

**Universidad de Buenos Aires
Facultades de Ciencias
Económicas,
Cs. Exactas y Naturales e Ingeniería**

Maestría en Seguridad Informática

Tesis

Técnicas y herramientas forenses
para la detección de Botnets

Autor: Ing. Juan A. Devincenzi
Director de Tesis: Ing. Hugo Pagola

AÑO - 2011

Cohorte

2009

Resumen

Una Botnet es una red formada por un conjunto de hosts de un tamaño determinado bajo el control centralizado de un tercero denominado Bot Herder; dicho control se consigue mediante la instalación de un programa llamado Bot en cada host.

El objetivo del presente trabajo es: Proporcionar un conjunto de herramientas y técnicas forenses adaptándolas para la detección e identificación de Botnets, y ofrecer una guía de implementación de las herramientas más significativas.

El aporte de este trabajo consiste primeramente en generar una metodología integral que mediante un enfoque Top-Down (de lo general a lo particular), es decir yendo desde la red hacia los hosts, instruya en cómo detectar Botnets. Para ello se adaptarán algunas herramientas que no son específicas para detectar Botnets (como Snort y Nepenthes) y se propondrán un conjunto de técnicas forenses especialmente pensadas para detectar e identificar Botnets.

Palabras clave

Botnet, Bot, Bot Herder, Honeypot, IDS.

Índice

Resumen.....	3
Palabras clave	3
Índice.....	5
Introducción	7
Sobre la metodología para la detección de Botnets	9
Conceptualización de una Botnet como un Worm	9
Capítulo 1: Snort aplicado a la detección de Botnets	11
Introducción.....	11
Introducción al protocolo IRC	13
Sobre la arquitectura y el funcionamiento de Snort	15
Instalando Snort	17
Comprendiendo las reglas de Snort	19
El archivo de configuración snort.conf	21
Proceso de detección de Botnets con Snort.....	24
Un modelo iterativo de detección de Botnets y mejora de reglas	26
Capítulo 2: Detección de Botnets con Nepenthes	33
Introducción a los Honeypots	33
Introducción a Nepenthes	34
Diseño arquitectónico de Nepenthes.....	35
Despliegue e instalación del sensor	37
Testing inicial de Nepenthes.....	38
Descripción del funcionamiento básico de Nepenthes	40
Procedimiento para la operación de Nepenthes	41
Capítulo 3: Ourmon como herramienta de detección de Botnets	47
Introducción y arquitectura	47
Instalación y despliegue de la herramienta.....	48

Sobre los reportes	50
Un enfoque iterativo para la detección de Botnets y generación de reglas de Snort	60
Capítulo 4: Forensia sobre host y malware	65
Introducción.....	65
Alcance pericial y consideraciones previas.....	66
Alcances y consideraciones sobre la pericia de malware.....	67
Ejecutando la pericia sobre host.....	68
Pericia sobre el malware	77
Conclusiones.....	87
Anexos.....	89
Anexo A: Instalación de Snort y componentes periféricos.....	89
Anexo B: Instalación y configuración de NTOP y Wireshark.....	94
Anexo C: Instalación y configuración de Nepenthes.....	97
Anexo D: Instalación y configuración de Ourmon.....	102
Bibliografía.....	107

Introducción

Las Botnets son una amenaza real, concreta y en crecimiento constante. Actualmente el problema de las Botnets casi se ha salido de control y el número de hosts zombie crece día a día. En este contexto turbulento se enmarca la investigación de este trabajo; el objetivo general del mismo es el siguiente: “Proporcionar un conjunto de herramientas y técnicas forenses adaptándolas para la detección e identificación de Botnets, y ofrecer una guía de implementación de las herramientas más significativas.”

La hipótesis en la que se inscribe este trabajo y que se intentará verificar a lo largo del mismo es la siguiente: “Se pueden detectar Botnets mediante la utilización de las herramientas Snort, Ourmon, y Nepenthes y un conjunto de herramientas y técnicas forenses para la identificación de los Bots instalados en los hosts.”

La idea general del trabajo es la de proponer una metodología integral que mediante un enfoque Top-Down informe sobre cómo detectar Botnets. Para ello se van a presentar e implementar un conjunto de herramientas específicas para la detección de Botnets en un entorno organizacional utilizando el enfoque forense para la utilización de dichas herramientas.

Dado que existen herramientas aplicables tanto a nivel de red como a nivel de host, se propone utilizar la metodología Top-Down comenzando desde “más arriba” y así describiendo e implementando tres herramientas específicas para la detección de Botnets a nivel de red como Ourmon, Nepenthes y Snort; y luego bajar al nivel de host y presentar allí un conjunto de herramientas y técnicas forenses para la detección de Bots a ese nivel.

Si bien existe un conjunto más amplio de herramientas a nivel de red para implementar se seleccionó a Ourmon, Nepenthes y Snort siguiendo algunos criterios generales y otros específicos.

Entre los criterios generales que involucran a estas 3 herramientas por igual se destacan que las 3 son gratuitas, se encuentran bien documentadas y todas también cuentan con una comunidad de usuarios que puede brindar soporte en caso de tener problemas con su uso o implementación. El hecho de que estas herramientas estén bien documentadas y tengan suficiente soporte resultó una característica primordial para su elección dado que de

esta forma se simplifica que futuros lectores de este trabajo puedan repetir las experiencias aquí presentadas e incluso ampliar aun más los conceptos. Así mismo el hecho de que sean herramientas gratuitas permite que más interesados puedan realizar estas implementaciones en un ámbito hogareño y que más organizaciones puedan llegar a utilizarlas.

Los criterios específicos que se aplicaron para la selección son independientes de cada herramienta.

Snort se seleccionó porque es una herramienta muy conocida de la cual hay mucha documentación incluso en formato de libro. Adicionalmente el autor de este trabajo ya tuvo experiencias previas con esta herramienta lo cual también jugó a favor de su elección. Snort es gratuito en sí, siendo pago nada más la suscripción para actualizarlo constantemente con las firmas más recientes o algunos módulos adicionales; dichos módulos no aplican a este trabajo y el hecho de actualizarlo con las firmas más recientes se recomienda aunque no es indispensable para este trabajo. De esta forma se considera como una herramienta y una opción gratuita excelentemente documentada y probada.

Nepenthes no solo es gratuito y está bien documentado, sino que también es la única herramienta tipo Honeypot que tiene la capacidad de capturar Bots y Worms como lo hace, por lo cual resulta irremplazable para este trabajo y sus objetivos.

Finalmente Ourmon es una herramienta gratuita que cuenta con una buena documentación y que tiene una característica distintiva: está pensada específicamente para detectar Bots y Botnets. De esta forma no hace falta adaptarla para la detección de estas amenazas como es el caso de Snort.

Resumiendo puede decirse que todas las características hasta aquí expuestas contribuyeron a la elección de estas 3 herramientas para la realización de este trabajo.

Sobre la metodología para la detección de Botnets

Las Botnets son a la vez una amenaza latente y un problema complejo; para evitar que una Botnet siga dañando los activos una vez que se insertó en una red hay que detectarla a tiempo y luego anularla.

Dentro de la rama de la Seguridad Informática que se encarga de estudiar el malware del tipo Botnets existe un amplio conjunto de enfoques en lo que se refiere a su detección y anulación. La idea del presente trabajo es presentar un conjunto diverso de técnicas para la detección de Botnets e implementar las más notables mediante las herramientas antes mencionadas. A continuación probablemente al lector le surja la siguiente pregunta ¿Cómo se detecta una Botnet? La respuesta a esta pregunta conformará el desarrollo este trabajo, pero inicialmente se puede afirmar que lo más conveniente es utilizar un enfoque que vaya desde lo general a lo particular (también denominado Top-Down); es decir, que si se desea detectar el funcionamiento de una Botnet considerando una red mediana/grande (como la que podría tener una Pyme o empresa grande) se debe comenzar por el análisis del tráfico de dicha red para identificar el tráfico producto de una actividad “normal” y aquel producto de una actividad extraordinaria como el que podría corresponder a una Botnet (o Worm) en la red. Contando con las herramientas de análisis de tráfico y/o patrones de red antes mencionadas (Snort y Ourmon) y mediante la metodología que se explicará en este trabajo, el Ingeniero de Seguridad responsable debería ser capaz de identificar desde que hosts podría provenir el tráfico sospechoso y luego realizar una investigación más en detalle utilizando diversas técnicas forenses sobre dichos hosts de forma de identificar si efectivamente existe un malware residente en los mismos de forma de anularlo y en lo posible identificarlo. Los Honeypots también tienen incidencia a lo largo de este proceso, ya que pueden ayudar a identificar Botnets (u otro tipo de malware) y aprender de su funcionamiento; por lo tanto pueden servir tanto como sistemas de alerta temprana así como herramientas forenses que el experto utilizará para analizar una pieza de malware determinada; aquí se utilizará Nepenthes como herramienta tipo Honeypot.

Resumiendo, en el presente trabajo se empleará una metodología Top-Down para atacar el problema de las Botnets yendo desde lo general a lo particular. Para ello se introducirán e implementarán las herramientas previamente mencionadas y se utilizará siempre un enfoque forense para identificar y anular todas las amenazas que se puedan encontrar.

Conceptualización de una Botnet como un Worm

Un concepto importante a introducir es aquel relacionado con comprender a una Botnet como una mera variante de un tipo de malware denominado Worm.

Wikipedia propone la siguiente definición de Worm: “A computer worm is a self-replicating Malware computer program. It uses a computer network to send copies of itself to other nodes (computers on the network) and it may do so without any user intervention. This is due to security shortcomings on the target computer.” [1]

Las Botnets utilizan un mecanismo similar a los Worms para expandirse en forma autónoma y automática; pero a diferencia de estos últimos tienen la capacidad de poder controlarse en forma remota por un operador humano. Es justamente este mecanismo de expansión automática lo que convierte a las Botnets en una amenaza tan perenne; a su vez la posibilidad del control remoto permite darles muchos más usos que a un Worm por lo cual se pueden considerar incluso como más nocivas.

Dado que las Botnets son en realidad Worms con un sofisticado mecanismo de control remoto, es posible utilizar muchas de las técnicas y herramientas forenses que se utilizan para detectar Worms sobre las Botnets; asimismo cuando se desarrollen mecanismos para la defensa y detección de Botnets, también se estará luchando a la vez contra los Worms.

Capítulo 1: Snort aplicado a la detección de Botnets

Introducción [2] [3]

Snort entra dentro de la categoría de los sistemas tipo IDS, pero: ¿Qué es exactamente un IDS? Una definición de IDS puede ser la de un Sistema que monitorizando el tráfico de una red y confrontando el mismo contra una serie de patrones predeterminados, puede discriminar la actividad “normal” de la red de aquella “anormal”. De hecho, las siglas IDS significan Intrusion Detection System o Sistema de Detección de Intrusiones, lo cual es justamente la función de este tipo de sistemas.

Básicamente el funcionamiento de Snort consiste en monitorizar continuamente el tráfico de una determinada red (o varias) y en base a un conjunto de reglas predeterminadas (que son totalmente configurables) actuar detectando los patrones que dichas reglas inducen a detectar. Todo patrón que coincida con alguna de las reglas será informado a los interesados por diversos medios que pueden abarcar desde una interfaz web, mails, mensajes de texto, etcétera.

Snort fue creado con la idea de brindar alertas tempranas de actividad anormal en una red, de forma que los administradores (u otro personal capacitado) puedan analizarlas para detectar posibles intentos de ataque o remediar ataques ya perpetrados. El hecho de ofrecer alertas tempranas permite que los Administradores cuenten con valiosa información de la situación de sus redes y servidores.

Hasta el momento se dijo que Snort entraba dentro de la categoría de los IDS, sin embargo Snort es también capaz de actuar como un IPS. Esta sigla es un acrónimo que significa Intrusion Prevention System o Sistema de Prevención de Intrusiones. Básicamente la diferencia entre un IDS y un IPS es que el IDS es pasivo y el IPS activo. Entonces ¿Qué implica esto? Implica que un IDS solo será capaz de enviar alertas a quien corresponda y bajo la actividad que corresponda (según su configuración) y no realizará acción alguna ante la actividad anormal. Un IPS tiene las funciones de un IDS, pero además es capaz de actuar activamente (previa configuración) ante una

actividad anormal; por ejemplo en caso de detectar la presencia de un intruso en el sistema protegido, podría cortarle la conexión al sistema a dicho intruso y además denegarle futuros intentos de acceso.

En lo que respecta al uso que se le planea dar a Snort en el presente trabajo, se utilizarán solamente las funciones de Snort como IDS, las cuales serán suficientes para detectar la presencia de Botnets.

Introducción al protocolo IRC [20]

IRC es un protocolo de comunicación que permite que múltiples usuarios se conecten a canales determinados en donde puedan intercambiar información que ingresan mediante teclado; es decir que es un sistema de chat cliente/servidor.

El protocolo fue desarrollado en el año 1988 por Jarkko Oikarinen y se lo estandarizó posteriormente mediante la RFC 1459.

Como parte de su funcionamiento, un servidor de IRC otorga una serie de canales que siempre empiezan con # (por ejemplo #Excavadoras) al que se conectan los usuarios mediante clientes IRC instalados en sus PC's. Todos los usuarios que se conectan a algún canal de un servidor IRC deben tener un nombre único denominado nickname que se elige al momento de conectarse al canal.

El protocolo IRC resulta importante para el estudio de los Botnets puesto que los mismos utilizan comúnmente a este protocolo como CyC. Es decir que un Bot determinado instalado en un host se comunica con su creador y el resto de la Botnet utilizando el protocolo IRC. De esta forma se hace importante conocer cuáles son los comandos IRC básicos, a saber:

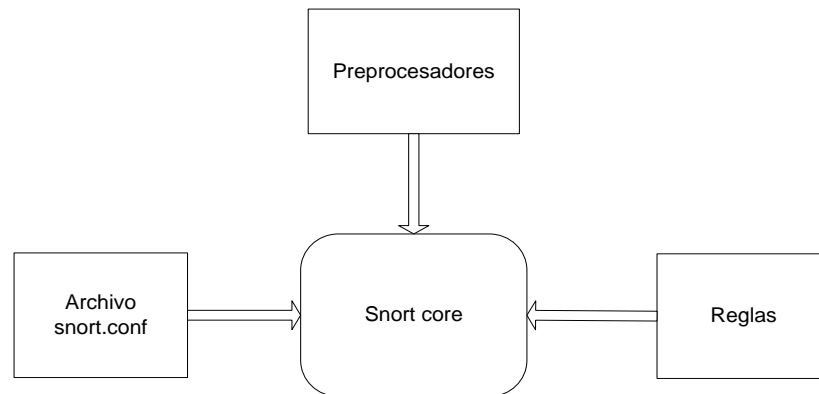
- NICK: identifica el usuario con un nickname.
- USER: identifica al usuario cliente y servidor (al igual que el NICK son los primeros comandos al ejecutarse antes de acceder a un canal IRC).
- JOIN: une un usuario a un canal específico.
- MODE: permite que un administrador realice modificaciones sobre un canal IRC.
- PING y PONG: se utilizan para asegurarse de que un enlace entre el cliente y un servidor está activo; el cliente hace un PING y el servidor responde con un PONG.

- PRIVMSG: permite que un usuario envíe mensajes privados a otro usuario.
- DCC SEND: para enviar archivos entre usuarios usando el protocolo CTCP.
- XDCC: una forma en que un script puede llamar a DCC para iniciar una transferencia de archivos.
- CONNECT: permite que un servidor IRC se contacte con otro para hacer de Gateway.

Todos estos mensajes de IRC pueden ser usados como tokens para detectar la actividad de posibles Botnets IRC. Tener en cuenta que algunos Bots cambian estos mensajes estándar por otros o bien cifran el tráfico para evitar detección.

Sobre la arquitectura y el funcionamiento de Snort [2] [3] [5] [12]

Arquitectónicamente se puede conceptualizar a Snort de la siguiente forma:



El núcleo de Snort (el demonio bajo el cual corre) se encuentra alimentado por tres componentes esenciales que se describen a continuación:

- **Preprocesadores:** son los encargados de realizar tareas preliminares sobre el tráfico antes de que Snort pueda procesar dicho tráfico; podría decirse que normalizan o “masajea” el tráfico que recibe Snort. Esta adecuación (o normalización) sobre el tráfico, ayuda a que el mismo pueda ser más eficientemente comparado contra la base de reglas disponibles y se reduzcan tanto los falsos positivos como los falsos negativos.
- **Reglas:** las reglas son un conjunto de imperativos que instruyen a Snort sobre lo que se debe considerar una amenaza y sobre lo que no.
- **Archivo snort.conf:** es el archivo principal de configuración donde se incluyen la configuración de los preprocesadores, las reglas a aplicar, las formas de notificación entre muchas otras.

En su nivel más esencial Snort puede conceptualizarse como un Sniffer. Un Sniffer es un software capaz de capturar la actividad de la red, procesarla y mostrarla por pantalla (o bien bajarla a un archivo) aplicando una variedad de filtros predefinidos por un administrador.

Para hacer que Snort se comporte como un Sniffer y loguee los paquetes capturados a un archivo, se lo puede iniciar del siguiente modo:

```
# snort -dve > temp.log
```

Cuando Snort actúa como un Sniffer no tiene capacidad de decisión, es decir que solo monitoriza el tráfico de red y lo muestra a los interesados pero no compara patrones contra las reglas ni mucho menos toma decisiones sobre las mismas. Para que Snort comience a actuar como un IDS (o bien como un IPS según la necesidad) hará falta alimentarlo con el archivo de configuración snort.conf que a su vez instruirá a Snort con las reglas que se aplicarán y los preprocesadores utilizados.

Para iniciar Snort como un IDS se deberá referenciar el archivo de configuración (como anteriormente se mencionó) del siguiente modo:

```
# snort -u snort -c /etc/snort/snort.conf
```


Instalando Snort [2] [4] [9] [13]

Ahora que ya se describió que es Snort y cuáles son sus componentes principales, se puede proceder a brindar instalarlo y configurarlo.

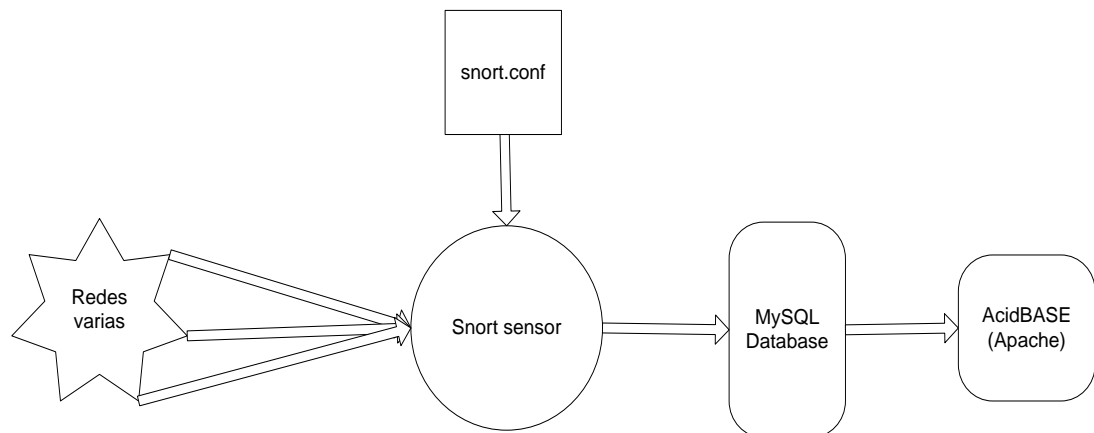
La instalación de Snort variará según la cantidad de componentes que se instalen para complementarlo. Estos componentes se encuentran principalmente relacionados con el repositorio para el almacenamiento de los logs que genere Snort, así como también con su visualización. Los logs se pueden almacenar en archivos planos que se depositarán en el File System o bien se los puede almacenar en una base de datos; asimismo la visualización de los logs se puede efectuar viendo directamente los logs “crudos” almacenados o bien mediante una herramienta especializada que mejore y facilite la visualización como así también la detección de patrones. Además, Snort funciona tanto en entornos tipo Windows o UNIX/Linux.

La instalación aquí propuesta es sobre un entorno Linux, específicamente Ubuntu, utilizando una base de datos tipo MySQL para almacenar los logs y la herramienta web ACID Base para visualizarlos. Para realizar una instalación de este tipo deben instalarse los siguientes componentes periféricos (en orden):

- MySQL
- Snort
 - MySQL Support
- Apache
 - GD Support
 - OpenSSL Support
 - MySQL Support
- PHP
 - GD Support
 - OpenSSL Support
 - MySQL Support

- ACID Base
 - Adodb
 - Jpgraph/phplot
 - GD

Una vez instalados y configurados todos estos componentes, la arquitectura de los mismos debería tener una apariencia como esta:



El sensor de Snort recibe el tráfico de múltiples redes, según la interconexión que tenga el host en donde se aloje el sensor; lo ideal es que la configuración de la red permita que el sensor pueda ver todo el tráfico de la red. Una vez que el tráfico llega al sensor, este lo parsea y normaliza con sus preprocesadores y luego en base a las reglas del archivo de configuración genera las alertas y las envía a una base de datos MySQL que cuenta con una estructura de datos normalizada en la cual Snort puede escribir.

Una vez que los logs con las alertas de Snort se encuentran en la base de datos (en MySQL en este caso) una herramienta web llamada ACID Base se encarga de recogerlos de la base, parsearlos y mostrarlos por pantalla en una interfaz web.

Dada la longitud del proceso de instalación de los componentes antes mencionados, se incluyó este proceso en una sección aparte denominada Anexo A.

Comprendiendo las reglas de Snort [6] [10]

Snort tiene la capacidad de detectar patrones basado en un conjunto de reglas, de esta forma las reglas son el componente más importante en el funcionamiento de Snort. Cada usuario es dueño de elegir que reglas quiere incluir dentro de las reconocidas por Snort; incluso puede crear e incluir sus propias reglas. Las reglas tienen una estructura con una cabecera fija y un conjunto de opciones variables; dado que existe gran cantidad de opciones, lo que se propone es describir las más representativas. Una regla tiene la siguiente estructura:

alert [Protocolo transporte] [IP/Red origen] [Puerto/s origen] [Dirección del tráfico] [IP/Red destino] [Puerto/s destino] ([Opciones])

La Cabecera es la siguiente:

- Protocolo transporte: tcp o udp
- IP/Red origen: puede ser una IP en concreto o una variable que se refiera a una subred como por ejemplo \$EXTERNAL_NET.
- Puerto/s origen: Se puede colocar un puerto en concreto, un rango, que no sea un puerto o un rango concreto, o any para referirse a todos los puertos.
- IP/Red destino: puede ser una IP en concreto o una variable que se refiera a una subred como por ejemplo \$HOME_NET.
- Puerto/s destino: Se puede colocar un puerto en concreto, un rango, que no sea un puerto o un rango concreto, o any para referirse a todos los puertos.
- Dirección del tráfico: puede ser ->, <-, <>

Entre las opciones más significativas de las reglas se encuentran las siguientes:

- Msg:"": Permite introducir un mensaje a elección del usuario que se mostrará cuando Snort encuentre tráfico que coincida con la regla.

- Classtype: Para categorizar el evento dentro de una categoría determinada y fija.
- Sid & Rev: Permite introducir un número único para diferenciar a la regla de otras así como también un número de revisión.
- Content:"": Para especificar el token a parsear.
- Nocase: Indica a Snort que no diferencie entre mayúsculas y minúsculas.
- Offset: Indica a partir de que byte se debe empezar a buscar el token.
- Depth: Indica la cantidad de bytes entre los cuales se debe buscar el token partiendo del offset.

