivo}.txt

awk '{ print $5, $7}' países${archivo}.txt >> datosPaises${archivo}.txt

awk 'BEGIN{FS=" "; print "country count total avg"} NR!=1
{a[$1]++;b[$1]=b[$1]+$2}END{for (i in a) printf("%s %10.0f %10.0f
%10.2f\n", i, a[i], b[i], b[i]/a[i])}' datosPaises${archivo}.txt >
temp_datosPaises${archivo}.txt

sed "1d" temp_datosPaises${archivo}.txt >
temp2_datosPaises${archivo}.txt

awk '{ print $1, $3}' temp2_datosPaises${archivo}.txt >
temp3_datosPaises${archivo}.txt

sed '11,99999999d' temp3_datosPaises${archivo}.txt >
datosPaisesFinal${archivo}.txt

rm temp_*
rm temp2_*
rm temp3_*
rm datosPaises${archivo}.txt

echo "<chart>

<license>HT7Q1M4SR.EAOZCI3PQUDSQT34CJOL</license>
```

```

        <axis_ticks value_ticks='true' category_ticks='true'
minor_count='1' />

        <axis_value shadow='low' size='10' color='FFFFFF' alpha='75' />

        <chart_border top_thickness='0' bottom_thickness='2'
left_thickness='2' right_thickness='0' />

        <chart_data>

                <row>

                        <null/>" >>
/home/rfaccilo/Desktop/graficosFlash/${directorio}/compositel.xml

cat datosPaisesFinal${archivo}.txt | awk '{ print "<string>" $1
"</string>" }' >>
/home/rfaccilo/Desktop/graficosFlash/${directorio}/compositel.xml

echo " </row>

                <row>

                        <string>Número de accesos diarios</string>" >>
/home/rfaccilo/Desktop/graficosFlash/${directorio}/compositel.xmlcomilla
="'"

cat datosPaisesFinal${archivo}.txt | awk '{ print "<number
tooltip=\047" $2 " accesos\047>" $2 "</number>" }' >>
/home/rfaccilo/Desktop/graficosFlash/${directorio}/compositel.xml

echo " </row>

        </chart_data>

<chart_grid_h thickness='1' type='solid' />

        <chart_grid_v thickness='1' type='solid' />

        <chart_rect x='80' y='60' width='320' height='150'
positive_color='888888' positive_alpha='50' />

```

```

    <chart_pref rotation_x='15' rotation_y='0' min_x='0' max_x='80'
min_y='0' max_y='60' />

    <chart_type>stacked 3d column</chart_type>

    <draw>

        <text shadow='high' color='ffffee' alpha='75' size='42' x='100'
y='7' width='400' height='50' h_align='left' v_align='bottom'>Accesos
por paÑ-s</text>

    </draw>

<filter>

        <shadow id='high' distance='5' angle='45' alpha='35'
blurX='15' blurY='15' />

        <shadow id='low' distance='2' angle='45' alpha='50'
blurX='5' blurY='5' />

    </filter>

    <legend shadow='high' x='35' y='250' width='410' height='50'
margin='20' fill_color='000000' fill_alpha='7' line_color='000000'
line_alpha='0' line_thickness='0' layout='horizontal' size='12'
color='000000' alpha='50' />

    <tooltip color='ffffcc' alpha='80' size='12'
background_color_1='444488' background_alpha='75' shadow='low' />

    <series_color>

        <color>88ffff</color>

    </series_color>

    <series bar_gap='0' set_gap='20' />

</chart>" >>
/home/rfaccilo/Desktop/graficosFlash/${directorio}/compositel.xml

```