Una regla de ejemplo podría ser la siguiente:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 3025:3050 (msg:"Escaneo  
ping con nmap"; classtype:misc-attack; Content:"PING"; nocase; offset:0;  
Depth:4; sid:6305; rev:1;)
```

El archivo de configuración snort.conf [5]

El archivo snort.conf permite definir cómo se va a comportar Snort en todos sus aspectos. Este archivo sigue una sintaxis específica que se detallará a continuación.

Variables de red y de configuración: se especifican con el prefijo var. Se puede especificar un solo host, varios hosts (encerrando entre corchetes y separando con comas), subnets (por ej. utilizando el subfijo /24), se puede utilizar un operador (!) para invertir el valor siguiente y hasta se pueden mezclar las todas las opciones mencionadas. Las variables más importantes son:

- HOME_NET: ésta variable identifica la red/hosts que va a “proteger” Snort.
- EXTERNAL_NET: ésta variable identifica la red/hosts que Snort considerará como posibles amenazas.
- RULE_PATH: indica la ruta hacia los archivos de reglas en el filesystem.

Una configuración de ejemplo es la siguiente:

```
> var HOME_NET [192.168.1.12,172.16.0.0/16,10.10.10.10,10.10.20.0/24]
```

```
> var EXTERNAL_NET !$HOME_NET
```

```
> var RULE_PATH /uncompressed/rules/path/
```

Configuraciones de salida (output): existen diversos plugins, muchos programados por terceros, para decidir como Snort va a informar de las alertas e información que detecta. Cabe destacar los siguientes:

- alert_syslog: este plugin enviará las alertas al syslog.
- log_tcpdump: este plugin volcará las alertas a un archivo específico.
- database: existen plugins para diversos motores de BD, en el caso de este trabajo se optó por MySQL.

La configuración queda de este modo:

```
> output database: log, mysql, user=snortuser password=snortpassword
dbname=snort host=localhost
```

Inclusión de reglas: las reglas se encuentran contenidas en archivos .rules dentro de la ruta especificada en RULE_PATH. Para cargar reglas al momento de iniciar Snort se puede hacer lo siguiente:

```
> include classification.config (rules classification & priority settings)

> include reference.config (links to web sites with information for all the
alerts)

> include $RULE_PATH/<rule_name>.rules (repetir esta sentencia por cada
archivo de reglas)
```

Las reglas con las que viene snort por defecto son las siguientes:

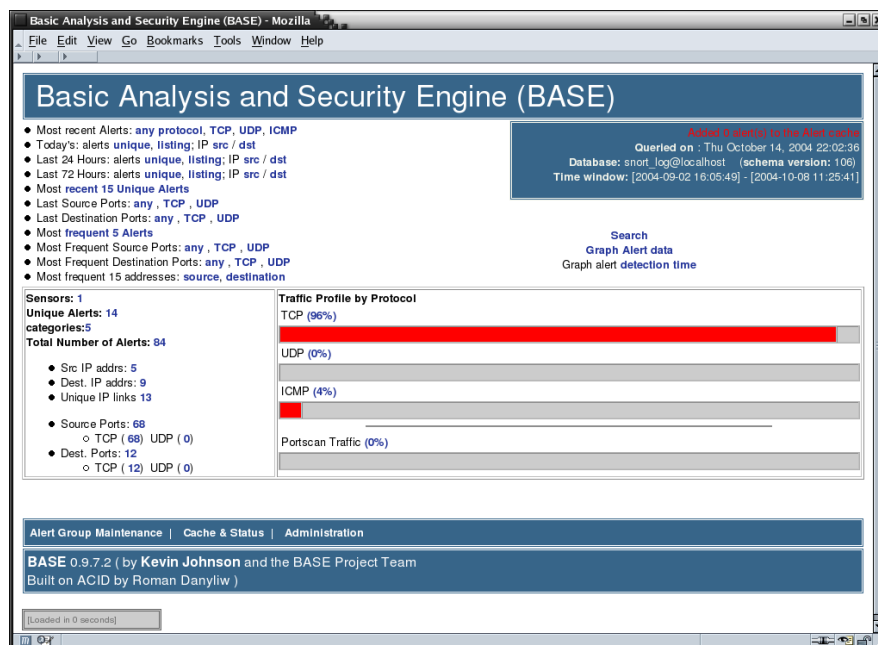
attack-responses.rules	local.rules	shellcode.rules
backdoor.rules	misc.rules	smtp.rules
bad-traffic.rules	multimedia.rules	snmp.rules
chat.rules	mysql.rules	sql.rules
ddos.rules	netbios.rules	telnet.rules
deleted.rules	nntp.rules	tftp.rules
dns.rules	oracle.rules	virus.rules
dos.rules	other-ids.rules	web-attacks.rules
experimental.rules	p2p.rules	web-cgi.rules
exploit.rules	policy.rules	web-client.rules
finger.rules	pop2.rules	web-coldfusion.rules
ftp.rules	pop3.rules	web-frontpage.rules
icmp.rules	rpc.rules	web-misc.rules
icmp-info.rules	porn.rules	web-iis.rules
imap.rules	rservices.rules	web-php.rules
info.rules	scan.rules	x11.rules

Configuración de los preprocesadores: se debe especificar en el archivo de configuración que preprocesadores va a utilizar Snort y con qué parámetros iniciales. Una configuración funcional de ejemplo es la siguiente:

- > preprocessor flow: stats_interval 0 hash 2
- > preprocessor frag2
- > preprocessor stream4: detect scans, disable_evasion_alerts, timeout 60, ttl_limit 10
- > preprocessor stream4_reassemble: both
- > preprocessor http_inspect: global iis_unicode_map unicode.map 1252
- > preprocessor http_inspect_server: server default profile all ports {80 8080}
- > preprocessor rpc_decode:111 32771 1024
- > preprocessor telnet_decode
- > preprocessor flow-portscan: server-watchnet [10.10.10.0/24,10.10.20.0/24]
- > preprocessor perfmonitor: time 300 file /var/tmp/snortstat pktcnt 10000

Proceso de detección de Botnets con Snort [2] [7] [11]

El proceso de detección de Botnets con Snort es sencillo, de hecho una vez realizada la instalación y configuración de Snort y sus componentes como se describe en el Anexo A, el operador debe simplemente ingresar a la página web de ACID Base; para ello se debe escribir <http://localhost/base> (o la IP del host donde se realizó la instalación) en su navegador web preferido e ingresar a la parte de Unique Alerts:



Dentro aparecerán todas las alertas asociadas a alguna actividad inusual.

	< Signature >	< Classification >	< Total # >	< Sensor # >	< Source Address >	< Dest. Address >	< First >	< Last >
<input type="checkbox"/>	snort		1	1	1	1		
<input type="checkbox"/>	snort		1	1	1	1		
<input type="checkbox"/>	[url] [nessus] [cve] [icaf] [bugtraq] [arachnIDS] [snort] WEB-IIS ISAPI .printer access	web-application-activity	1(13%)	1	1	1	2009-12-09 21:32:10	2009-12-09 21:32:10
<input type="checkbox"/>	[snort] (http_inspect) OVERSIZE REQUEST-URI DIRECTORY	unclassified	1(13%)	1	1	1	2009-12-09 21:32:10	2009-12-09 21:32:10
<input type="checkbox"/>	[nessus] [cve] [icaf] [cve] [icaf] [bugtraq] [bugtraq] [snort] FTP wu-ftp bad file completion attempt [misc-attack	1(13%)	1	1	1	2009-12-09 21:30:41	2009-12-09 21:30:41
<input type="checkbox"/>	[snort] (ftp_telnet) Invalid FTP Command	unclassified	1(13%)	1	1	1	2009-12-09 21:30:41	2009-12-09 21:30:41
<input type="checkbox"/>	[snort] (ftp_telnet) FTP traffic encrypted	unclassified	1(13%)	1	1	1	2009-12-09 21:30:41	2009-12-09 21:30:41
<input type="checkbox"/>	[url] [cve] [icaf] [snort] NETBIOS Microsoft Windows SMB malformed process ID high field remote code execution attempt	attempted-dos	1(13%)	1	1	1	2009-12-09 21:32:32	2009-12-09 21:32:32

Recordar que aparecerán alertas de muchos tipos de actividad de red, incluyendo la actividad de posibles Botnets; además cabe aclarar que el hecho de que aparezca una alerta no implica necesariamente que algo esté pasando pues Snort está asociado a Falsos Positivos.

En base a la descripción de la alerta que aparezca en “Signature” el operador podrá identificar claramente eventos asociados a la actividad de Botnets; utilizando los campos “Source Address” y “Destination Address” podrá identificar entre que hosts se produjo el tráfico de red anómalo.

Una vez que tenga todo identificado resta que en caso de actividad de una Botnet se realice una pericia sobre el host afectado para extirpar el Bot que en este pueda residir.

Adicionalmente cabe aclarar que este proceso no estaría completo sin la actualización periódica de las firmas de Snort. Dichas firmas vienen autocontenidas en archivos del tipo .rules que se pueden bajar desde el repositorio oficial de Snort (www.snort.com). Para obtener las últimas versiones de las firmas desde el repositorio oficial se debe pagar una mensualidad pero se pueden obtener gratis las firmas con más de 1 mes de antigüedad desde dicho repositorio. Más allá de este hecho, cabe destacar que el proceso de recambio de firmas se puede realizar manualmente o mediante la utilización un programa llamado OikMaster que automatiza este proceso; hay muchos libros y papers que explican cómo utilizar este programa, por lo cual la instalación y utilización queda fuera del alcance de este trabajo.

Un modelo iterativo de detección de Botnets y mejora de reglas

El proceso presentado anteriormente permite detectar tráfico asociado a Botnets utilizando las reglas de Snort y también actualizar las reglas del mismo para estar “al día” de los nuevos ataques; más allá de esto, es estático. Esto quiere decir que no permite un crecimiento de las reglas de detección basándose en las características únicas de cada entorno. El modelo que a continuación se presenta es evolutivo, dado que permite personalizar y adaptar nuevas reglas de Snort según las características del entorno. Este modelo evolutivo no reemplaza sino que complementa al “Proceso de detección de Botnets con Snort” presentado anteriormente.

Para poder configurar este modelo se necesitan dos herramientas adicionales a Snort (y el resto de los componentes comentados en el Anexo A), a saber: NTOP y Wireshark; optativamente podría utilizarse también un Firewall para bloquear tráfico. Estas dos herramientas se deben instalar en el mismo host en donde se instaló Snort; cabe recordar que la infraestructura de red debe estar configurada para que a dicho host le lleguen todos los paquetes de red. Por la longitud del proceso de instalación y configuración de NTOP y Wireshark, se incluyó este proceso en una sección aparte denominada Anexo B.

Una vez completada la instalación y configuración de todos los sistemas descritos en el Anexo A y en el Anexo B, y previo a la aplicación del modelo iterativo, hay que realizar un trabajo de normalización del tráfico con NTOP. Dicho trabajo implica detectar e identificar cuáles son los protocolos que circulan por la red en forma corriente para generar un patrón de comportamiento “normal” de la red; la idea es que todo tráfico que salga de ese patrón sea postulado como sospechoso en forma inicial hasta que se pruebe lo contrario. Este no es un proceso estático, sino que debe irse adaptando a medida que avanza el conocimiento que el operador tiene sobre la red y su tráfico; el mismo modelo iterativo aquí descripto ayuda a tratar con esta evolución del tráfico.

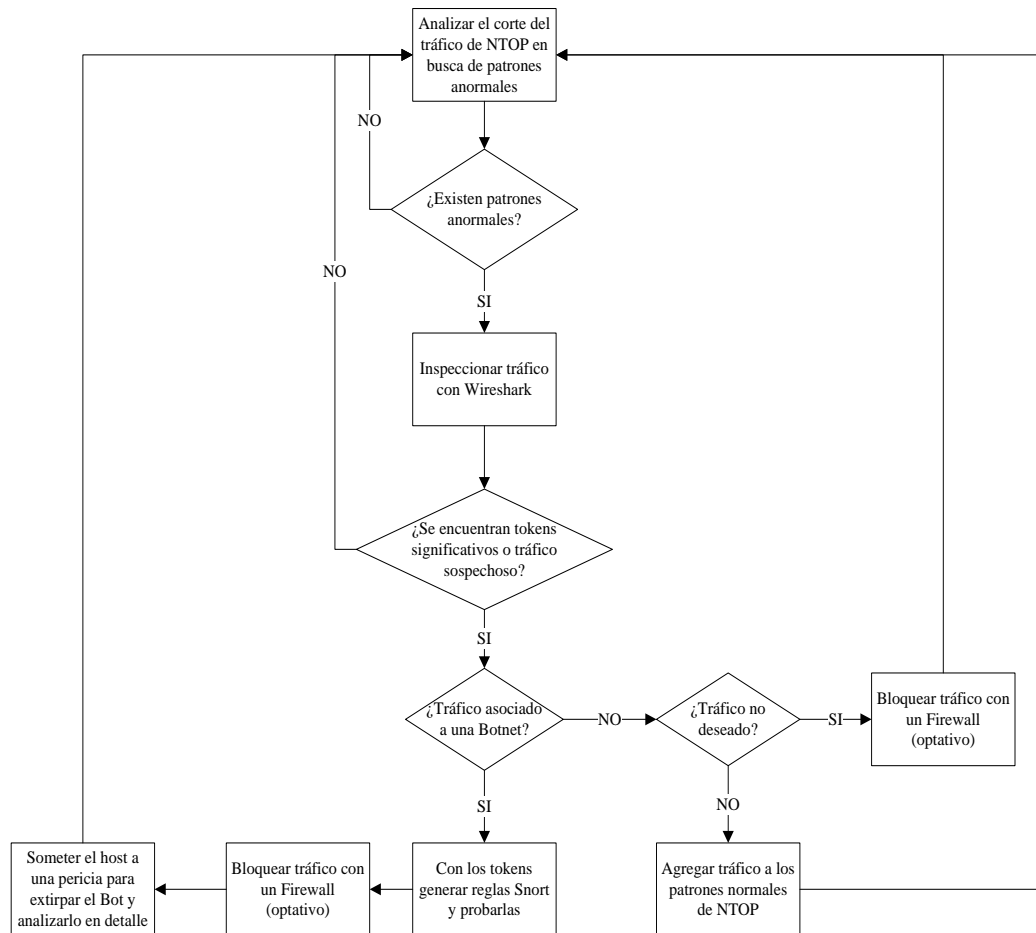
Para implementar esta normalización de tráfico de red se puede utilizar el parámetro “--protocols” cuando se ejecuta NTOP como se menciona en el Anexo B; todo tráfico no normalizado aparecerá en “Other IP” cuando se ingrese al menú IP -> Summary -> Traffic como se aprecia a continuación:

FTP	HTTP	DNS	Telnet	NBios-IP	Mail	DHCP-BOOTP	SNMP	NNTP	NFS/AFS	VoIP	X11	SSH	Gnutella	Kazaa	WinMX	DC++	eDonkey	BitTorrent	Messenger	Other IP
0	926.4 KBytes	32.5 KBytes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.8 KBytes
0	710.9 KBytes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	18.1 KBytes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50.4 KBytes
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35.2 KBytes

Cabe aclarar que el hecho de que un determinado tráfico esté asociado a un tipo de protocolo que circula por un puerto estandarizado, no quiere decir que entre dicho tráfico lícito no figure otro ilícito, es decir que por ejemplo en un puerto asociado al protocolo Telnet no circule tráfico IRC de una Botnet. Es por ello que se debe controlar el tráfico de todos los puertos, prestando especial atención al tráfico que aparezca en la categoría “Other IP”.

Una vez normalizado el tráfico, queda como opción del Administrador de Redes/Operador si quiere bloquear con un Firewall todo el resto del tráfico no estandarizado o bloquear solo una parte y ver cómo se comporta la red; estas decisiones pueden también estar ligadas a políticas organizacionales.

Realizado el paso anterior, se puede proceder a detectar anomalías y en base a eso generar nuevas reglas para Snort; para ello se propone el siguiente modelo iterativo que se materializa en el consiguiente procedimiento:



Conceptualmente el procedimiento es sencillo, implica pasar por una serie de etapas y procesos decisorios que desembocarán en una o más acciones.

El primer paso instruye a “Analizar el corte del tráfico de NTOP en busca de patrones anormales”. Pero ¿Qué es un patrón anormal? Lamentablemente esta pregunta admite varias respuestas y varía según el tráfico de cada red en donde el sistema se implemente. Este en sí es un proceso heurístico que requiere práctica del operador que lleve a cabo el proceso. Algunas de las mejores prácticas son:

- Analizar especialmente el tráfico en la categoría “Other IP”.
- Analizar los hosts de Internet a los que se conectan los hosts de la Intranet en busca de nombres sospechosos.
- Analizar los hosts de Internet con mayor volumen de tráfico (pueden ser posibles servidores de Botnets).
- Analizar cambios bruscos en el tráfico.








- Comparar el tráfico contra una lista negra de hosts y puertos asociados a Botnets.
- Analizar el tráfico en los puertos 6660-6669, 7000 y 113 que están vinculados a actividades de Bots IRC.
- Controlar el tráfico asociado a SMTP que puede evidenciar un Bot enviando spam.
- Controlar el tráfico asociado al File Sharing de Windows (puertos 135, 139 y 445) que puede evidenciar una Botnet instalada en la Intranet tratando de expandirse.
- Prestar atención a alertas de Snort evidenciando escaneo de puertos ya que es una actividad típica de las Botnets.

A continuación se provee un ejemplo, si de un análisis surgiese que el host con el nombre irc.ubuntu.com es en realidad un Bot instalado en un ordenador de la Intranet transmitiendo comandos de IRC, el operador puede adentrarse en el mismo para recabar más información.



Host	Domain	Data Descending order, click to reverse	FTP	HTTP	DNS	Telnet	NBios-IP	Mail	DHCP-BOOTP	SNMP	NNTP
192.168.1.121 Low Risk		1.2 MBytes 52.4 %	0	943.2 KBytes 36.2 KBytes	0	0	0	0	0	0	0
ar.archive.ubuntu.com OS: Windows		710.9 KBytes 31.2 %	0	710.9 KBytes	0	0	0	0	0	0	0
irc.ubuntu.com		211.9 KBytes 9.3 %	0	0	0	0	0	0	0	0	0
devinlaptop [NetBIOS]		78.1 KBytes 3.4 %	0	0	0	0	18.5 KBytes	0	0	0	0
239.255.255.250		44.3 KBytes 1.9 %	0	0	0	0	0	0	0	0	0
feeds.bbc.co.uk		12.3 KBytes 0.5 %	0	12.3 KBytes	0	0	0	0	0	0	0
ns1.telecentro.com.ar		8.9 KBytes 0.4 %	0	0	8.9 KBytes	0	0	0	0	0	0
security.ubuntu.com	Flag for gTLD code com (Guessing from gTLD)	6.7 KBytes 0.3 %	0	6.7 KBytes	0	0	0	0	0	0	0
ocsp.verisign.com		3.4 KBytes 0.2 %	0	3.4 KBytes	0	0	0	0	0	0	0
evsecure-ocsp.verisign.com		3.2 KBytes 0.1 %	0	3.2 KBytes	0	0	0	0	0	0	0

Para ello ingresa al link anteriormente marcado y puede obtener información del host:

Info about [irc.ubuntu.com](#)

IP Address	86.65.39.15 <small>Flag for gTLD code com (Guessing from gTLD) [unicast] [Purge Asset]</small>
First/Last Seen	lun 02 ago 2010 22:36:07 ART - lun 02 ago 2010 22:38:21 ART [Inactive since 3 sec]
Domain	ubuntu.com
Last MAC Address/Router Network Interface Card (NIC)/Router	00:1E:E5:7A:A9:19
OS Name	 [Linux.2.4.20-web100]
Host Location	Remote (outside specified/local subnet)
IP TTL (Time to Live)	54:54 [~10 hop(s)]
Total Data Sent	206.7 KBytes/171 Pkts/0 Retran. Pkts [0%]
Broadcast Pkts Sent	0 Pkts
Data Sent Stats	Local 100 %  Rem 0 %
IP vs. Non-IP Sent	IP 100 %  Non-IP 0 %
Total Data Rcvd	8.5 KBytes/128 Pkts/0 Retran. Pkts [0%]
Data Rcvd Stats	Local 100 %  Rem 0 %
IP vs. Non-IP Rcvd	IP 100 %  Non-IP 0 %
Sent vs. Rcvd Pkts	Sent 57.2 %  Rcvd 42.8 %
Sent vs. Rcvd Data	Sent 96.0 %  Rcvd 4.0 %
Further Host Information	[Whois] [🔗]

Last Contacted Peers

Sent To	IP Address	Received From	IP Address
192.168.1.121 	192.168.1.121	192.168.1.121 	192.168.1.121
Total Contacts	1	Total Contacts	2


TCP/UDP - Traffic on Other Ports

Client Port	Server Port
	• 8001

TCP/UDP Recently Used Ports

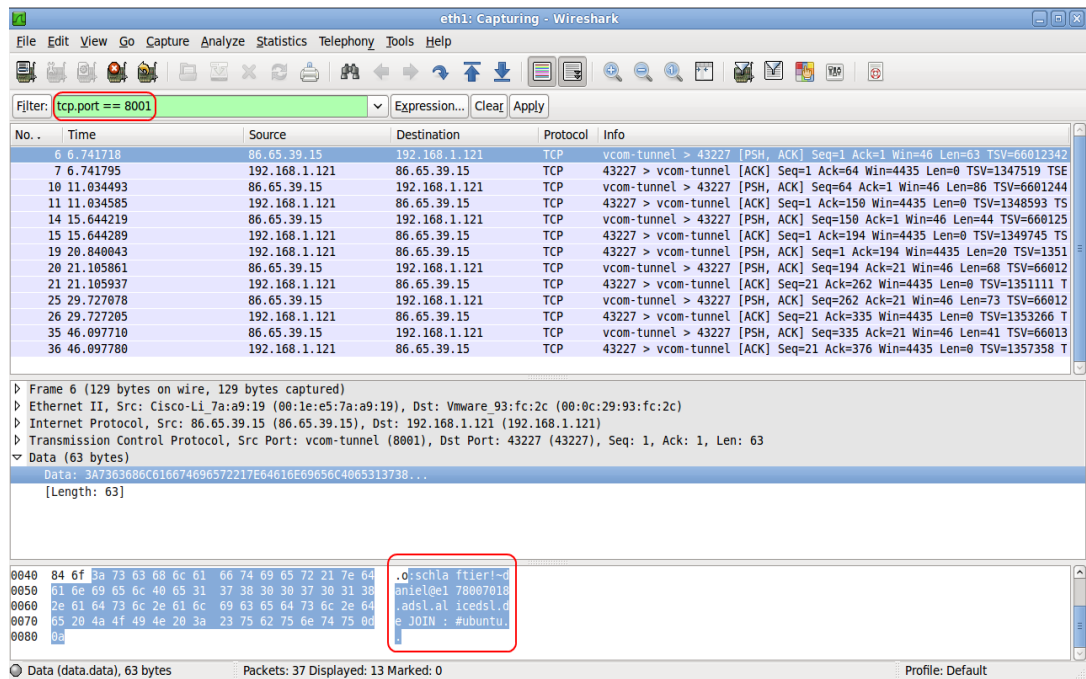
Client Port	Server Port
	• 8001

18 Active TCP/UDP Sessions

Client	Server	Data Sent	Data Rcvd	Active Since	Last Seen	Duration	Inactive	Latency	L7 Proto	Note
192.168.1.121 Low Risk :43227	irc.ubuntu.com  :8001	8.4 KBytes	208.4 KBytes	lun 02 ago 2010 22:36:07 ART	lun 02 ago 2010 22:40:27 ART	4:20	4 sec			SYN ACK PUSH

En este caso se puede observar que se produjo una conexión al puerto 8001 del host irc.ubuntu.com desde el host 192.168.1.121 de la Intranet y que aún la conexión se encuentra activa.

Si hay razones para creer que el patrón descubierto es anormal, llega el momento de analizar con Wireshark el tráfico asociado al puerto donde se está produciendo el tráfico con el patrón anormal. Wireshark otorga una visión mucho más precisa del tráfico de red. Siguiendo el ejemplo anterior se debería filtrar el tráfico al menos al puerto 8001 y opcionalmente también a la IP 192.168.1.121 para ver la situación. Wireshark tiene una sintaxis específica para realizar estas acciones.



En este caso del análisis del payload surge que hay un comando “JOIN” que como anteriormente en este trabajo se mencionó en “Introducción al protocolo IRC” está asociado al proceso de asociación de un cliente (posiblemente Bot) a un canal IRC.

Este análisis es también heurístico, de forma que no hay un camino determinado a seguir. Algunas recomendaciones son:

- Para Bots de IRC buscar comando específicos de IRC.
- Tener en cuenta que a veces los Bots cambian los comandos estándar de IRC por otros nombres o utilizan puertos distintos a los tradicionales para evitar detección.
- Algunos BotHerders toman paquetes de Bots y los modifican un poco, por lo cual un conocimiento de los Bots más populares es recomendable.
- Si se observa tráfico encriptado (o no legible) es posible que el Bot esté utilizando tecnología de cifrado para evitar detección.
- Se pueden buscar tokens que evidencien transferencia de información (videos, mp3, etc) para Bots tipo P2P.

- Se pueden buscar tokens que evidencien escaneos de la red como por ejemplo scan <IP> o símil lo cual es una actividad común de los Bots.

Luego de esta evaluación habrá que determinar si el tráfico posee tokens significativos o al menos es sospechoso y en dicho caso ver si está asociado a una Botnet siguiendo las recomendaciones anteriores.

De determinarse que el tráfico está asociado a una Botnet hay que generar una nueva regla para Snort con dicho tráfico (véase Comprendiendo las Reglas de Snort), someter el host a una pericia para extirpar el Bot y opcionalmente bloquear el tráfico con un Firewall.

En el caso del ejemplo anterior una regla Snort podría ser la siguiente:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 8001 (msg:"Bot desconocido haciendo JOIN en el puerto 8001"; classtype:misc-attack; Content:"JOIN"; nocase; sid:6309; rev:1;)
```

Sobre las reglas también se pueden dar algunas recomendaciones.

- Las reglas deben estar limitadas a un rango de puertos específico o a algunos puertos en especial para evitar sobrecargar al motor de Snort.
- Las reglas también pueden estar dirigidas a una IP en particular, aunque no es del todo necesario.

Finalmente si el tráfico no estuviese asociado a una Botnet habría que evaluar si es deseado o no. Si lo fuera convendría agregarlo al tráfico reconocido por NTOP como normal y en caso contrario se lo podría bloquear con un Firewall.

Luego, el proceso empieza nuevamente en busca de su próxima iteración.

Capítulo 2: Detección de Botnets con Nepenthes

Introducción a los Honeypots [8] [14] [16]

Los Honeypots son recursos que mienten sobre su verdadero fin; pueden bien considerarse como cebos o trampas informáticas. El principal objetivo de un Honeypot es “tentar” a un atacante (automatizado o no) para que se conecte a este y realice una serie de acciones para así poder determinar y estudiar su comportamiento.

Se puede clasificar a los Honeypots desde dos perspectivas diferentes, a saber: desde su basamento físico y desde su grado de interacción.

Se denomina basamento físico al hardware en el cual el Honeypot se ejecuta, este puede ser un equipo físico, es decir una computadora con su hardware dedicado al Honeypot o bien una máquina virtual; en este último caso se habla de Virtual Honeypots.

Con respecto al grado de interacción puede decirse que los Honeypots se dividen en Honeypots de alta y de baja interacción. Un Honeypot de alta interacción provee una emulación completa del sistema víctima, es decir, que es un sistema operativo completo que el atacante podrá comprometer en su totalidad. Por su parte un Honeypot de baja interacción provee solamente una emulación parcial del sistema víctima; en este caso la emulación estará centrada en los aspectos vitales necesarios para que el sistema víctima pueda ser comprometido y se pueda extraer suficiente información de dicho hecho.

Introducción a Nepenthes [15] [17]

Nepenthes es una solución tipo Honeypot pensada para recolectar malware de forma automática (sin intervención humana alguna). Para lograr este objetivo realiza una emulación de determinados servicios vulnerables que escuchan en determinados puertos de red a los cuales se conectará el malware y donde será capturado. El objetivo de Nepenthes es recolectar malware para aprender sobre su comportamiento y distribución.

Nepenthes por definición se centra en el malware que se esparce en la red (especialmente Internet) de forma automática, denominado Worm (o gusano). Dado que como ya se mencionó, una Botnet es en realidad un Worm con un sofisticado mecanismo de control remoto (ver “Conceptualización de una Botnet como un Worm”), Nepenthes es una solución perfectamente aplicable para la detección e incluso erradicación de Botnets.

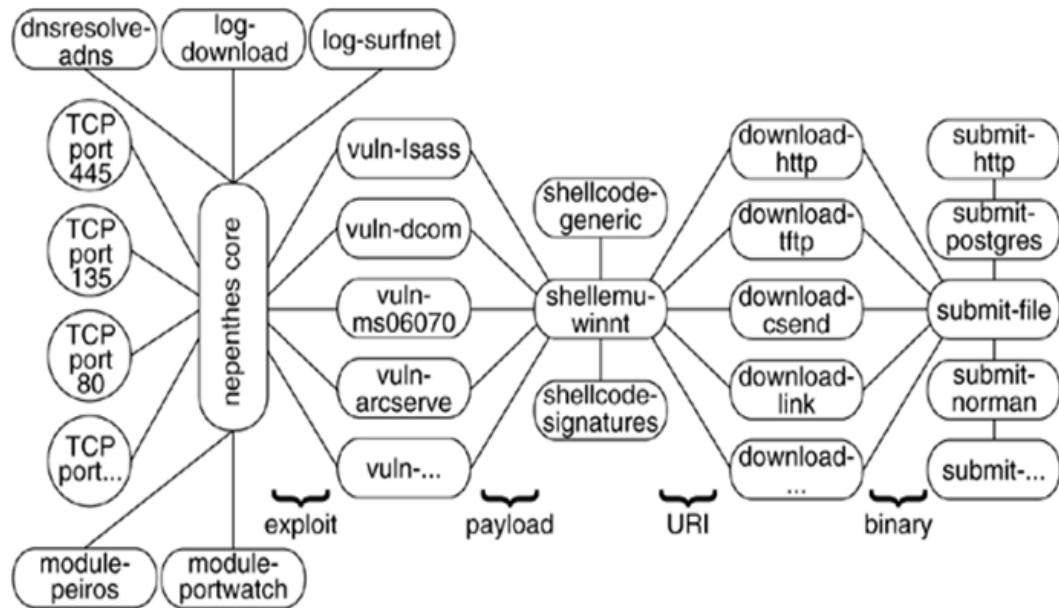
A diferencia de un IDS, Nepenthes no requiere “ver” todo el tráfico de la red para funcionar; solo se lo coloca en un host o más y se queda a la espera que el malware lo impacte.

Debido a la naturaleza de su funcionamiento, este Honeypot no es capaz de detectar troyanos que vengan por ejemplo en un pendrive y que ejecute un usuario o virus que requieren interacción humana para copiarse entre ordenadores o rootkits. Adicionalmente, en caso de enfrentarse a un atacante humano (hacker), los resultados pueden ser variables dependiendo de las acciones de este; pero en general se puede afirmar que el sistema está pobremente preparado para enfrentarse a un atacante de este tipo por lo que el hacker podría descubrir fácilmente la trampa.

Puede concluirse que Nepenthes es una solución Honeypot que se enfoca en la recolección y el análisis del malware tipo Worm y/o Bot que circula por Internet o en determinadas condiciones dentro de la Intranet.

Diseño arquitectónico de Nepenthes [15] [17]

Nepenthes se diseñó de forma modular con un núcleo que gestiona la interfaz de red en la cual escucha el Honeypot y que coordina a un conjunto de módulos periféricos. Una forma de visualizar el funcionamiento del Honeypot en detalle es mediante el siguiente gráfico obtenido del libro Virtual Honeypots (ver Bibliografía):



Actualmente Nepenthes maneja los siguientes módulos:

- Módulos de simulación de vulnerabilidades: emulan un conjunto de servicios vulnerables en determinados puertos prefijados.
- Módulos de parseo de shellcode: analizan el payload que inyecta el malware al Honeypot luego de atacar exitosamente el servicio vulnerable simulado.
- Módulos de captura: recogen las credenciales y las direcciones remotas provistas por los módulos de parseo de shellcode para bajar el resto del malware.
- Módulos de sumisión: bajan el malware a una ubicación configurable y (opcionalmente) lo envían a una página web de análisis de malware.

- Módulos de logueo: obtienen información del proceso de emulación para poder observar patrones comunes del malware recolectado.

Ahora que se conoce la funcionalidad básica de cada uno de los principales módulos de Nepenthes, es mucho más sencillo describir el gráfico previamente presentado. De hecho, este gráfico puede interpretarse si se lo lee de izquierda a derecha: El núcleo de Nepenthes se encuentra rodeado por una serie de módulos periféricos y presenta una serie de puertos de red abiertos detrás de los cuales yacen módulos que emulan determinadas vulnerabilidades fácilmente explotables. Una vez que un exploit compromete satisfactoriamente uno (o más) de los módulos vulnerables este inyectará un payload (conjunto de sentencias) que normalmente instruirá al malware a bajar el resto de sí desde una ubicación remota y ejecutarse para continuar el proceso infeccioso; estos payloads normalmente constan de un conjunto de instrucciones de consola las cuales Nepenthes recibe mediante sus Módulos de parseo de shellcode. Una vez recibido el payload completo, Nepenthes intentará obtener las credenciales y la dirección desde donde bajar el binario del malware y lo intentará bajar y almacenar en una ubicación determinada mediante sus Módulos de captura y sumisión; estos últimos módulos también otorgan la posibilidad de enviar el binario a una ubicación remota como una Sandbox web para aprender más de su comportamiento.

Despliegue e instalación del sensor [15] [17]

Al igual que ocurría con Snort, en el caso de Nepenthes también es muy importante la ubicación del sensor para conseguir los resultados esperados. Si lo que se desea es conocer el tráfico malicioso que circula por Internet entonces el sensor debe estar en contacto directo con esta red; en cambio si lo que se desea es conocer el posible malware que puede estar circulando por la Intranet, el sensor debe estar colocado dentro de alguna de las subredes organizacionales.

Dado que el malware tipo worm/bot trata normalmente de expandirse agresivamente explorando hosts vulnerables y explotando vulnerabilidades en ellos, si se coloca Nepenthes en uno (o más) hosts de la Intranet, es ampliamente probable que el malware impacte tarde o temprano el Honeypot. De esta forma, no hace falta que host donde reside el sensor tenga acceso a todo el tráfico de red, sino que el mismo se instale en uno o más hosts y quede a la espera de que el malware lo impacte, lo cual es cuestión de probabilidades y suerte. Para evitar estar instalando múltiples sensores, Linux permite que un solo sensor Nepenthes pueda tomar múltiples IP's facilitando el trabajo del despliegue; sin embargo esta configuración excede el alcance del presente trabajo.

La instalación y configuración de Nepenthes es relativamente sencilla y se encuentra especificada en el "Anexo C".

Testing inicial de Nepenthes [15] [17]

Una vez que el Honeypot se encuentra instalado y sus módulos adecuadamente configurados (lo cual como se vio no es estrictamente necesario ya que funciona con la instalación por defecto) se puede proceder a testear el correcto funcionamiento del sensor; para ello se comienza con la ejecución de Nepenthes:

```
# sudo /usr/sbin/nepenthes --user=nepenthes --group=nepenthes -C -D
```

Como se puede ver, ahora el demonio de Nepenthes se encuentra ejecutándose:

```
# ps -ef | grep -i nepenthes
```

```
113      2705      1  0 16:52 ?          00:00:00 /usr/sbin/nepenthes --
user=nepenthes --group=nepenthes -C
```

Ahora que Nepenthes se encuentra ejecutándose podemos ver un listado de los puertos sobre los que simula vulnerabilidades:

```
# netstat -tpan
```

	Protocolo	RecvQ	Sent-Q	Dirección Local	Dirección remota	Estado	PID/Program name
	tcp	0	0	0.0.0.0:1025	0.0.0.0:*	ESCUCHAR	3442/nepenthes
	tcp	0	0	0.0.0.0:993	0.0.0.0:*	ESCUCHAR	3442/nepenthes
	tcp	0	0	0.0.0.0:995	0.0.0.0:*	ESCUCHAR	3442/nepenthes
	tcp	0	0	0.0.0.0:3140	0.0.0.0:*	ESCUCHAR	3442/nepenthes
	tcp	0	0	0.0.0.0:135	0.0.0.0:*	ESCUCHAR	3442/nepenthes
	tcp	0	0	0.0.0.0:5000	0.0.0.0:*	ESCUCHAR	3442/nepenthes
	tcp	0	0	0.0.0.0:42	0.0.0.0:*	ESCUCHAR	3442/nepenthes
	tcp	0	0	0.0.0.0:139	0.0.0.0:*	ESCUCHAR	3442/nepenthes
	tcp	0	0	0.0.0.0:3372	0.0.0.0:*	ESCUCHAR	3442/nepenthes
	tcp	0	0	0.0.0.0:110	0.0.0.0:*	ESCUCHAR	3442/nepenthes
	tcp	0	0	0.0.0.0:143	0.0.0.0:*	ESCUCHAR	3442/nepenthes

```
tcp    0      0  0.0.0.0:80      0.0.0.0:*ESCUCHAR  3442/nepenthes
tcp    0      0  0.0.0.0:10000  0.0.0.0:*ESCUCHAR  3442/nepenthes
```

Para testear que Nepenthes funciona correctamente solo hay que intentar una conexión contra alguno de estos puertos a la IP donde se encuentra escuchando Nepenthes. Por ejemplo se puede hacer un ftp al puerto 21:

```
C:\> ftp 192.168.1.124
```

```
Conectado a 192.168.1.124.
```

```
220 ---freeFTPd 1.0---warFTPd 1.65---
```

```
Usuario (192.168.1.124:(none)): jdevin
```

```
331 User OK, Password required
```

```
Contraseña:
```

```
530 Authentication failed, sorry
```

```
Error al iniciar la sesión.
```

```
ftp> quit
```

```
C:\>
```

Como se aprecia, Nepenthes se encuentra funcionando y simula el servidor de FTP freeFTPd en su versión 1.0; cualquier conjunto de usuarios/contraseñas será rechazado porque la idea aquí es que un Worm explote alguna vulnerabilidad conocida de este servicio para que luego inyecte su payload.

Descripción del funcionamiento básico de Nepenthes [15] [17]

Ahora que Nepenthes se encuentra correctamente instalado, el sensor está ubicado en el lugar adecuado de la red, se han configurado correctamente sus módulos y se ha corroborado su correcto funcionamiento, es hora de describir su funcionamiento básico.

Nepenthes loguea todo lo que ocurre en un archivo de log el cual resulta sumamente útil para la pericia posterior; por defecto este se encuentra en `/var/log/nepenthes.log`

Si Nepenthes es explotado exitosamente en alguna de las vulnerabilidades que maneja, y baja algún archivo relacionado con el malware, se podrá encontrar más información en los siguientes archivos:

- `/var/log/nepenthes/logged_downloads`
- `/var/log/nepenthes/logged_submissions`

Por su parte los archivos binarios bajados se pueden encontrar en el directorio `/var/lib/nepenthes/binaries`.

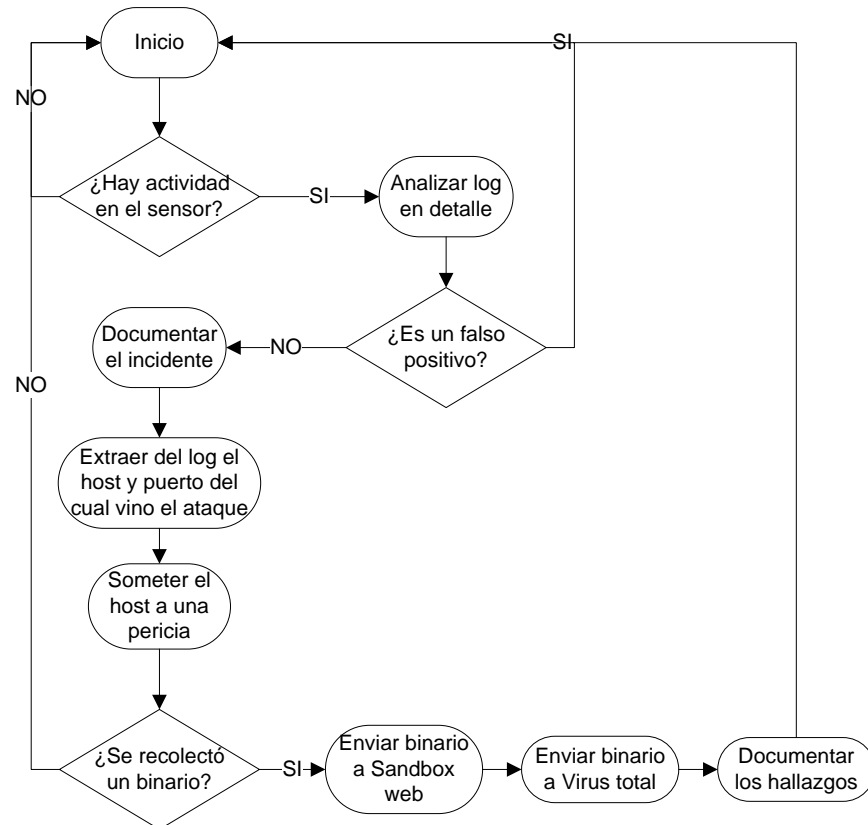
En caso de que el exploit de un malware determinado falle, Nepenthes va a arrojar un hexdump que después podrá ser analizado para ver la razón del fallo; dichos hexdumps se pueden hallar en el directorio `/var/lib/nepenthes/hexdumps`.

En resumen estos son los archivos y directorios más importantes para el Honeypot Nepenthes:

Archivo / Directorio	Descripción
<code>/var/log/nepenthes</code>	Contiene los archivos de configuración
<code>/var/lib/nepenthes</code>	Contiene binarios bajados y <u>hexdump</u> de <u>exploits</u> fallidos
<code>/usr/share/doc/nepenthes</code>	Documentación
<code>/usr/sbin/nepenthes</code>	Ejecutable de <u>Nepenthes</u>
<code>/usr/lib/nepenthes</code>	Contiene los binarios de los módulos de <u>Nepenthes</u>
<code>/etc/nepenthes</code>	Contiene los archivos de configuración de los módulos de <u>Nepenthes</u>
<code>/etc/init.d/nepenthes</code>	Archivo de configuración que ejecuta <u>Nepenthes</u> al iniciarse el sistema

Procedimiento para la operación de Nepenthes

A continuación se expone un diagrama procedimental sobre una forma de operar este Honeypot para alcanzar los mejores resultados en cuanto a la detección de Botnets.



Periódicamente el administrador debe revisar el sensor para ver si se registra nueva actividad; para ello bien podría servirse de analizar cualquier variación del archivo `/var/log/nepenthes.log`. Este análisis puede automatizarse en entornos UNIX mediante un cron que determine cambios en el archivo mencionado.

En caso de encontrar actividad se deberá realizar un análisis del log en detalle para determinar se trata de un falso positivo; los falsos positivos en este caso se refieren a cualquier cosa que altere el archivo de log sin que signifique una intrusión real o intento de ella. Ejemplos de esto son un usuario que se conecta a un servicio del Honeypot por error o un escaneo de puertos por parte del algún administrador. Debe tomarse en cuenta que los falsos positivos sobre los sistemas Honeypots de recolección de malware de este tipo son muy poco comunes porque para “excitarse” el sensor necesita

de un malware real que explote alguna de las vulnerabilidades que este emula.

A modo de ejemplo un fragmento de una posible intrusión podría ser:

```
05122010 14:21:08 spam net handler] Socket TCP (accept)
192.168.133.133:33175 -> 192.168.133.132:443 clearing DialogueList (1
entries)
```

```
[05122010 14:21:08 spam net handler] Removing Dialogue "IISDialogue"
```

```
[05122010 14:21:08 warn module] Unknown IIS SSL exploit 0 bytes State 0
```

```
[05122010 14:21:08 spam mgr event] <in virtual uint32_t
nepenthes::EventManager::handleEvent(nepenthes::Event*)>
```

```
[05122010 14:21:08 spam mgr event] <in virtual uint32_t
nepenthes::EventManager::handleEvent(nepenthes::Event*)>
```

Como se puede apreciar la intrusión se encuentra catalogada como un exploit para IIS; también se puede identificar el host y el puerto desde donde dicha intrusión provino, lo que es especialmente útil cuando no se sabe si se trata o no de una intrusión real.

Si se trata de una intrusión real se deberá documentar el incidente y extraer el host y el puerto del cual vino el ataque para identificar cual es el host infectado de la Intranet y someterlo a una pericia.

Finalmente es posible que Nepenthes logre recolectar el binario del malware cuyo payload recibe; para determinar si alguna de las intrusiones permitió que Nepenthes obtuviera un binario se debe revisar el directorio binaries en /var/lib/nepenthes/binaries (también llamado hexdumps en algunas instalaciones) como se ve en la siguiente captura:

```
[root@localhost hexdumps]# ls
061a2b0a815d9e8e94de1c6c58454e09.bin  77b57cdb87a3b1606a04fcd389e5cea3.bin
07727088d9f9a7a0f49b86c2afbc5057.bin  7b05b77d4e47fe46999f91cc5ea05ace.bin
12f78d3dd0244ee45936d6a3f280dbf4.bin  7fc37ff3e67797a0943fdd094af471b4.bin
198ca24c849f4c8c157cc2dbd22cd433.bin  8293677d52b96fbba4d051dba9cd3d41.bin
1a2208dfed3c875bf5a8a4e825a84cb0.bin  86ce63631c462004419392f57b650423.bin
1be25e278c7709fa554bdab609c974e3.bin  924310d3efb4075220b417e0bf2e3583.bin
2c223a01c305635bdcbf53194abe835.bin  9e2dcb186123105d4117afcc35e62924.bin
304bd501a7f0181363e74d02035dae83.bin  a276a5b0740c65f2d4edc69b54ea50b8.bin
33151a694b0738864897b1efb6babd05.bin  a8505fc508818ba30e183a2c6d4d461a.bin
4010912894614be79fa7b433fb9fd731.bin  ab1de38606d8a133e799e7afb7646ed0.bin
4049775a0036ec53fa39aba787fd0b24.bin  baac5dcac9d9003356614296d6946728.bin
4faa48136da471ec60be996a83b7cf6b.bin  cff7c8445c9e67c823394860a070423a.bin
5068459797375d0e96145f43508b4305.bin  e6b072a19c435c872a85b8ca08f68410.bin
6b730c468f590b187f7e59a9acececab.bin  ed7c99952875d43cae68761ed215ac3.bin
6f19c36c71de3b45b9a61d95e35df511.bin  f3cf7fb5518a29ee6dc36ed38c6b5634.bin
[root@localhost hexdumps]# _
```

Adicionalmente se puede consultar el log de binarios de malware bajados para ver la bitácora en el registrada (/var/log/nepenthes/logged_downloads):

```
019a8 , 0x00000010).
[ spam ] Stored Hexdump var/hexdumps/77b57cdb87a3b1606a04fcd389e5cea3.bin (0x0a0
019a8 , 0x0000001a).
[ warn module ] Unknown exploit 0 bytes
[ module ] Ignoring zero-length hexdump.
[ warn module ] Unknown WatchDialogue 0 bytes, port 143
[ module ] Ignoring zero-length hexdump.
[ warn dia ] Unknown IIS 7059 bytes State 2
[ dia ] Stored Hexdump var/hexdumps/304bd501a7f0181363e74d02035dae83.bin (0x0a0
02a30 , 0x00001b93).
[ warn dia ] Unknown ASN1_SMB Shellcode (Buffer 88 bytes) (State 0)
[ dia ] Stored Hexdump var/hexdumps/a8505fc508818ba30e183a2c6d4d461a.bin (0x0a0
00118 , 0x00000058).
[ warn module ] Unknown PNP Shellcode (Buffer 88 bytes) (State 0)
[ module ] Stored Hexdump var/hexdumps/a8505fc508818ba30e183a2c6d4d461a.bin (0x0
9fffc78 , 0x00000058).
[ warn module ] Unknown LSASS Shellcode (Buffer 88 bytes) (State 0)
[ module ] Stored Hexdump var/hexdumps/a8505fc508818ba30e183a2c6d4d461a.bin (0x0
9fff7e8 , 0x00000058).
[ warn handler dia ] Unknown DCOM Shellcode (Buffer 88 bytes) (State 0)
[ handler dia ] Stored Hexdump var/hexdumps/a8505fc508818ba30e183a2c6d4d461a.bin
(0x09fff380 , 0x00000058).
[ warn module ] Unknown WatchDialogue 0 bytes, port 143
[ module ] Ignoring zero-length hexdump.
```

Finalmente se deberá enviar el binario malicioso a un Sandbox web para realizar una ingeniería inversa automatizada (este proceso puede automatizarse, véase “Anexo C”) y a Virus total () y documentar los resultados de los hallazgos.

Un excelente Sandbox web es Norman Sandobox

(http://www.norman.com/security_center/security_tools/) cuya salida se muestra a modo de ejemplo para un posible binario malicioso:

Técnicas y herramientas forenses para la detección de Botnets

Your message ID (for later reference): 20060225-1370

```
nepenthes-3799434f10827ae73deb41faf72ba6ae-svhostcs32.exe :
[SANDBOX] contains a security risk - W32/Spybot.gen3 (Signature:
W32/Spybot.JZZ)
[ General information ]
  * **IMPORTANT: PLEASE SEND THE SCANNED FILE TO:
ANALYSIS@NORMAN.NO - REMEMBER TO ENCRYPT IT (E.G. ZIP WITH
PASSWORD)**.
  * **Locates window "NULL [class mIRC]" on desktop.
  * File length:          89600 bytes.
  * MD5 hash: 3799434f10827ae73deb41faf72ba6ae.

[ Changes to filesystem ]
  * Creates file C:\WINDOWS\SYSTEM32\svhostcs32.exe.
  * Deletes file 1.

[ Changes to registry ]
  * Creates value "Windows Update System Shell"="svhostcs32.exe"
in key "HKLM\Software\Microsoft\Windows\CurrentVersion\Run".
  * Creates value "Windows Update System Shell"="svhostcs32.exe"
in key "HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices".
  * Creates value "Windows Update System Shell"="svhostcs32.exe"
in key "HKCU\Software\Microsoft\Windows\CurrentVersion\Run".
  * Sets value "restrictanonymous"="" in key
"HKLM\System\CurrentControlSet\Control\Lsa".

[ Network services ]
  * Looks for an Internet connection.
  * Connects to "test.chung.li" on port 80 (TCP).
  * Connects to IRC Server.

[ Security issues ]
  * Possible backdoor functionality [Authenticate] port 113.

[ Process/window information ]
  * Creates a mutex chimera.
  * Will automatically restart after boot (I'll be back...).
  * Enumerates running processes.

[ Signature Scanning ]
  * C:\WINDOWS\SYSTEM32\svhostcs32.exe (89600 bytes) :
W32/Spybot.JZZ.
```

Con respecto a enviar el binario a Virus total se puede decir que este portal permite escanear el binario con muchos antivirus a la vez, lo cual claramente ayuda a su clasificación. Este portal presenta una interfaz web para subir los binarios y luego presenta un informe similar al que puede visualizarse a continuación:

Ing. Juan A. Devincenzi



Capítulo 3: Ourmon como herramienta de detección de Botnets

Introducción y arquitectura [18]

Ourmon se enmarca dentro de los escáneres de red (al igual que Snort). En realidad Ourmon nació como una herramienta para realizar análisis del tráfico de red para realizar diagnósticos de errores o situaciones problemáticas y a posteriori evolucionó para permitir la detección de Botnets; justamente esta última característica separa a Ourmon del resto de las herramientas que hasta aquí se presentaron... Ourmon se encuentra pensada específicamente para la detección de Botnets a diferencia de Snort y Nmap que también admiten otros usos.

Esta herramienta monitoriza el tráfico de red y es capaz de producir reportes de la actividad que en la misma acontece ya sea en forma de gráficos y/o tablas. La herramienta de por sí misma no interpreta los reportes, ni tiene reglas para comparar el tráfico contra ciertos patrones determinados como Snort; en cambio genera solamente algunas sugerencias de la actividad de red que procesa y muestra dicha actividad al operador y luego corresponde a este el determinar si el tráfico se corresponde con una actividad potencialmente sospechosa o no.

Arquitectónicamente Ourmon se encuentra compuesto por 2 módulos bien diferenciados, a saber: probe y graphics engine. El probe se encarga de recoger los paquetes de red desde el buffer del kernel y procesarlos para obtener de ellos información estadística. El graphics engine (o motor gráfico) se alimenta de la salida que produce el módulo probe y con ella produce funciones gráficas de tráfico en el tiempo, reportes ASCII y una variedad de eventos de log.

Instalación y despliegue de la herramienta [18]

El sistema Ourmon fue pensado para funcionar en sistemas operativos tipo BSD pero se adapta perfectamente a su uso en numerosas variantes de Linux. En este caso se presentará la instalación de Ourmon en Ubuntu. El proceso en si es bastante sencillo y directo; primero hay que satisfacer varias dependencias y luego se ejecuta un script de Perl que termina de realizar el proceso de instalación de Ourmon en sí.

Al momento de realizar el despliegue de Ourmon, dado que es un escáner de red (al igual que Snort) las consideraciones a tener en cuenta son las mismas que al momento de desplegar Snort; es decir, que se debe colocar el sistema en un host por el cual pase todo el tráfico de la red organizacional (o subred) que se desee monitorizar.

Ourmon se puede bajar desde Internet (desde la página <http://sourceforge.net/projects/ourmon/>) como un paquete .tar.gz que luego se puede descomprimir en el directorio de instalación deseado. Se debe tener en cuenta que Ourmon se instalará en el mismo directorio en donde previamente se lo descomprimió.

Una vez descomprimido el programa y previo a la instalación de deben satisfacer las dependencias mediante la instalación de los siguientes componentes:

```
# apt-get install libpcap0.8-dev
```

```
# apt-get install libpcrc3
```

```
# apt-get install libpcrc3-dev
```

```
# apt-get install libjudy-dev
```

```
# apt-get install librrds-perl
```

```
# apt-get install apache2
```

Luego de esto se debe ejecutar el script de instalación de Ourmon, el cual se encontrará en el directorio donde previamente se lo descomprimió:


```
# ./configure.pl
```

El proceso de instalación completo como anteriormente se dijo es sencillo pero extenso, por lo cual se decidió incluirlo en su totalidad en el Anexo D.

Una vez instalada la herramienta, habrá que utilizar un explorador web para ver la interface web de Ourmon que se encontrará por defecto en el puerto 80 del equipo donde se lo instaló.

Cabe aclarar adicionalmente que Ourmon posee dos interfaces distintas, una es la interfaz web (ya mostrada anteriormente) y otra es la interfaz 'cruda' que consiste básicamente en la visualización de una serie de archivos de log con herramientas propias de Linux desde la línea de comandos.

Sobre los reportes [19]

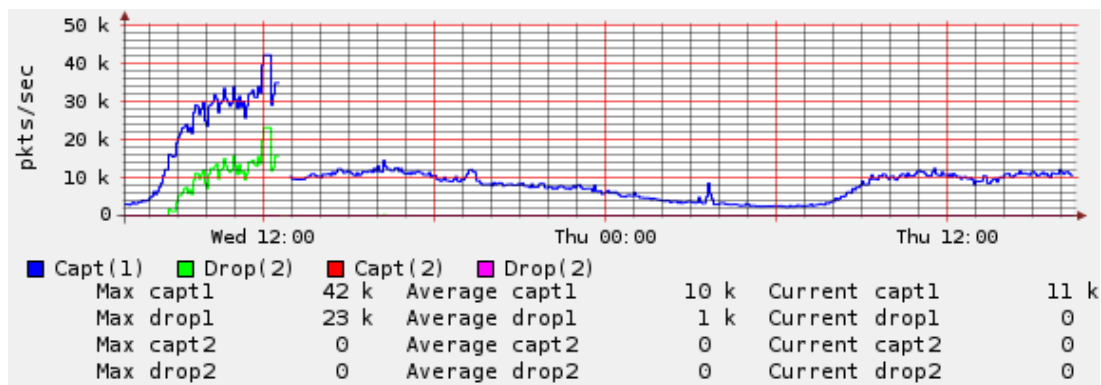
El sistema Ourmon está conformado por una interface web la cual consta de un conjunto de reportes ya sea gráficos o en texto (o en algunos casos en forma gráfica y como texto al mismo tiempo). Los reportes se generan siempre a intervalos regulares de 30 segundos y luego se sumarizan para obtener reportes a lo largo de un intervalo de tiempo más extenso; las principales sumarizaciones son: 1 hora, 1 día, 1 mes, 1 año. Adicionalmente siempre se guardan en línea (web) las sumarizaciones diarias de la última semana que van rotando a medida de que transcurre el tiempo. Cabe aclarar igualmente que no todos los reportes se encuentran disponibles en todas las sumarizaciones posibles siendo esto una limitación misma del sistema.

La idea principal de cualquier reporte es que un operador experimentado pueda al verlos detectar hosts que se comportan como Bots (o worms) o que realizan alguna actividad relacionada con su funcionamiento como realizar un escaneo de puertos o enviar tráfico IRC, etc.

Los reportes principales sobre los que se va a trabajar en este escrito son los siguientes, a saber: Reporte TCP, Reporte UDP, Worm Graph (us vs. them), Reporte IRC, Sysdump, Reporte Email Syn. A continuación se detallará cada uno de estos reportes:

Reporte TCP

Este es uno de los reportes más importantes de Ourmon y viene tanto en versión gráfica como en forma de tabla. Un ejemplo de su versión gráfica (como tráfico TCP en función del tiempo, para un intervalo de tiempo determinado) es la siguiente:



[21]

En forma de tabla el reporte se encuentra compuesto por una serie de columnas que a continuación se detallan:

- IP source adress: IP origen desde la cual proviene el tráfico.
- Work (carga de trabajo): es el porcentaje (que puede variar entre 0 y 100 %) de paquetes de control vistos en un período de tiempo en el tráfico; se entiende por paquete de control a por ejemplo un SYN para iniciar una conexión, un SYN+ACK para responder a esta petición, un RESET, un FIN, etc. Normalmente un host realizando un escaneo de puertos tiene una alta carga de trabajo; en general cualquier host con alta carga de trabajo es al menos sospechoso.
- Flags: las banderas son E (errores en el protocolo ICMP), W (la carga de trabajo de TCP es ≥ 90), w (la carga de trabajo de TCP es ≥ 50), O (la conexión no presenta FIN's), R (la conexión presenta Reset's), y M (la conexión presenta pocos o ningún paquete de datos). Si el host se comporta como un escáner posiblemente los flags sean WOM o EWORM.
- Apps: se hacen sugerencias en base al tráfico visto en la red; las sugerencias son B (BitTorrent protocol), G (Gnutella protocol), K (Kazaa protocol), M (Morpheus o P2P protocol), P (tráfico hacia la red Honeypot), E (se visualizó un puerto de Email), H (se visualizó un puerto Web), I (se vieron mensajes IRC).
- SA/S: es un porcentaje (que varía entre 0 y 100%) del número total de paquetes SYN+ACK divididos por los paquetes SYN; un alto porcentaje

indica que el host probablemente sea un servidor, mientras que un valor bajo sugiere un cliente.

- L3D/L4D: implica la cantidad de IP's y puertos destino con los cuales se comunica un host en particular; es común que los escáneres de puertos tengan un alto valor de L3D y un valor pequeño de L4D (o sea que escanean pocos puertos de muchas IP's).
- L4S/src: información relacionada con los puertos origen de las conexiones de red de un host en un intervalo de tiempo; si por ejemplo el host fuese un server y el valor de este parámetro fuese 1/80 es claro que se trata de un servidor web.
- snt/rcv: conteo de paquetes enviados y recibidos; si un host tiene muchos paquetes enviados y muy pocos (o ninguno) recibidos eso es una pista que se puede tratar de un escáner de puertos.
- Port signatures: es una lista de puertos (ordenada por puerto creciente) de la forma [puerto,% de tráfico] de un sampleo de 10 puertos vistos en la red en un intervalo de tiempo para un host en particular; es especialmente útil cuando el host es sospechado de ser un escáner de puertos puesto que los mismos muchas veces se concentran en determinados puertos en particular asociados a determinados servicios que bien podrían ser vulnerables.

Como corolario de la información antepuesta es interesante generar un ejemplo de la apariencia que tendría en el reporte TCP un host que estuviera actuando como un posible escáner. El reporte en cuestión para el host es el siguiente:

Ip_src	Flags	Apps	Work	SA/S	L3D/L4D	L4S/src	Snt/rcv	Port Signature	Ip_dst
192.168.1.100	WOR		100	0	53/1	Oct-26	89/18	[139,100]	200.22.x.x

Como se puede apreciar según los flags hay una alta carga de trabajo y la conexión presenta muchos RESET's y ningún FIN, lo cual es típico en el port scanning; adicionalmente se aprecia que la carga de trabajo es de un 100%

y que no corresponde a ningún tipo de tráfico en particular identificable por Ourmon (Apps vacío). Por su parte el SA/S sugiere un cliente por ser 0, el L3D/L4D sugiere que se está probando un puerto en muchas IP's y adicionalmente se ve que hay más paquetes enviados que recibidos y que el total de la carga de tráfico está dirigida al puerto 139 donde escucha NetBIOS.

De este ejemplo se puede deducir que el operador se encuentra ante la presencia de un host posiblemente infectado con un Bot (o un Worm).

Reporte UDP

Este reporte es uno de los más significativos junto con el reporte TCP, más allá de esto, este último tiene mayor importancia puesto que la mayoría de los ataques se registran utilizando el protocolo TCP. Al igual que el reporte TCP en este caso el reporte viene tanto en formato gráfico (como tráfico en función del tiempo en un formato muy similar al de TCP) y en forma de tabla cuyos campos se describirán a continuación evitando repetir las descripciones ya anteriormente dadas:

- IP scr.
- UDP_sent: cantidad de paquete UDP enviados en un intervalo de tiempo.
- UDP_recv: cantidad de paquete UDP enviados en un intervalo de tiempo.
- Unreachs: cantidad de paquetes ICMP indicando que el puerto destino no se puede alcanzar; similar al RESET para TCP.
- Weight: carga de trabajo UDP calculada como el producto de los paquetes UDP enviados y los errores ICMP recibidos en un intervalo de tiempo; un valor alto de este parámetro junto con los Unreachs sugiere que el host se puede estar comportando como un escáner de puertos mediante UDP.
- L3D/L4D.
- Appflags: idem a Apps para el "Reporte TCP".
- Port signatures.

Un caso similar al del escaner de puertos del Reporte TCP podría tener esta apariencia para el reporte UDP:

Ip_src	Weight	UDP_sent	UDP_rcv	Unreachs	L3D/L4D	Appflags	Port signature	Ip_dst
192.168.1.100	114696930	72685	0	1578	720/1		[139,100]	200.22.x.x

Como se puede ver hay muchos paquetes UDP enviados y ninguno recibido, adicionalmente hay muchos errores de ICPM expresados como un “Unreachs” alto; como consecuencia de lo anterior la carga de trabajo es bastante alta. Por su parte se puede apreciar que se están escaneando muchas IP’s concentrándose solo en el puerto 139. Este es un claro ejemplo de un probable Bot que está actuando como escáner UDP de puertos.

Reporte Email Syn

Este reporte es en sí una variante del “Reporte TCP” compartiendo con este los siguientes campos: Ip_src, Esyn/eww, Work, SA/S, L3D/L4D, Ip_dst, Snt/rcv, Port Signature. La diferencia principal es que se agrega un campo Esyn/eww que mide la cantidad de SYN vistos en los mails por sobre la carga total de trabajo de un intervalo de tiempo determinado. Muchos Bot’s al albergarse en un host determinado utilizan dicho host como plataforma de spamming para hacer dinero con ello y es por eso que Ourmon incluye este reporte para identificar a los posibles “spammers”. A continuación se provee un reporte de ejemplo de un host que resulta sospechoso de estar enviando spam:

Ip_src	Esyn/eww	Work	SA/S	L3D/L4D	Snt/rcv	Port Signature
192.168.1.100	22/98	98	4	78/1	489/228	[25,100]

Como se aprecia se tiene una carga de trabajo de casi el 100% lo cual indica que el host está enviando mucho tráfico; asimismo hay más paquetes enviados que recibidos y el SA/S identifica al host como un posible cliente. Además no resulta extraño que todo el tráfico esté concentrado en un solo puerto (el 25) que corresponde a SMTP.

Reporte IRC

El reporte IRC en sí se encuentra conformado por un conjunto de subreportes; básicamente existen 2 categorías de subreportes: por canal IRC, por hosts relacionados con un canal IRC.

El subreporte por IRC puede verse ordenado de varias formas y básicamente es un listado de todos los canales IRC conformado básicamente por los siguientes campos:

- Channel: nombre del canal IRC.
- Msgs: total de mensajes IRC para todos los host en un canal.
- Joins: total de JOIN's.
- Privmsgs: total de PRIVMSG's
- Ipcount: número total de IP's asociadas a un canal.
- Wormyhosts: el número de host que se comportan como escáneres de acuerdo a su carga de trabajo.
- Evil: es un flag que toma el valor de e cuando hay al menos un wormyhost o el valor de E cuando hay al menos 2.

Por cada subreporte por IRC existe un reporte por hosts que enumera todos los host pertenecientes a un canal IRC determinado y los ordena por algún criterio. Este subreporte se encuentra compuesto por los siguientes campos:

- Ip_src: IP que se encuentra vinculada al canal IRC.
- Tmsgs: total de mensajes IRC (JOINS, PINGS, PONGS, PRIVMSGs) para esa IP; estos valores también se cuentan en forma individual.
- Maxworm: una versión particular de la carga de trabajo que en una sumariazación es la más alta registrada de todas las instancias de tiempo de 30 segundos.

- Server: un flag que indica H si la IP es un cliente o S si la IP es un servidor.
- Sport/dport: un muestreo de un puerto de origen y destino para el tráfico IRC de ese host.
- First_ts: una marca de tiempo con la primera vez que el host mostró actividad IRC en un período de tiempo determinado al que corresponde el reporte.

Si bien los reportes de IRC son útiles, también son limitados en el sentido de que Ourmon solo identificará canales IRC cuando vea las palabras reservadas JOINS, PINGS, PONGS, PRIVMSGs para cualquier puerto; sin embargo, en algunos casos los Bots utilizan versiones modificadas de IRC en cuyo caso Ourmon no sería capaz de detectar este canal subliminal de comunicación.

Más allá de la limitación anteriormente planteada el reporte puede ser útil en algunos otros casos; a continuación se muestra un ejemplo de este tipo de reporte:

Channel	Msgs	Joins	Privmsgs	Ipcount	Wormyhosts	Evil?
CanalA	2820	2500	320	3	2	E
CanalB	1940	1700	240	2	0	

La tabla que se visualiza previamente corresponde al subreporte de canal IRC, como se aprecia el “CanalA” tiene 2 host que podrían estar auspiciando de escáneres y por lo tanto podrían tener un Bot instalado en ellos. Si se ingresa dentro del subreporte de host para el “CanalA” que se visualiza a continuación:

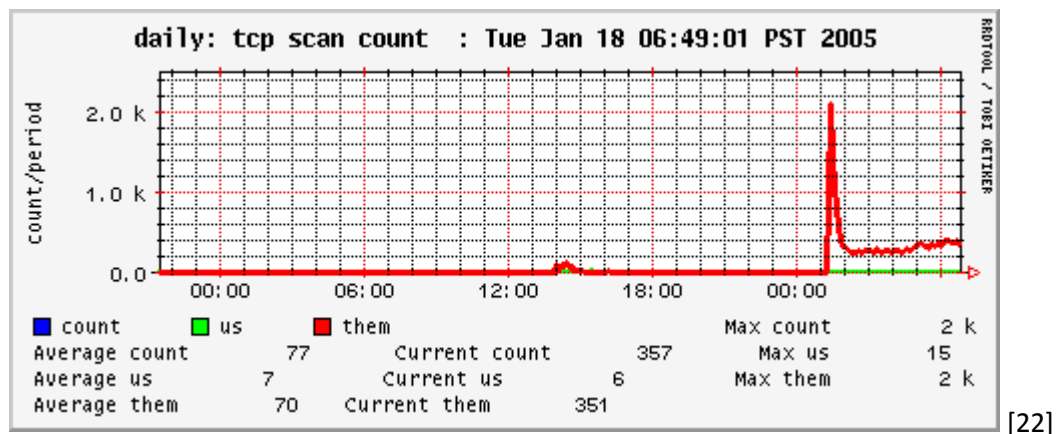
Ip_src	Tmsgs	Maxworm	Server?	Sport/dport	First_ts
192.168.1.100	1000	8	H	25321/3322	Sun_Oct_15_1:30:32
192.168.1.101	1200	7	H	25541/3322	Sun_Oct_15_0:22:30
192.168.1.102	620	2	H	22314/1254	Sun_Oct_15_0:20:15

Se puede apreciar que existen dos hosts con una alta carga de trabajo (Maxworm) que son los “Wormyhosts” declarados para este canal; ambos

son hosts y apuntan hacia el puerto 3322 donde debe residir el servidor destino. Con esta información se puede visualizar el Reporte TCP para obtener más información sobre el host en cuestión y tomar medidas adicionales.

Worm Graph

Este reporte se presenta solo en forma gráfica; un ejemplo del mismo es el que a continuación se puede ver:



Básicamente este gráfico es una comparación de los paquetes enviados desde la Intranet hacia Internet (Us) y viceversa (Them) durante un período de tiempo determinado (por ejemplo 24 hs) y de allí el modismo de Us vs. Them.

Para que este reporte pueda diferenciar que es la red local (o Intranet) de lo que es internet hace falta declarar cual es la “local home” en el archivo de configuración de Ourmon como se puede ver a continuación:

```
# vi ourmon/etc/ourmon.conf
```

```
topn_syn_homeip 192.168.0.0/16
```

Algo interesante de este gráfico es que se pueden llegar a detectar picos de tráfico o tráfico alto sostenido; si el caso es este último se puede realizar un escaneo de la red con ngrep como se verá posteriormente en este capítulo, si el caso es que se encontró un pico de tráfico puede resultar muy útil verificar en que instante de tiempo se produjo el mismo y mirar los logs TCP de Ourmon (de 30 segundos de intervalo de tiempo) para ese día. Si por

ejemplo el incidente resultase ser un lunes 17 de enero del 2011 cerca de las 4:25 PM para la IP 192.168.1.100 entonces se podría encarar la búsqueda en los logs del siguiente modo:

```
# cd /home/mrourmon/logs/portreport/Sun
```

```
# find . | xargs wc -l | sort
```

```
...
```

```
196 ./Tue_Jan_17_16:25:05_PDT_2011.portreport.txt
```

```
509 ./Tue_Jan_17_16:25:35_PDT_2011.portreport.txt
```

```
2214./Tue_Jan_17_16:26:07_PDT_2011portreport.txt
```

```
# grep 192.168.1.100 Tue_Jan_17_16:25:05_PDT_2011.portreport.txt
```

Luego se deberá efectuar los análisis sobre el Reporte TCP como ya anteriormente se especificó.

Sysdump

Este reporte (que por cierto posee una sumarización diaria) es una variante del Reporte TCP con un mayor detalle de información; la mejor manera de comprenderlo es mediante un ejemplo como el que se presenta a continuación:

```
192.168.1.105 EW IP (72:78:82) 0: (1552/8) (4221:33:0) (4337:417)
```

```
dns: juan.com.ar
```

```
:20: Sun_Feb__13_11:54:18_PDT_2011:  
Thu_Feb__15_22:34:21_PDT_2011:
```

```
portuples[7]: [445 72588] [139 22214] [8080 5226]***
```

Línea por línea los campos son los siguientes:

En la primera línea: dirección IP, Flags, App flags, carga de trabajo (presentada como min, promedio y max), SA/S promedio, L3D/L4D, promedio de SYN/FIN/RESET, Snt/rcv.

En la segunda línea el nombre del host resuelto por DNS (siempre que sea posible).

En la tercera línea: Instance count (es el conteo de la cantidad de veces que apareció la IP en un reporte), First timestamp (es una marca de tiempo de la primera vez que apareció la IP en un reporte), Last timestamp (es una marca de tiempo de la última vez que apareció la IP en un reporte).

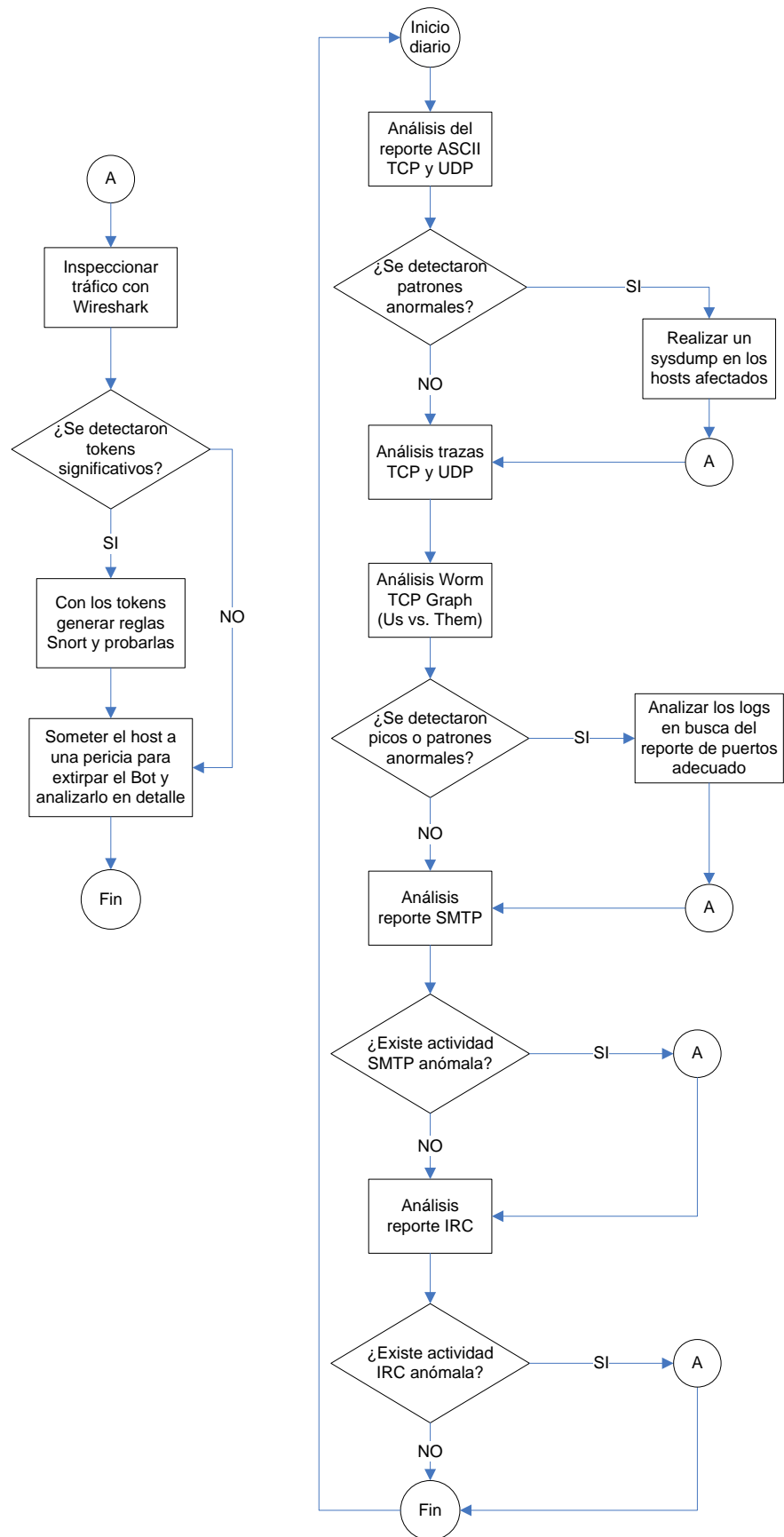
La cuarta línea contiene una versión especial del campo Port Signature donde se hace un listado ordenado por cantidad de paquetes mostrándose la cantidad de paquetes para cada uno de esos puertos de una determinada IP; por una cuestión de prolijidad en el log se muestran solo los 10 puertos más congestionados y en el caso de este ejemplo solo 3.

Como se puede ver, este reporte puede ayudar a un operador a encontrar más información sobre una IP sospechosa en caso de que no sea suficiente con el Reporte TCP o el Reporte UDP y se quiera recabar más información.

Un enfoque iterativo para la detección de Botnets y generación de reglas de Snort

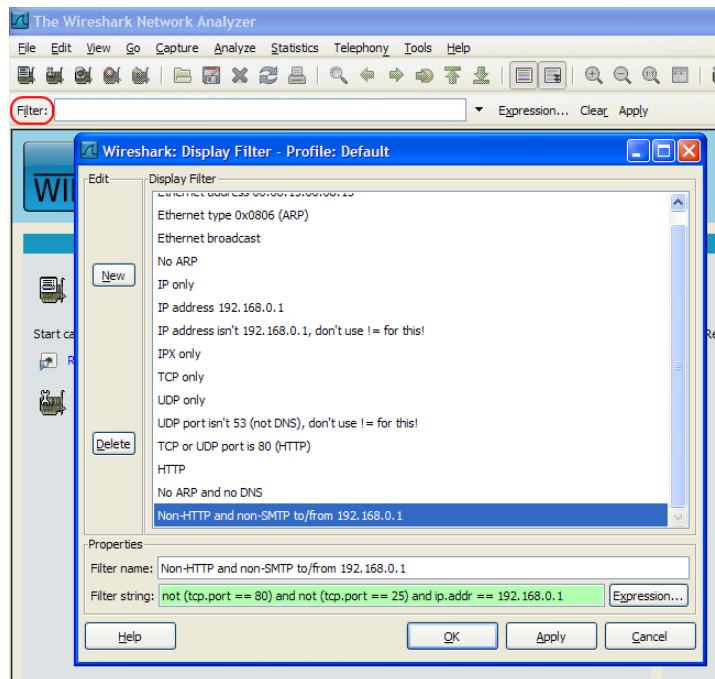
De forma similar al enfoque iterativo que ya se presentó en el Capítulo 1 utilizando las herramientas Snort, NTOP y Wireshark es posible trabajar con Ourmon de forma de utilizar las experiencias aprendidas del monitoreo diario del tráfico de red organizacional para enriquecer el repertorio de reglas de Snort (este enfoque considera que ya se ha instalado Snort como se indica en el Capítulo 1 y en el Anexo A).

En síntesis el enfoque metodológico iterativo es el siguiente:



El procedimiento se inicia cuando diariamente un operador comienza analizando los listados (o tablas) de TCP y UDP que provee Ourmon, de allí y utilizando las técnicas descritas en la sección anterior (véase “Sobre los reportes”) se debe derivar en si se encontraron o no patrones anormales. Si este último es el caso entonces se debe proceder a revisar un reporte más detallado sobre la actividad TCP y UDP que Ourmon denomina Sysdump; la razón de realizar este paso es tener una idea más detallada de la anomalía.

Luego se debe revisar cada uno de los hosts comprometidos inspeccionando el tráfico con Wireshark; para ello se debe instalar previamente la herramienta en cada uno de ellos. La idea aquí (al igual que en el enfoque iterativo del Capítulo 1) es obtener tokens significativos con los cuales se puedan generar reglas de Snort para detectar futuras amenazas automáticamente. Dado que posiblemente el tráfico del host sea mucho es preferible filtrar por algún patrón como puerto, IP o protocolo que deben ser obtenidos de los listados de Ourmon sobre los cuales se derivo en la sospecha de un host comprometido. Wireshark cuenta de por sí con un amplio repertorio de patrones y expresiones para aplicar filtros sobre el tráfico como se muestra a continuación:



Una vez aplicado el/los filtros correspondientes se deberá determinar si se encuentran o no tokens significativos; en caso de que la respuesta sea

positiva se deberán utilizar los mismos para generar reglas de Snort (véase "Un modelo iterativo de detección de Botnets y mejora de reglas") y en caso negativo (o luego de generar las reglas de Snort) se deberá proceder a someter el/los hosts a una pericia para determinar que amenaza los aqueja y proceder a su eventual eliminación siguiendo la metodología propuesta en el Capítulo 4.

Volviendo al punto en donde se verificaba si se encontraban patrones anormales de los reportes TCP y UDP se deberá proseguir hacia los siguientes pasos que implican un análisis de las trazas (gráficos 2D) TCP, UDP y el Worm Graph. El análisis de estos gráficos consiste básicamente en la detección de patrones anormales como picos abruptos en el tráfico o alzas mantenidas en la carga que hagan sospechar de una situación anormal (cabe aclarar que es necesario conocer primero el tráfico "normal" de la red analizada para poder determinar que se considera "anormal"). En caso de que se detecten picos o patrones anormales se debe proceder a analizar los logs en busca del reporte de puertos adecuado como se indica en el apartado "Worm Graph" de la sección "Sobre los reportes" del Capítulo 4. Una vez realizado esto se debe pasar nuevamente al paso de "revisar cada uno de los hosts comprometidos inspeccionando el tráfico con Wireshark" como anteriormente se detalló.

Continuando con el procedimiento se deberá proceder al análisis de los reportes SMTP e IRC como se indica respectivamente en los apartados "Reporte Email Syn" y "Reporte IRC" de la sección "Sobre los reportes" del Capítulo 4; en cualquiera de los casos si se encontrase actividad SMTP o IRC anómala respectivamente se deberá proceder nuevamente al paso de "revisar cada uno de los hosts comprometidos inspeccionando el tráfico con Wireshark" como anteriormente se detalló.

Luego de los pasos anteriormente mencionados, el procedimiento se termina y se deberá volver al principio para empezar nuevamente.

Es una recomendación realizar las tareas del procedimiento en forma diaria y trabajar con la red organizacional durante un tiempo razonable con la

herramienta Ourmon para poder identificar cuáles son los patrones normales del funcionamiento de la red como ya anteriormente se sugirió.

Capítulo 4: Forensia sobre host y malware

Introducción

Una vez que se ha detectado actividad sospechosa con alguno de los sensores instalados en la Intranet (Snort, Nepenthes y/o Ourmon) es tiempo de someter el host destino a una pericia de forma de detectar y neutralizar el malware que lo afecta. Cabe aclarar que muchas veces la mejor solución para terminar con el problema del malware en el host es salvar la información importante que este tiene y proceder a su formateo y posterior implantación de una imagen organizacional “fresca”; la razón de este proceder en el ámbito organizacional tiene muchas veces que ver con cuestiones de tiempo, recursos capacitados o infraestructura, pero más allá de eso es conveniente siempre que se pueda realizar un esfuerzo adicional en tratar de detectar el malware y eliminarlo. Justamente el objetivo de este capítulo es brindar al analista de seguridad de algunas de las herramientas esenciales para realizar una pericia sobre un host infectado y determinar la razón de la infección como así también resolverla.

Más allá de la pericia sobre el host que siempre será necesaria, es posible que se cuente con una muestra del binario del malware capturado por Nepenthes. Si este es el caso, se puede realizar un análisis sobre el binario para comprender su funcionamiento y algunas de sus características de comportamiento más importantes. En este capítulo además cubre el análisis forense sobre el malware.

Se comenzará con un detallado análisis de las técnicas forenses necesarias para realizar una pericia sobre un host infectado por malware y luego se cerrará con un análisis del comportamiento del malware.

Nota: a lo largo de este capítulo se utilizan los términos pericia y forensia de forma indistinta.

Alcance pericial y consideraciones previas

Antes de comenzar a detallar como se realiza técnicamente una pericia en un host es conveniente detallar un poco el alcance pericial planteado en el presente trabajo.

Algunas veces las pericias que se realizan en el ámbito organizacional, trascienden más allá de este ámbito y adquieren una esfera legal. Es por ello, que se deben tomar una serie de consideraciones previas para preservar la evidencia. Este tipo de procedimiento es sumamente común cuando la pericia que se realiza sobre el equipo estuvo motivada en algún comportamiento incorrecto por parte del personal organizacional; por ejemplo: un alto gerente que posee material pornográfico en un PC o un empleado que robó o accedió a determinado contenido confidencial, entre otros. Sin embargo, en el caso de un ataque automatizado (como el caso de los Botnets) no se justifica llevar una pericia sobre un host al ámbito judicial; de esta forma todas las consideraciones previas de preservación de la evidencia no serán tenidas en cuenta en este trabajo.

Realmente en lo que respecta a este tipo de pericias la única consideración previa a tener en cuenta sería la de desconectar el host infectado de la red tan pronto como se pueda para evitar que la infección prolifere en la Intranet.

Con respecto a las herramientas que se presentaran a posteriori para realizar la pericia, podría armarse un CD con un compilado de las mismas de forma de poder copiar las herramientas fácilmente en el host destino.

Alcances y consideraciones sobre la pericia de malware

Realizar una pericia sobre una pieza de código malicioso es en sí una actividad compleja que requiere de cierta expertise. Básicamente existen 2 tipos de pericia sobre malware, a saber: caja negra y caja blanca.

Una pericia de caja blanca implica un análisis completo y total del funcionamiento del malware; un análisis de este tipo implica una ingeniería inversa sobre el código del malware. Sacando la complejidad misma que este proceso tiene, es conveniente también aclarar que el código binario del malware la mayoría de las veces se encuentra protegido (mediante packers y cifradores) para impedir su correcta identificación. De esta forma primero habrá que identificar la protección y luego removerla antes de que se pueda realizar el proceso de ingeniería inversa en sí.

Por su parte una pericia de caja negra es en sí más sencilla e implica un análisis de comportamiento del malware (behavioral analysis); se llama de caja negra porque justamente no mira el funcionamiento interno del malware sino el funcionamiento que este exhibe en sus interfaces. Un análisis de este tipo puede automatizarse bastante mediante el uso de herramientas con Sandboxes web (ya presentadas previamente), aunque muchas veces vale la pena hacerlo manualmente en un entorno controlado ya que se puede extraer mayor información además de ganar conocimiento.

Tomando en cuenta lo hasta aquí dicho, el tipo de análisis de malware que se utilizará será el de caja negra en forma manual. La decisión se basa en hacer un balance entre el tiempo de pericia y su complejidad.

Ejecutando la pericia sobre host [23]

Una vez que se cuenta con la información necesaria para determinar desde que host proviene el ataque, se deberá proceder a desconectar ese host de la Intranet y transferir desde un CD el pack de herramientas forenses al host destino. A lo largo de este apartado se presentarán un conjunto de herramientas probables que podría incluir dicho pack y se informará como utilizarlas.

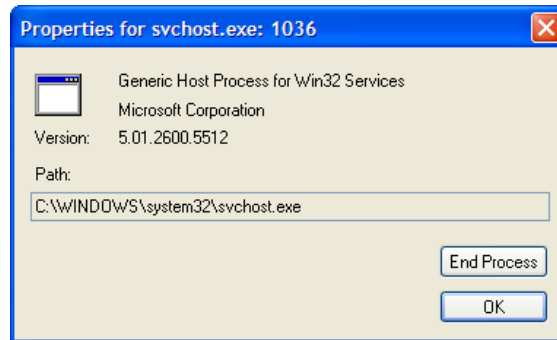
Si se está utilizando alguna de las técnicas para detectar los Botnets ya explicitadas en la sección precedente, entonces probablemente se tiene al menos una IP desde la cual provino la conexión que inició el posible malware y el puerto. Tomando esta información como base para el análisis lo primero que hay que hacer es asociar el puerto al programa que yace tras este. Para ello una excelente alternativa consiste en utilizar TCPView de Sysinternals cuya captura se presenta a continuación:

Proc...	PID	Protocol	Local Address	Local Port	Remote Address
alg.exe	2596	TCP	juanale	1029	juanale
cmdagent.exe	996	TCP	juanale	1887	vip1.sa-anycast1....
cmdagent.exe	996	TCP	juanale	1886	download.comodo...
FileZilla Serve...	3832	TCP	juanale	1036	localhost
FileZilla server...	720	TCP	juanale	14147	localhost
FileZilla server...	720	TCP	juanale	14147	juanale
FileZilla server...	720	TCP	juanale	ftp	juanale
firefox.exe	2088	TCP	juanale	1384	localhost
firefox.exe	2088	TCP	juanale	1383	localhost
firefox.exe	2088	TCP	juanale	1385	localhost
firefox.exe	2088	TCP	juanale	1386	localhost
googletalk.exe	2468	TCP	juanale	1047	yx-in-f125.1e100.ne
jqs.exe	984	TCP	juanale	5152	juanale
LEXPPS.EXE	1404	TCP	juanale	1025	juanale
lsass.exe	692	UDP	juanale	isakmp	*
lsass.exe	692	UDP	juanale	4500	*
rapimgr.exe	3884	TCP	juanale	990	juanale
svchost.exe	924	TCP	juanale	epmap	juanale
svchost.exe	1036	UDP	juanale	ntp	*
svchost.exe	1212	UDP	juanale	1900	*
svchost.exe	1036	UDP	juanale	ntp	*
svchost.exe	1212	UDP	juanale	1900	*
System	4	TCP	juanale	microsoft-ds	juanale
System	4	TCP	juanale	netbios-ssn	juanale
System	4	UDP	juanale	netbios-ns	*
System	4	UDP	juanale	microsoft-ds	*
System	4	UDP	juanale	netbios-dgm	*
wcescomm.exe	3792	TCP	juanale	7438	juanale
wcescomm.exe	3792	TCP	juanale	5679	juanale

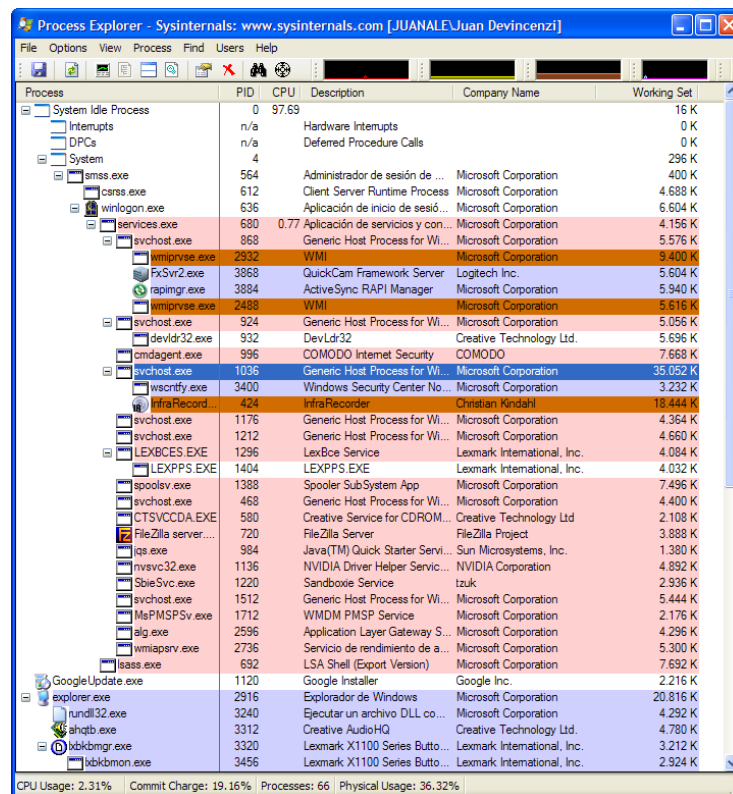
Endpoints: 29 Established: 7 Listening: 11 Time Wait: 0 Close Wait: 2

Como se puede ver, posee una interfaz intuitiva que permite identificar rápidamente el puerto sospechoso y el proceso asociado a él.

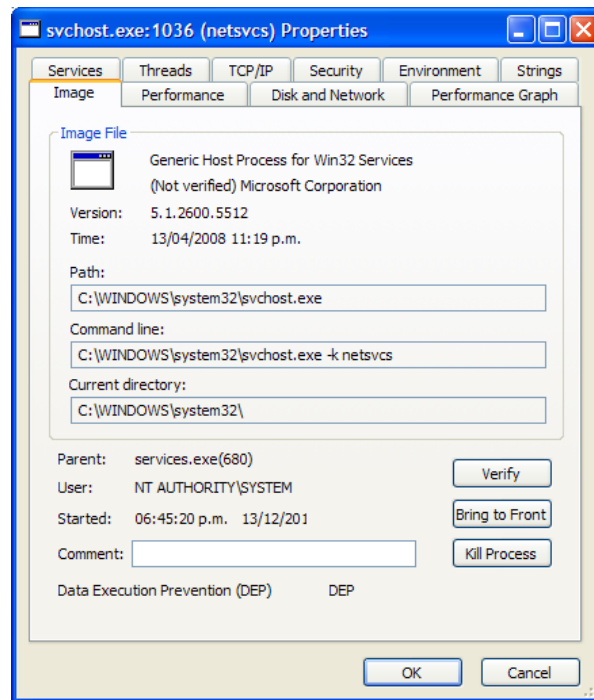
Es posible también obtener información extra sobre el programa que yace tras la conexión seleccionada si se hace doble click sobre ella:



Una vez que se posee el nombre del proceso, el siguiente paso es obtener más información del mismo; con este objetivo en mente se perfila como la mejor alternativa la herramienta Process Explorer de Sysinternals. El primer paso es ubicar el proceso y puerto que se había individualizado con TCPView.

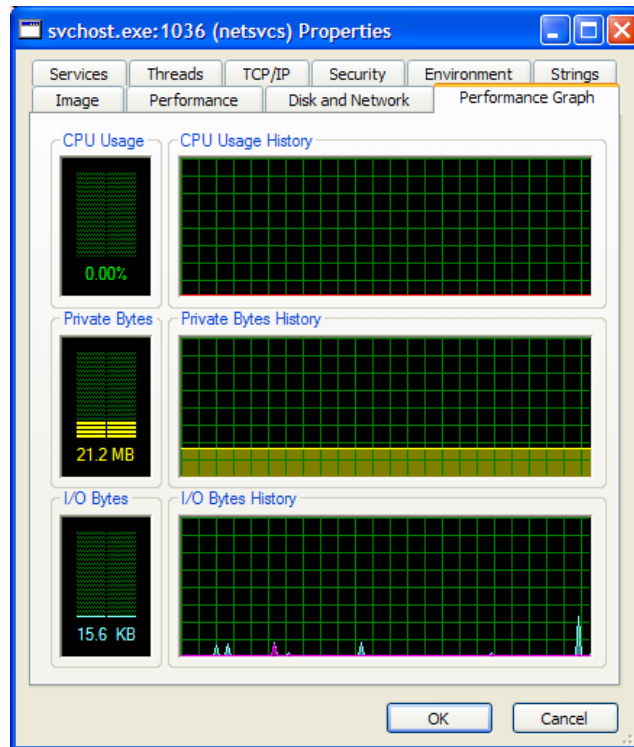


Haciendo doble click sobre el proceso se puede acceder a información extra que será muy útil para la pericia; hay varias solapas pero se comenzará con la descripción de "Image".



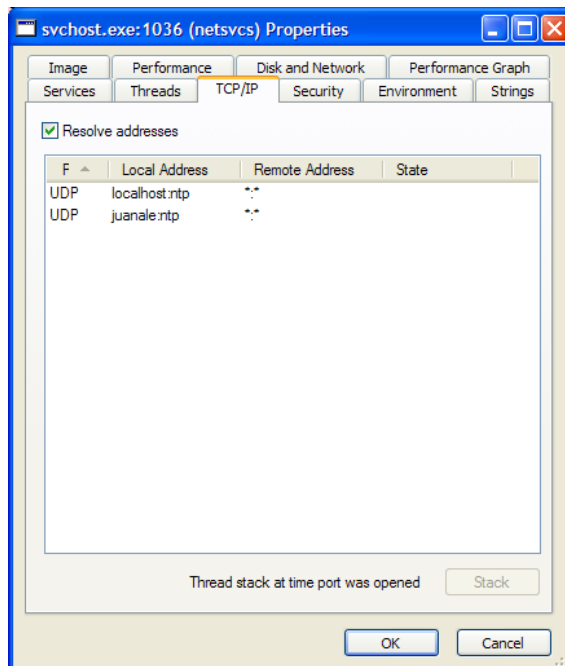
Como se puede apreciar, además de toda la información relativa al proceso que se está ejecutando, también se puede visualizar la ruta completa al proceso y su directorio (como con TCPView) y los parámetros con los cuales se ejecutó el proceso. Una característica interesante es la posibilidad de verificar si el proceso (si es un proceso propio del sistema operativo) está certificado por Microsoft; esto se puede conseguir presionando el botón "Verify" sobre el proceso deseado. La característica de "Verify" resulta interesante puesto que hay muchos malware's que intentan camuflarse llamándose a sí mismos con nombres similares a procesos del sistema operativo. Si el proceso es legítimo, al presionar "Verify" dirá "Verified".

Otra solapa interesante que vale la pena mencionar es la denominada "Performance Graph" que otorga una vista del consumo de CPU, memoria y E/S de un proceso determinado y se presenta a continuación.



Es muy común que algunos malware (incluidos los Bots) tengan un consumo anormalmente alto de CPU, por lo que una inspección a esta solapa es siempre recomendada.

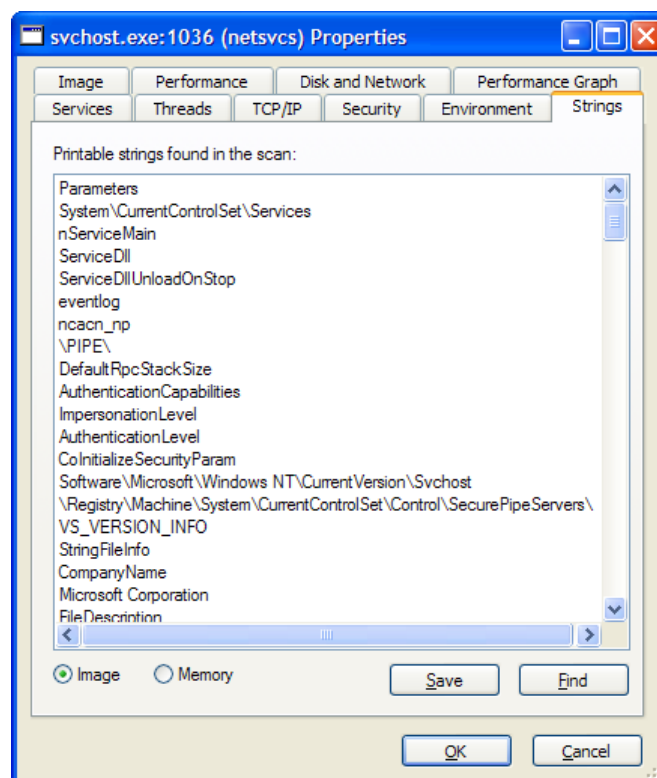
La solapa "TCP/IP" que se puede apreciar a continuación:



Permite identificar las conexiones que un proceso ha establecido al momento en el cual se lo está analizando. En caso de que este proceso sea

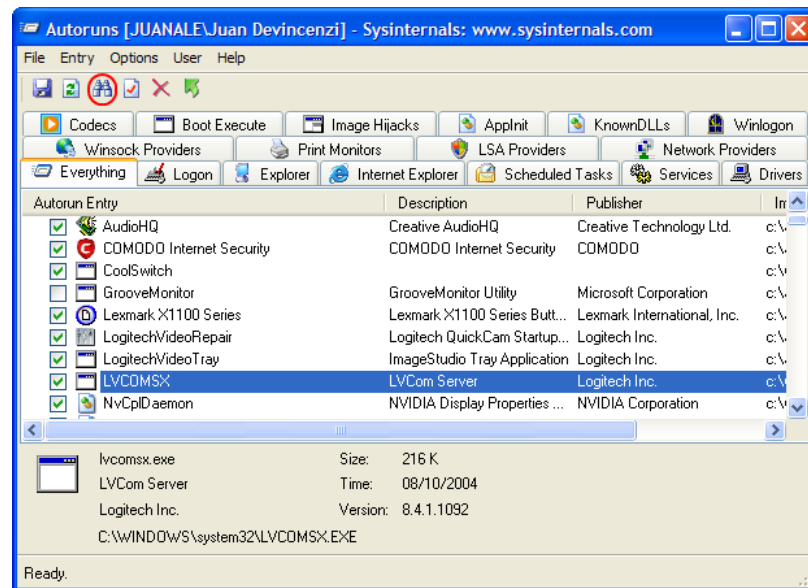
un malware del tipo Bot, entonces la información aquí listada puede identificar la dirección IP del servidor que contiene el servicio de la Botnet o alguna dirección de la Intranet al cual el Bot está intentando expandirse; ambos datos son importantes para la investigación forense.

Por último, la solapa “Strings” muestra los caracteres imprimibles sucesivos que superen una determinada longitud. Esta opción resulta interesante para identificar si el proceso es alguna clase de malware. ¿Por qué? La razón es que muchos malware’s se encuentran empaquetados (packaged) en disco y se descomprimen en memoria en tiempo real cuando se los ejecuta; el empaquetado en si es una técnica para dificultar la ingeniería inversa de una pieza de malware y es un componente muy común en los binarios de los códigos maliciosos que circulan en Internet. La solapa “Strings” permite ver tanto las cadenas en memoria como en disco como se puede apreciar en la siguiente imagen:

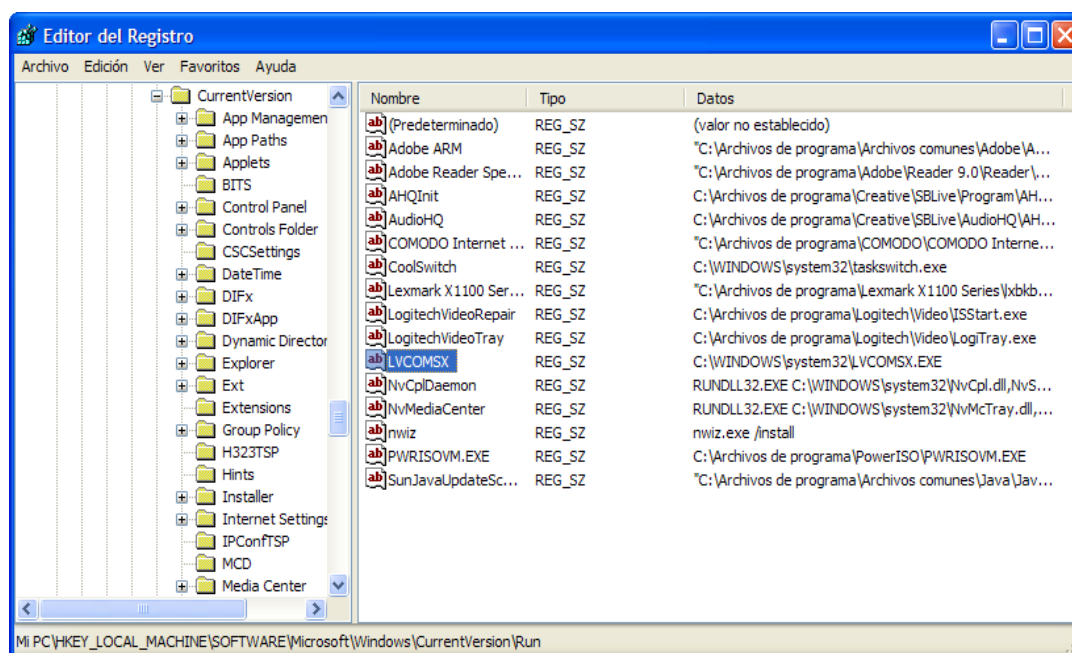


Cuando un malware se encuentra empaquetado las cadenas en memoria difieren considerablemente a las de disco y en estas últimas puede encontrarse gran cantidad de contenido “basura”. Estas premisas resultan útiles para identificar si el proceso es un Bot.

Los Bots son procesos maliciosos que necesitan persistir, es decir estar en ejecución todo el tiempo y volverse a iniciar cada vez que el host se reinicia; para ello necesitan hacer uso de una característica específica del sistema operativo para iniciar junto con este y cada vez que este inicia. Realmente se pueden encontrar muchas formas de iniciar un proceso junto con el sistema operativo, pero existe un aplicativo que las reúne a todas. El nombre de este aplicativo es Autoruns y es también de Sysinternals.



Autoruns cuenta con varias solapas en las que se clasifican los programas que inician con el sistema; la solapa de “Everything” los contiene a todos y es muchas veces conveniente buscar el programa deseado en esta solapa con la ayuda de la búsqueda que se encuentra resaltada en la imagen previa con el ícono de los binoculares. Una vez que se haya encontrado el proceso sospechoso, haciendo doble click sobre el mismo, Autoruns lleva a la clave del registro asociada al inicio automático de dicho proceso como se puede ver:

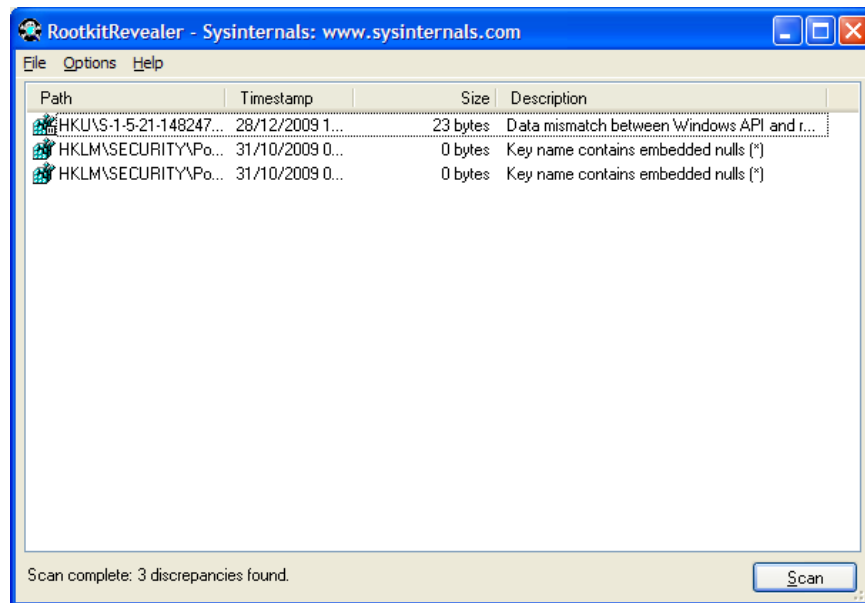


Más allá de estas medidas para identificar el proceso asociado al posible malware y determinar si efectivamente este es un malware, existen una serie de acciones que conviene llevar a cabo sobre el host posiblemente infectado.

La primera y más directa de estas medidas consiste en llevar a cabo un scan antivirus; sobre esta medida no se va a entrar en detalle por ser bastante directa y fácil de realizar.

La segunda medida consiste en llevar a cabo un análisis de los posibles rootkits que pudiera tener el host. Existen muchas herramientas que son capaces de descubrir posibles archivos, procesos, conexiones ocultas mediante un rootkit, pero la que aquí se recomienda es Rootkit Revealer de Sysinternals.

Rootkit Revealer permite identificar determinados objetos del sistema operativo que se encuentran ocultos a la API de Windows; por ejemplo un archivo que mediante el explorador de Windows no se ve pero si puede visualizarse de forma alternativa. La función de Rootkit Revealer es justamente encontrar y presentar al usuario los objetos que cumplen con estas diferencias.



Una tercera y última medida consiste en realizar un scan de los Alternate Data Streams en todo el disco rígido o en parte de él. En realidad todo archivo que se almacena en un FileSystem tipo NTFS se encuentra compuesto por una serie de flujos; estos flujos son en realidad metadatos del sistema operativo y existen 2 flujos obligatorios en todo archivo, a saber: seguridad que contiene la ACL del archivo y datos que contiene al contenido del archivo en sí. Luego de estos flujos obligatorios un archivo en sí puede contener infinitos flujos extra o alternativos que siempre deben ser considerados como sospechosos ya que muchos hackers los utilizan para camuflar binarios enteros dentro de otros binarios sin alterar el tamaño del binario original. Es conveniente decir también que no todo flujo alternativo tiene por qué ser ilegítimo; existen algunos casos donde estos flujos se utilizan para usos lícitos como por ejemplo información de cifrado de los archivos.

Un excelente aplicativo para conocer todos los flujos alternativos que pueden encontrarse en un determinado directorio es Sfind incluido en el toolkit denominado The Forensic ToolKit (FTK) de AccessData. Este aplicativo se acciona por línea de comandos y se lo puede utilizar de la siguiente forma para identificar los Alternate Data Streams del siguiente directorio:

```
C:\ftk> sfind c:\windows\system32
```

```
Searching...
```

```
c:\windows\system32
```

```
calc.exe:bot.exe Size: 105232
```

```
Finished
```

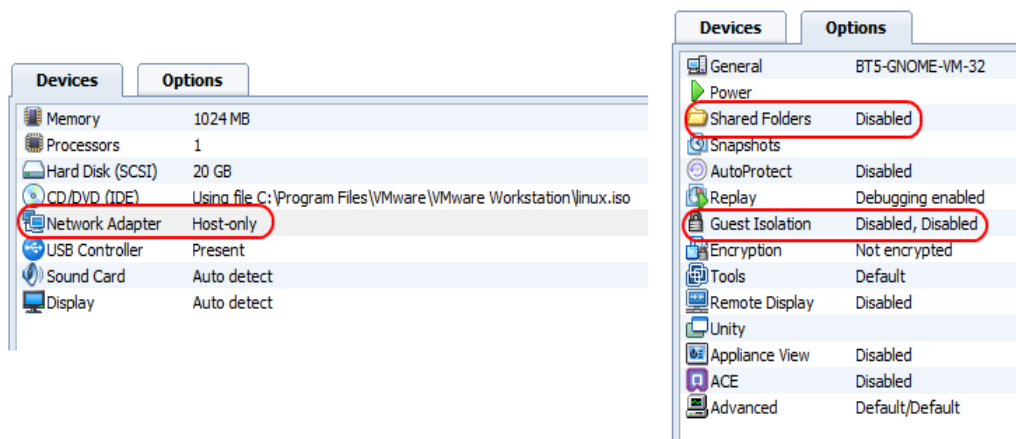
En este caso se pudo identificar un Alternate Data Stream denominado “bot.exe” dentro del binario legítimo de la calculadora de Windows (calc.exe).

Pericia sobre el malware [23]

En forma alternativa a la pericia que se realiza sobre un host, es posible en algunas situaciones realizar una pericia sobre el binario que representa al código malicioso. Este binario puede provenir de de distintas fuentes como por ejemplo puede resultar recolectado de una pericia practicada sobre un host o como resultado de la captura de Nepenthes. Sea cual fuere el motivo de la obtención del binario, es posible practicarle una pericia en forma sencilla y directa utilizando una serie de herramientas y técnicas como las que a continuación se proponen.

Antes de proceder a realizar la pericia se deben tomar algunos recaudos para evitar que el analista resulte infectado como producto de su análisis sobre el malware. Dado que para realizar un análisis forense del malware en caja negra es necesario ejecutar binario que contiene el código malicioso, dicha ejecución se debe llevar a cabo en un entorno virtual controlado. Hay muchos productos de virtualización, entre ellos se encuentra uno muy reconocido denominado VMware con el cual se trabajará en este caso.

Una vez creado el entorno de virtualización (bajo un servidor Windows en este caso) es pertinente realizar las configuraciones que a continuación aparecen expuestas:

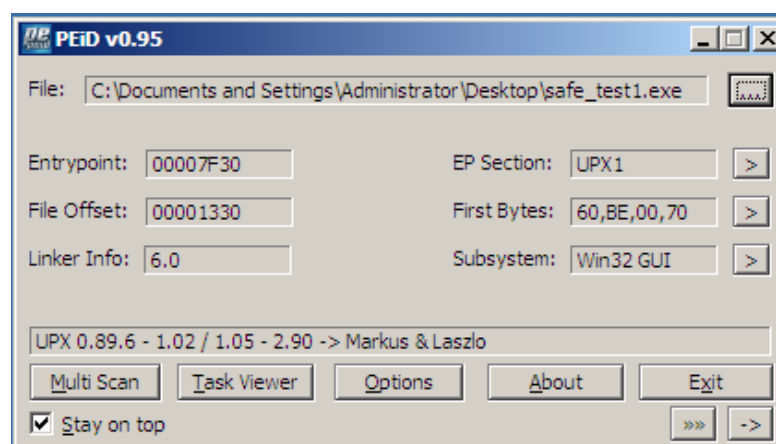


Las configuraciones referentes a “Shared Folders” y “Guest Isolation” colaboran con evitar que el malware se filtre de la máquina virtual (guest) haciendo uso de la compartición de recursos entre el host y el guest.

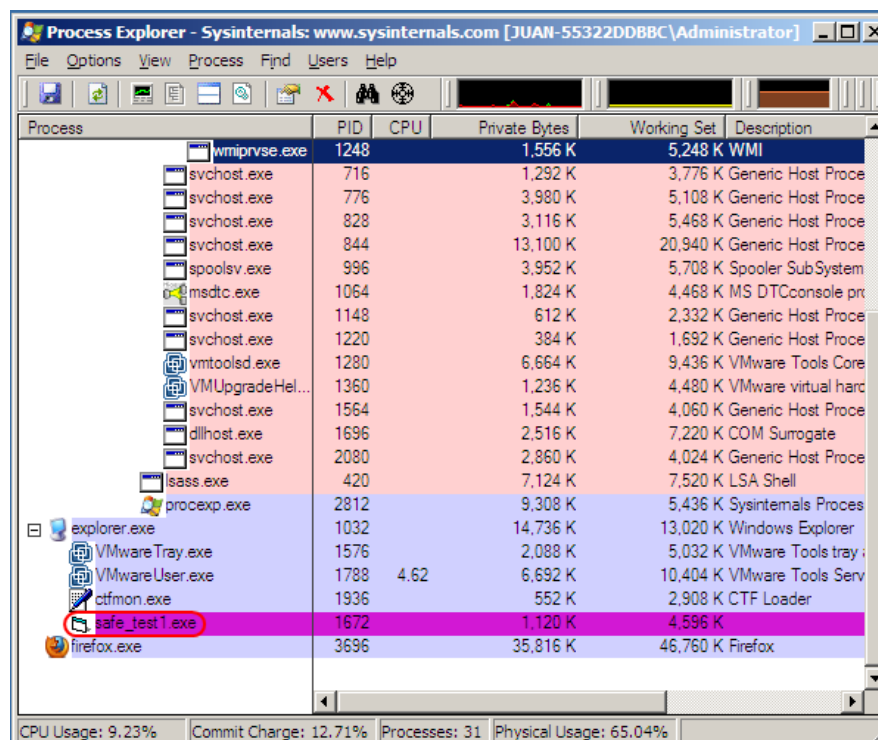
Por su parte, la configuración del adaptador de Ethernet en “Host-only” permite que el guest solo pueda comunicarse con el host. Uno de los mayores peligros cuando se ejecuta un malware tipo Bot o Worm es que es muy probable que este comience a intentar expandirse a otros host haciendo uso del mecanismo de explotación de vulnerabilidades en los hosts remotos. Justamente, al solo permitir que el guest se comuniquen con el host, se reducen las posibilidades de infección. Para eliminar totalmente dichas posibilidades sin quitarle los adaptadores de red al guest, es posible cambiar la dirección IP del adaptador de red del guest para que apunte a una dirección errónea o bien utilizar un Firewall en el host para realizar el bloqueo de todo tráfico saliente de la dirección IP del guest. Cabe aclarar que es importante que el guest posea una conexión de red, puesto que la ausencia de esta desembocará en que la pieza de malware no realizará acciones de red que son sumamente importantes para la pericia.

Una vez que ya se tiene la máquina virtual ejecutándose, con la configuración antes mencionada, se procede a realizar una imagen de la máquina virtual que luego podrá ser cargada para volver al estado original previo a la infección. Una vez realizado esto, se debe colocar en la máquina virtual el binario del malware y una serie de herramientas para la investigación que a continuación se enumeran y explican.

La primera herramienta que se detalla aquí es PEiD. Su principal función es realizar un análisis de un archivo binario sin necesidad de ejecutarlo.



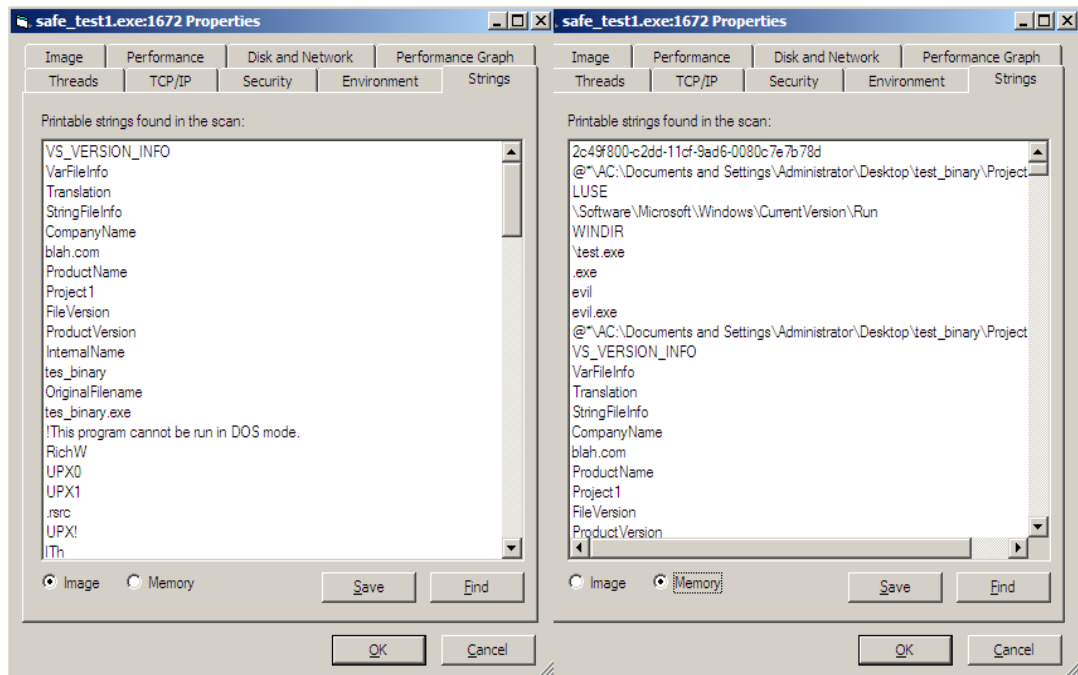
Para realizar el análisis debe ejecutarse PEiD y luego abrir mediante este el binario deseado. La herramienta brinda bastante información, pero la más valiosa de dichas informaciones es que en este caso el binario seleccionado se encuentra empaquetado (ver “Ejecutando la pericia sobre host” para una discusión más detallada del concepto de empaquetado). En este caso el empaquetado se ha realizado con una herramienta packer ampliamente utilizada y OpenSource denominada UPX. El hallazgo obtenido con PEiD puede confirmarse mediante la herramienta Process Explorer (ya vista anteriormente), pero requiere la ejecución del binario del malware; una vez que se lo ejecuta, en el Process Explorer puede verse algo como lo siguiente:



Process	PID	CPU	Private Bytes	Working Set	Description
wmiprvse.exe	1248		1,556 K	5,248 K	WMI
svchost.exe	716		1,292 K	3,776 K	Generic Host Process
svchost.exe	776		3,980 K	5,108 K	Generic Host Process
svchost.exe	828		3,116 K	5,468 K	Generic Host Process
svchost.exe	844		13,100 K	20,940 K	Generic Host Process
spoolsv.exe	996		3,952 K	5,708 K	Spooler SubSystem
msdtc.exe	1064		1,824 K	4,468 K	MS DTCConsole process
svchost.exe	1148		612 K	2,332 K	Generic Host Process
svchost.exe	1220		384 K	1,692 K	Generic Host Process
vmtoolsd.exe	1280		6,664 K	9,436 K	VMware Tools Core
VMUpgradeHel...	1360		1,236 K	4,480 K	VMware virtual hard
svchost.exe	1564		1,544 K	4,060 K	Generic Host Process
dllhost.exe	1696		2,516 K	7,220 K	COM Surrogate
svchost.exe	2080		2,860 K	4,024 K	Generic Host Process
lsass.exe	420		7,124 K	7,520 K	LSA Shell
procexp.exe	2812		9,308 K	5,436 K	Sysinternals Process
explorer.exe	1032		14,736 K	13,020 K	Windows Explorer
VMwareTray.exe	1576		2,088 K	5,032 K	VMware Tools tray
VMwareUser.exe	1788	4.62	6,692 K	10,404 K	VMware Tools Serv
ctfmon.exe	1936		552 K	2,908 K	CTF Loader
safe_test1.exe	1672		1,120 K	4,596 K	
firefox.exe	3696		35,816 K	46,760 K	Firefox

CPU Usage: 9.23% Commit Charge: 12.71% Processes: 31 Physical Usage: 65.04%

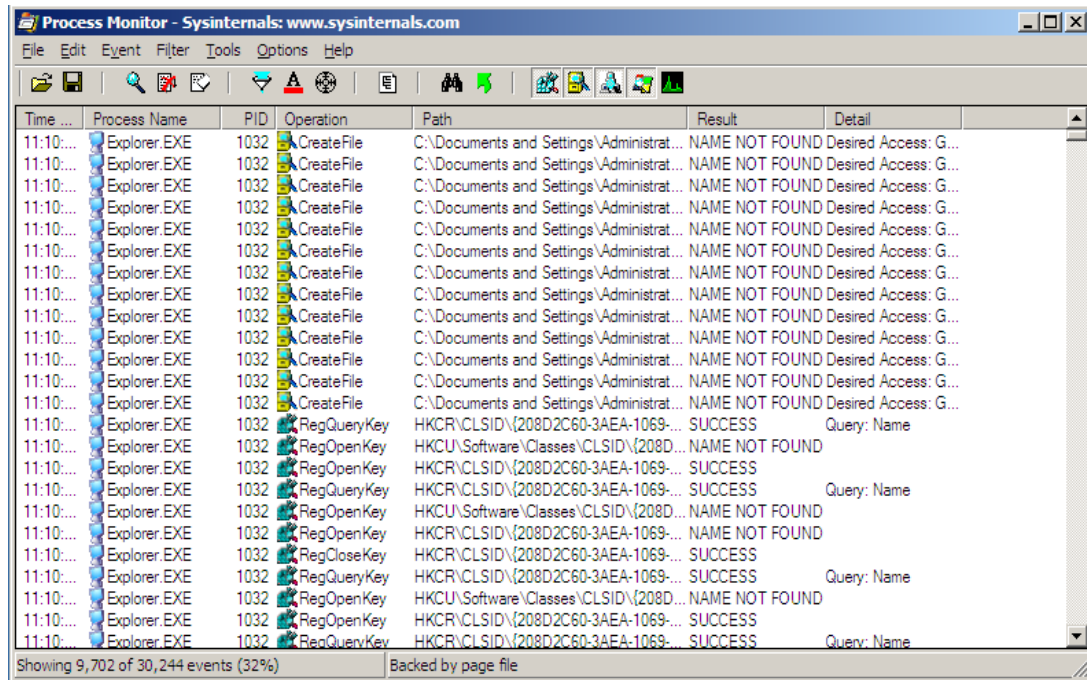
Como se puede apreciar el proceso representativo del binario sospechoso aparece con un color violeta claro que representa que las strings del archivo en disco difieren con las del archivo en memoria, lo cual sugiere fuertemente la presencia de un packer. Esto puede confirmarse analizando las strings en disco y en memoria:



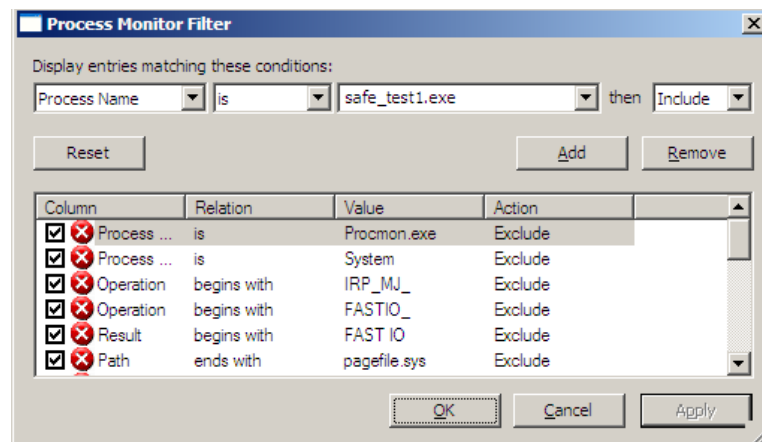
Como se ve las strings difieren enormemente, lo cual confirma el hallazgo. Una vez que se sabe que el binario está empaquetado y el algoritmo con el cual está empaquetado debe procederse a desempaquetar el binario si es que se quiere realizar una pericia de caja blanca; si lo que se va a realizar es una pericia de caja negra como en este caso, no es necesario realizar el procedimiento de desempaquetado.

Una vez realizado este análisis inicial sobre el malware, comienza el verdadero análisis de caja negra o de comportamiento. Este análisis puede realizarse con una variedad de herramientas, aquí se detallarán dos de ellas: Process Monitor de Sysinternals y SysAnalyzer.

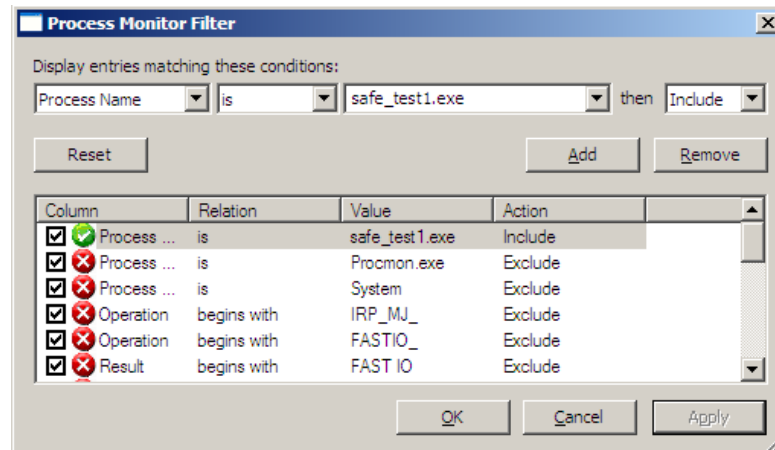
Process Monitor consiste en realidad en una herramienta capaz de recolectar y presentar los cambios que realizan todos los aplicativos que se ejecutan en un sistema operativo determinado sobre: el registro, el FileSystem, la red y las acciones importantes que ejecutan los procesos e hilos que los representan. La interfaz principal de la herramienta es la que sigue:



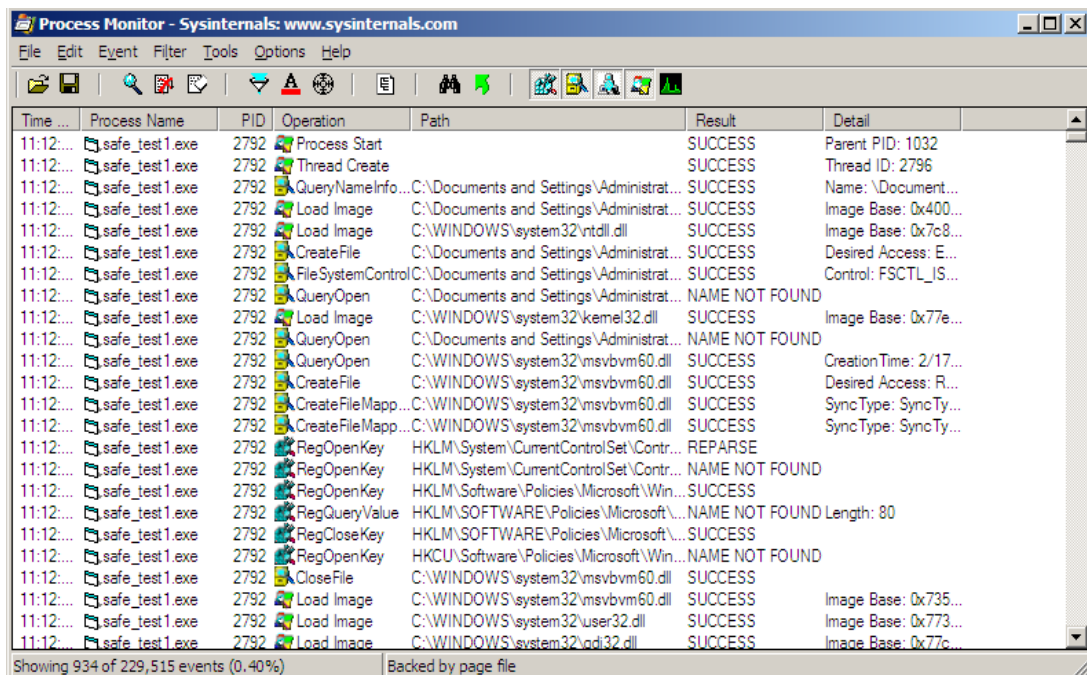
Una vez ejecutado Process Monitor (y sin cerrarlo) hay que ejecutar el binario del programa malicioso y colocar un filtro en la herramienta para reducir el “ruido” y solo ver las acciones del binario requerido.



Una buena forma de filtrar es mediante el nombre del proceso, que es “safe_test1.exe”; una vez creada la regla se debe presionar “Add” y se visualizará algo como lo que muestra a continuación:

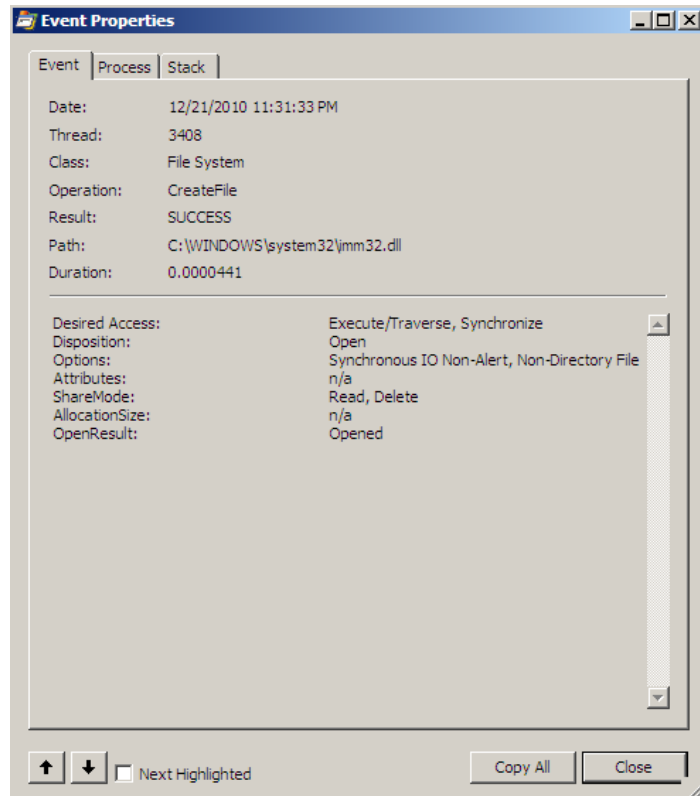


Una vez agregada la regla se puede presionar “OK” y solamente se verán los eventos relacionados con el proceso en ejecutándose que se continuarán actualizando si el proceso sigue realizando acciones.



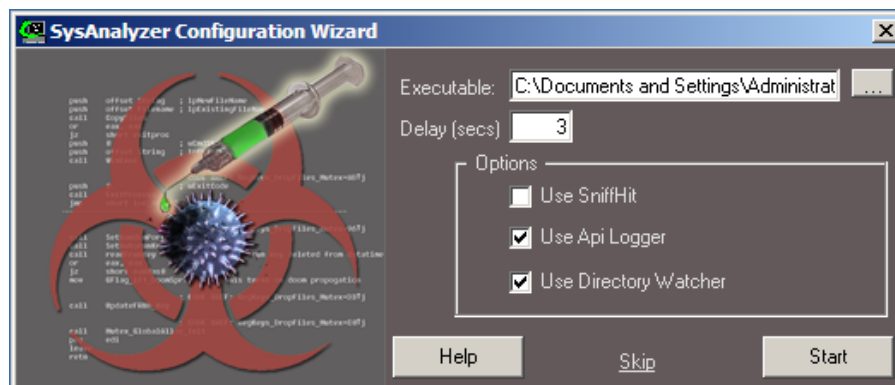
De aquí en más la destreza de cada investigador y la cantidad de acciones que realice el malware determinará los resultados finales de la investigación. La cantidad de eventos que puede generar un binario puede ser muy grande así que habrá que focalizar en los principales como la creación de determinadas claves del registro o el acceso a determinados archivos.

Además de esto, por cada uno de los eventos generados, Process Monitor permite obtener más detalle sobre el mismo si se hace doble click sobre un evento determinado.

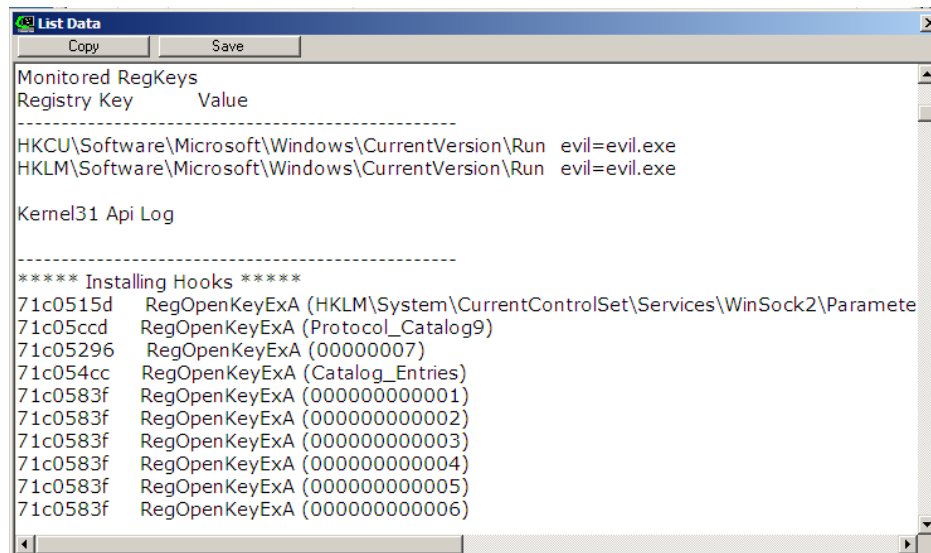


En este caso se aprecia la creación satisfactoria de un archivo en el path que se muestra.

Una herramienta alternativa para realizar el mismo proceso es SysAnalyzer cuya interfaz se presenta a continuación:

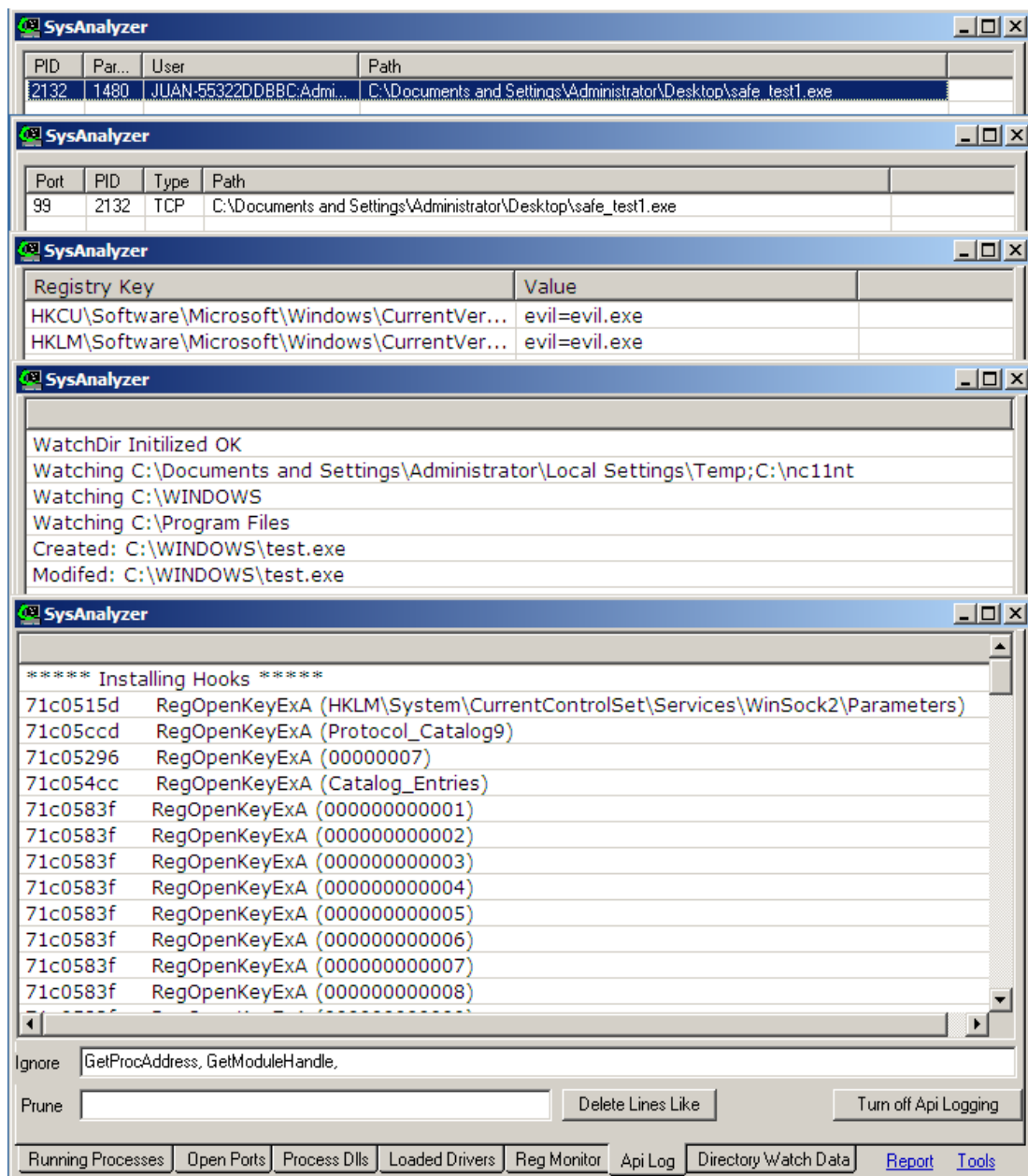


La interfaz permite elegir un archivo para abrir que en este caso será el binario con el código malicioso y existen 3 opciones para seleccionar de las cuales se utilizarán “Api Logger” (que permite ver la dependencia de API’s del binario) y “Directory Watcher” (que muestra las operaciones sobre el FileSystem que efectua el binario). Una vez seleccionadas estas opciones se procede a presionar el botón “Start” y se visualizará un reporte como este:



El reporte contiene un informe completo de las características del binario explorado que incluye: accesos al FileSystem, API's utilizadas, puertos abiertos, accesos al Registry, procesos levantados, drivers utilizados, entre otros.

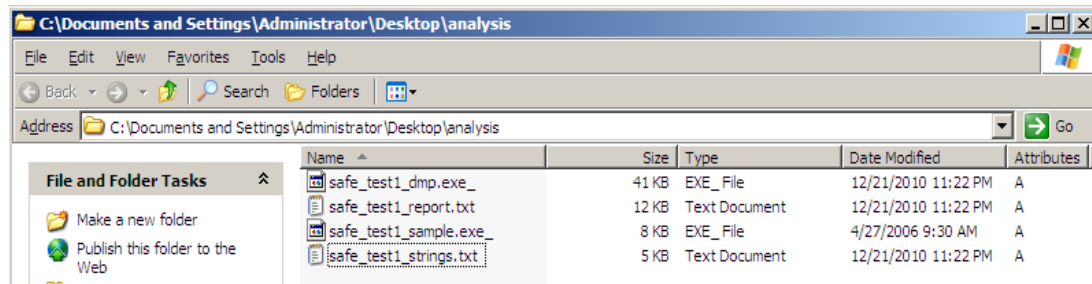
Para conseguir este reporte, SysAnalyzer toma una imagen del sistema, luego ejecuta el binario y luego vuelve a tomar otra imagen del sistema y muestra las diferencias en un reporte como se mostró previamente y el un conjunto de solapas como a continuación se puede visualizar:



En el caso puntual del binario analizado (una vez más corresponde al “safe_test1.exe”) se puede visualizar mediante la herramienta que el mismo levanta un proceso y un socket, genera dos claves en el registro con el valor “evil”, genera un archivo denominado “test.exe” y realiza numerosos accesos al Registry.

Además de ello y para colaborar con el análisis también genera una carpeta denominada “analysis” por cada binario analizado donde permite guardar el reporte generado, coloca las strings de disco del binario y hace un dump del mismo de memoria. Este dump (dependiendo del empaquetado utilizado sobre el binario) puede ayudar a saltar el empaquetado sin siquiera conocer

que técnica se utilizó si es que se piensa realizar una pericia de caja blanca sobre la pieza de malware a posteriori.



Conclusiones

A lo largo del presente trabajo se tuvo la posibilidad de presentar diversos enfoques para la detección y posterior eliminación de los Bots y las Botnets que estos conforman.

El objetivo general de este trabajo que se planteó al comienzo del mismo es el siguiente: “Proporcionar un conjunto de herramientas y técnicas forenses adaptándolas para la detección e identificación de Botnets, y ofrecer una guía de implementación de las herramientas más significativas.”; por su parte la hipótesis en la que se inscribe este trabajo es la siguiente: “Se pueden detectar Botnets mediante la utilización de las herramientas Snort, Ourmon, y Nepenthes y un conjunto de herramientas y técnicas forenses para la identificación de los Bots instalados en los hosts.”

Existe una gran variedad de formas de luchar contra las Botnets utilizando un sin número de herramientas y técnicas distintas, sin embargo no existe un enfoque global para encarar el problema. Con la idea de cumplir el objetivo del trabajo pero buscando un enfoque innovador se plantearon a lo largo del escrito un conjunto de enfoques metodológicos que permiten encarar la problemática de forma universal desde todos sus ángulos. Para ello se propuso un enfoque Top-Down que va desde lo general a lo particular, es decir, que planteado en un ambiente organizacional donde se desarrolla la problemática el enfoque consiste en discriminar cuales hosts podrían estar infectados por un Bot y luego proceder a aislar los mismos y someter los hosts uno por uno a una pericia para extirpar así el Bot de los hosts.

Este enfoque Top-Down se divide a su vez en dos partes bien diferenciadas, (cada una con sus sub-enfoques específicos) permite validar ampliamente la hipótesis propuesta. Para "discriminar cuales hosts podrían estar infectados por un Bot" se plantean tres sub-enfoques metodológicos expuestos en los capítulos 1, 2 y 3 respectivamente. Sin entrar en detalle sobre estos sub-enfoques, se puede afirmar que dos de los mismos se encuentran basados en el análisis de anomalías del tráfico de red (utilizando las herramientas Snort, Ourmon) para discriminar así el tráfico organizacional normal del que no lo es y el restante se encuentra basado en la utilización de Honeypots

(empleando Nepenthes) para poder capturar los Bots que circulen en la organización dado que estos exhiben el comportamiento de un Worm como se mencionó al principio del trabajo. Para "someter los hosts uno por uno a una pericia para extirpar así el Bot de los hosts" se plantea una metodología que sigue el enfoque forense y que mediante una serie de pasos permite arribar a detectar cual fue la amenaza específica y así poder eliminarla u opcionalmente someterla a un estudio más profundo. En síntesis puede decirse que este trabajo presenta distintas formas de luchar contra las Botnets que pueden ser analizadas y combinadas de distintas maneras de forma de generar el esquema que le sea más satisfactorio a cada organización.

Anexos

Anexo A: Instalación de Snort y componentes periféricos [2] [5] [9] [13]

Para instalar y configurar MySQL se deben ejecutar los siguientes comandos:

```
# apt-get update
```

```
# apt-get install mysql
```

```
# /etc/init.d/mysqld start
```

```
# mysql -u root
```

```
mysql> set password for \
```

```
'root'@'localhost'=password('su_contraseña');
```

Con esto finaliza la instalación y configuración inicial de MySQL; a continuación se debe bajar el último paquete de Snort desde www.snort.com y luego instalarlo siguiendo este procedimiento:

```
# apt-get install build-essential
```

```
# apt-get install libnet1-dev
```

```
# apt-get install libpcap0.8-dev
```

```
# apt-get install libpcrc3-dev
```

```
# apt-get install libmysqlclient16-dev
```

```
# apt-get install checkinstall
```

```
# tar xvft snort-2.1.x.tar.gz
```

```
# cd snort-2.1.x
```

```
# ./configure --enable-flexresp --with-mysql
```

```
# make
```

```
# make install (This last one as root)
```

```
# mkdir /var/log/snort
```

```
# useradd snort -d /var/log/snort -s /bin/false -c SNORT_IDS
```

```
# chmod snort /var/log/snortsnort/
```

Luego se debe configurar Snort para que funcione con las reglas con las que ya viene:

```
# mkdir /etc/snort/rules
```

```
# tar zxvf snortrules-snapshot-CURRENT.tar.gz
```

```
# cp snortrules-snapshot-CURRENT/etc/* /etc/snort/
```

```
# cp snortrules-snapshot-CURRENT/rules/* /etc/snort/rules/
```

```
# chown -R snort /etc/snort/
```

```
# chmod -R 775 /etc/snort/
```

```
# cp snortrules-snapshot-CURRENT/so_rules/precompiled/Ubuntu-6.01.1/i386/2.8.4.1/* /usr/local/lib/snort_dynamicrules/
```

```
# chown -R snort /usr/local/lib/snort_dynamicrules/
```

```
# chmod -R 775 /usr/local/lib/snort_dynamicrules/
```

Luego se debe crear y configurar una Base de Datos en MySQL donde almacenar los logs de Snort:

```
# mysql -u root -p
```

```
mysql> create database snort;
```

```
mysql> grant all on snort.* to snortuser@localhost identified by 'snortpassword';
```

```
mysql> flush privileges;
```

```
mysql> exit;
```

```
mysql> connect snort;
```

```
mysql> source /usr/local/src/snort-2.1.x/contrib/create_mysql;
```

Ahora faltaría configurar Snort para que coloque los logs en esta base recién creada, para ello se debe modificar el archivo snort.conf como se mostrará en la siguiente sección.

A continuación se procederá a la instalación de Apache y PHP que es sencilla si se hace desde el repositorio como se muestra a continuación:

```
# apt-get install apache
```

```
# apt-get install php
```

Finalmente resta instalar ACID Base para poder visualizar los logs de Snort almacenados en MySQL a través de una interfaz web. ACID Base es en realidad una página web programada en PHP que lee los logs de Snort desde MySQL y los muestra vía web. Para poner en funcionamiento esta interfaz web primero se debe bajar la última versión de ACID Base (<http://base.secureideas.net/>) y de ADOdb (<http://adodb.sourceforge.net/>) y luego se debe proceder del siguiente modo:

```
# tar -xvf base-x.x.tar.gz
```

```
# mv /home/user/Desktop/base-1.2.5 /var/www/base/
```

```
# tar -xvf adodb490.tgz
```

```
# mv /home/user/Desktop/adodb /var/www/base
```

```
# chown -R www-data /var/www/base/
```

```
# mv /var/www/base/base_conf.php.dist /var/www/base/base_conf.php
```

```
# vi /var/www/base/base_conf.php
```

```
> $DBlib_path="./adodb";
```

```
> $DBtype="mysql";
```

```
> $alert_dbname = snort;
```

```
> $alert_host = localhost;

> $alert_port = "";

> $alert_user = snortuser;

> $alert_password = snortpassword;

> $archive_dbname = snort;

> $archive_host = localhost;

> $archive_port = "";

> $archive_user = snortuser;

> $archive_password = snortpassword;

# mysql -u root -p snort < /var/www/base/sql/create_base_tbls_mysql.sql

# apt-get install php5-gd

# /etc/init.d/apache2 restart

# apt-get install php-pear

#pear install --force Image_Color

#pear install --force Image_Canvas

#pear install --force Image_Graph

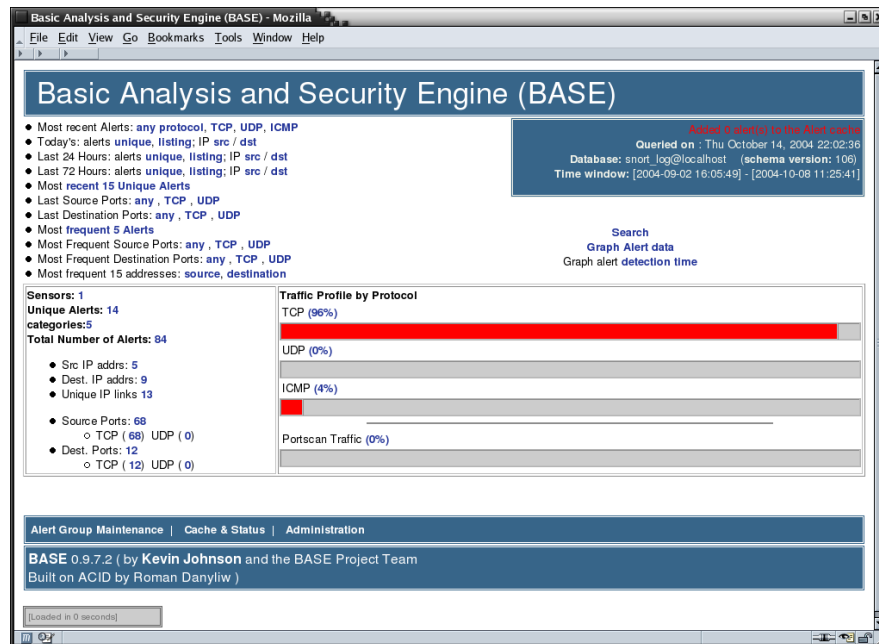
# vi /var/www/base/base_config.php

> $resolve_IP= 1;

> $colored_alerts = 1;

> $priority_colors = array
('FF0000','FFFF00','FF9900','999999','FFFFFF','006600');
```

Una vez realizados los pasos anteriores ya se puede entrar a la página web de ACID Base; para ello se debe escribir <http://localhost/base> en nuestro navegador web preferido y una página similar a esta debería aparecer:



Aun no se podrá visualizar ningún paquete logeado puesto que todavía hace falta modificar el archivo snort.conf para hacer que Snort derive sus logs a la base Snort de MySQL que será leída por ACID Base. Más adelante se explicará cómo realizar dicha configuración además de cómo explotar las funcionalidades de esta interfaz web para la detección de Botnets.

Anexo B: Instalación y configuración de NTOP y Wireshark

NTOP es una herramienta que permite analizar el tráfico pasante por un host donde se encuentra instalada; dicho análisis se puede hacer mediante una GUI a la que se accede por Web

Wireshark es el sniffer “de facto” para analizar el tráfico de los protocolos red con gran detalle. El aplicativo cuenta con una versión GUI y una versión basada en texto denominada T-shark.

La instalación de NTOP y Wireshark en un entorno como Ubuntu es directa, a saber:

```
# apt-get install ntop
```

```
# apt-get install wireshark
```

El sistema resolverá automáticamente las dependencias; para ejecutar Wireshark hay que hacerlo como root para que pueda capturar el tráfico satisfactoriamente.

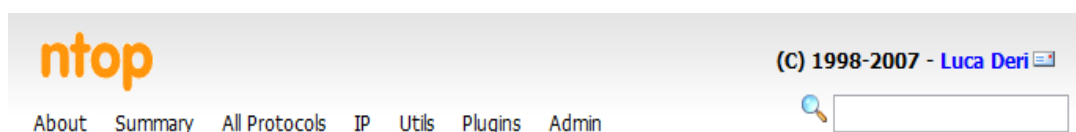
```
# sudo wireshark
```

Por su parte para ejecutar NTOP hay que ejecutar el siguiente comando:

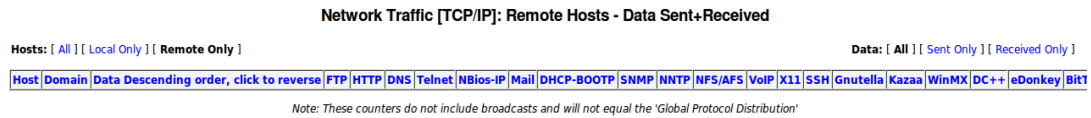
```
# ntop --daemon --sticky-hosts --user root
```

El comando anterior instruye a levantar NTOP como demonio con el usuario root y a no limpiar el caché en intervalos de tiempo fijo.

Luego de ejecutar el comando anterior, hay que dirigirse al explorador de Internet y escribir en la barra de direcciones lo siguiente: <http://127.0.0.1:3000> (o la IP del host donde se realizó la instalación) y entonces aparecerá la pantalla de NTOP.



NTOP también permite realizar un “corte del tráfico” según protocolo de nivel de aplicación y visualizarlo por su interfaz como se puede ver a continuación:



Los protocolos de nivel de aplicación que se visualizan en la imagen anterior son configurables utilizando el parámetro “--protocols” al momento de lanzar NTOP. La sintaxis de este parámetro es la siguiente: [--protocol <label>=<protocol list>]. Donde “label” es un nombre a elección representativo del protocolo y “protocol list” es un listado de puertos que corresponden a ese protocolo o nombres según aparecen en el archivo /etc/services. Un ejemplo sería el siguiente:

```
#      ntop      --daemon      --sticky-hosts      --user      root      --  
protocols="HTTP=http|www|https|3128,FTP=ftp|ftp-data"
```

Si no se utiliza el parámetro “--protocols” se tomarán y visualizarán los siguientes protocolos por defecto:

FTP=ftp|ftp-data

HTTP=http|www|https|3128

DNS=name|domain

Telnet=telnet|login

NBios-IP=netbios-ns|netbios-dgm|netbios-ssn

Mail=pop-2|pop-3|pop3|kpop|smtp|imap|imap2

DHCP-BOOTP=67-68

SNMP=snmp|snmp-trap

NNTP=nntp

NFS=mount|pcnfs|bwnfs|nfsd|nfsd-status

X11=6000-6010

SSH=22

Gnutella=6346|6347|6348

Kazaa=1214

WinMX=6699|7730

DirectConnect=0

eDonkey=4661-4665

Messenger=1863|5000|5001|5190-5193

Anexo C: Instalación y configuración de Nepenthes [15] [17]

En este anexo se detallará la instalación de Nepenthes en un entorno Linux (tipo Ubuntu/Debian) y su posterior configuración.

Antes de empezar a instalar Nepenthes cabe aclarar que este utiliza libcap1 que se encuentra actualmente deprecada en los repositorios por su nueva versión que es la libcap2; de esta forma hay que bajar libcap1 desde los repositorios de Debian y luego instalarla por separado; libcap1 se puede bajar desde: <http://packages.debian.org/lenny/i386/libcap1/download>. Luego se procede a la instalación del siguiente modo:

```
# sudo dpkg -i libcap1_1.10-14_i386.deb
```

Ahora sí, se puede instalar Nepenthes sin correr riesgo de no satisfacer las dependencias necesarias:

```
# sudo apt-get install nepenthes
```

Se puede apreciar que el instalador crea una cuenta de usuario y un grupo llamados 'nepenthes' para evitar levantar este demonio con root.

```
# cat /etc/passwd | grep nepenthes
```

```
nepenthes:x:113:121::/home/nepenthes:/bin/false
```

```
# cat /etc/group | grep nepenthes
```

```
nepenthes:x:121:
```

Los archivos de configuración de Nepenthes se encuentran en el directorio /etc/nepenthes y corresponden con las opciones de configuración los módulos del HoneyPot. A continuación se detallan los módulos por defecto:

```
# ls /etc/nepenthes
```

download-csend.conf	submit-file.conf	vuln-msdtc.conf
download-curl.conf	submit-gotek.conf	vuln-msmq.conf
download-ftp.conf	submit-http.conf	vuln-mssql.conf
download-link.conf	submit-mwserv.conf	vuln-mydoom.conf
download-tftp.conf	submit-norman.conf	vuln-netbiosname.conf
log-download.conf	submit-postgres.conf	vuln-netdde.conf
log-irc.conf	vuln-asn1.conf	vuln-optix.conf
log-prelude.conf	vuln-bagle.conf	vuln-pnp.conf
log-surfnet.conf	vuln-dameware.conf	vuln-sasserftpd.conf
module-honeytrap.conf	vuln-dcom.conf	vuln-sub7.conf
module-portwatch.conf	vuln-ftpd.conf	vuln-upnp.conf
nepenthes.conf	vuln-iis.conf	vuln-veritas.conf
shellcode-generic.conf	vuln-kuang2.conf	vuln-wins.conf
signatures	vuln-lsass.conf	x-2.conf

Se puede identificar a qué tipo de módulo se refieren los archivos mediante el análisis del nombre con el cual comienza el archivo:

Tipo de módulo	Descripción
vuln-*.conf	Módulos de simulación de vulnerabilidades
signatures/shellcode-generic.conf	Módulo de parseo de shellcode
download-*.conf	Módulos de captura
submit-*.conf	Módulos de sumisión
log-*.conf	Módulos de logeo
module-*.conf	Permite configurar características especiales de Nepenthes
x-2.conf	Una introducción a la implementación de módulos propios
nepenthes.conf	El archivo de configuración principal de Nepenthes

Los archivos .conf de los módulos tienen siempre la misma estructura:

```
Nombre_modulo{  
  
    <parámetro1>      <valor>  
  
    <parámetro2>      <valor>  
  
    <parámetroN>      <valor>  
  
};
```

Un ejemplo podría ser el módulo vuln-ftp:

```
# cat /etc/nepenthes/vuln-ftp.conf  
  
vuln-ftp {  
  
    ports      ("21");  
  
    accepttimeout  "45";  
  
};
```

Cada uno de estos módulos puede ser configurado independientemente, y dado la cantidad de ellos que existen aquí no se va a hablar sobre la configuración individual de cada uno (salvo del submit-norman como se verá más abajo); lo que sí cabe aclarar es que Nepenthes viene por defecto preparado para funcionar sin modificar ningún módulo.

El archivo nepenthes.conf no corresponde a la configuración particular de un módulo, sino que representa el archivo de configuración principal de Nepenthes, cuya apariencia es la siguiente (se expone solo un fragmento):

```
# cat /etc/nepenthes/nepenthes.conf  
  
// main configuration file for nepenthes  
  
// see also configuration files for modules (second row in the modules  
section)
```

nepenthes

```
{  
  
    moduledir          "/usr/lib/nepenthes";    // relative to workdir  
  
    moduleconfigdir    "/etc/nepenthes";        // relative to workdir  
  
    modules(  
  
// module name (in moduledir)    config file (in moduleconfigdir)  
  
// download handler for various protocols  
  
    "downloadcsend.so",    "download-csend.conf",    ""  
  
    "downloadcreceive.so",    "",    ""  
  
// submission handler  
  
    "submitfile.so",    "submit-file.conf",    "" // save to disk  
  
    "submitnorman.so",    "submit-norman.conf",    ""  
  
// "submitnepenthes.so",    "submit-nepenthes.conf",    "" // send to  
download-nepenthes in other nepenthes instances  
  
// "submitxmlrpc.so",    "submit-xmlrpc.conf",    "" // submit files to  
a xmlrpc server  
  
// "submithttp.so",    "submit-http.conf",    "" // submit files to a  
web server  
  
// logging  
  
    "logdownload.so",    "log-download.conf",    ""  
  
// "logirc.so",    "log-irc.conf",    "" // needs configuration  
  
// "logprelude.so",    "log-prelude.conf",    ""  
  
// "loghexdump.so"    ""    ""  
  
};
```

La idea principal del archivo de configuración es relacionar el core de Nepenthes con todos sus módulos activos (archivos con extensión .so) y sus correspondientes archivos de configuración (archivos con extensión .conf); por lo cual, si no se van a agregar o quitar módulos o sus archivos de configuración es conveniente no modificar este archivo. Una modificación que si vale la pena hacer es descomentar la siguiente línea:

```
"submitnorman.so",          "submit-norman.conf",      ""
```

Esto permite que Nepenthes envíe automáticamente los binarios capturados a la sandbox web “Norman Sandbox” y se devuelvan los resultados a una dirección de correo a elección; para lograr esto último también hay que editar el archivo de configuración del módulo de la siguiente forma:

```
# vi /etc/nepenthes/submit-norman.conf

submit-norman

{

// this is the address where norman sandbox reports will be sent

email "youraddresshere@yourdomain.com";

urls ("http://sandbox.norman.no/live_4.html",

"http://luigi.informatik.uni-mannheim.de/submit.php?action=

verify");

};
```

Anexo D: Instalación y configuración de Ourmon [18]

En este anexo se detallará la instalación de Ourmon en un entorno Ubuntu Linux y su posterior configuración:

```
# ./configure.pl
```

configuration script to install ourmon.

note: default is suggested like so: [default]

note: just hit carriage-return for default actions

Would you like to install the ourmon probe? [y]

Front-end configuration phase started #####

pcap in general needs to be reinstalled from www.tcpdump.org before ourmon install

found pcap lib in /usr/local/lib/libpcap.a - which is a good thing

hit CR to continue:

Would you like to compile/install ourmon? [y]

ourmon build: using make -f Makefile.bsd LIBS=/usr/local/lib/libpcap.a /usr/local/lib/libpcap.a

`ourmon' is up to date.

Next we determine the ourmon config/filter file to use.

By default, we use the local /usr/local/mrourmon/etc/ourmon.conf to provide input filters to ourmon.

WARNING: you should read/edit/understand ourmon.conf!

Do you want to use another ourmon.conf file in some other directory than /usr/local/mrourmon/etc? [n]

Next we suggest one modification to the ourmon.conf file.

If this is a default install, you should change the following config directive:

```
topn_syn_homeip network/netmask
```

and set it to your home network and mask (A.B.C.D/maskbits style)

Do you want to change the topn_syn home network address? [y]

note: the home net address may be a subnet or host address (/32).

enter a home net address and mask. [127.0.0.1/32] 192.168.1.0/24

netmask: 192.168.1.0/24

Do you want to install the ourmon startup script in the ourmon bin? [y]

WARNING: the default for the interface may not be what you want.

WARNING: use #ifconfig -a to determine interfaces.

Please enter the input interface name to sniff from: [eth0]

input interface is eth0

Please enter directory for probe output files (mon.lite, etc.):
[/usr/local/mrourmon/tmp]

probe output directory name is: /usr/local/mrourmon/tmp

Creating bin/ourmon.sh driver for startup of ourmon.

ourmon.sh placed in ourmon bin for ourmon front-end/probe startup

./ourmon.sh start

copy the startup script (bin/ourmon.sh) to /usr/local/etc/rc.d for boot startup?
[y]

ourmon front-end install complete

ourmon front-end build worked

You should now run /usr/local/mrourmon/bin/ourmon.sh to start ourmon

e.g., # /usr/local/mrourmon/bin/ourmon.sh start

You can use ourmon.sh stop to stop ourmon

part 2: install the back-end, omupdate.pl, etc. (web part)? [y]

Back-end	configuration	phase	started
#####			

We need a local web directory for generated web output.

hint: the webpath given here is a guess: give the CORRECT base web directory with /ourmon at the end

enter absolute web server web path directory: [/usr/local/www/data/ourmon]
/usr/local/www/ourmon

your output web path is: /usr/local/www/ourmon

Do you want to create the web directory for ourmon?

HINT: good idea if it doesn't exist. [y]

mkdir: /usr/local/www/ourmon: File exists

In: web.pages: File exists

cp bard/* /usr/local/www/ourmon/bard

cp batchip.sh batchipall.sh omupdate.sh /usr/local/mrourmon/bin

cp ombatch*.pl wormtolog.pl daily.pl monbackup.pl /usr/local/mrourmon/bin

cp omupdate.pl tcpworm.pl irc.pl /usr/local/mrourmon/bin

cp mklogdir.sh /usr/local/mrourmon/bin

chmod +x /usr/local/mrourmon/bin/*.sh

chmod +x /usr/local/mrourmon/bin/*.pl

INFO only: also setting up logging directory (if needed)

creating log rddata tmp dirs, if necessary, in /usr/local/mrourmon

hit CR to continue:

If different, enter front-end output file directory absolute path:
[/usr/local/mrourmon/tmp]

probe output file path (back-end input/s) is /usr/local/mrourmon/tmp

Now we copy supplied .html files to the web directory for later editing

do you want to copy base web files to the web directory? [y]

INFO only: setting up local rrdbase directory at /usr/local/mrourmon/rrddata

your runtime rrds get stored in this directory, along with the rrd error log file

if you create new BPF filters, check rrdbase/ourmon.log for errors.

hit CR to continue:

We need a UDP weight threshold for UDP scan alerts

what should the weight be (default is given): [10000000]

Install backend crontab commands in /etc/crontab (default answer y)?: [y]

y

ourmon system config complete

see INSTALL for post-config sanity checking

Bibliografía

[1] Wikipedia, Computer Worm, http://en.wikipedia.org/wiki/Computer_worm, fecha de captura: 05/06/2010

[2] Abbamonte Hernan / Devincenzi Juan / Giraldo Cesar / Sieradzki Adrian / Vigliocco Luciano, Implementación Práctica del NIDS SNORT, UBA – Especialización en Seguridad Informática – Seguridad en Redes II – Julio 2009

[3] Cox Kerry J. / Gerg Christopher, Managing Security with Snort and IDS Tools, O'Reilly Agosto 2004, Cap. 1 (Introduction)

[4] Cox Kerry J. / Gerg Christopher, Managing Security with Snort and IDS Tools, O'Reilly Agosto 2004, Cap. 3 (Installing Snort)

[5] Cox Kerry J. / Gerg Christopher, Managing Security with Snort and IDS Tools, O'Reilly Agosto 2004, Cap. 5 (The snort.conf File)

[6] Cox Kerry J. / Gerg Christopher, Managing Security with Snort and IDS Tools, O'Reilly Agosto 2004, Cap. 7 (Creating and Managing Snort Rules)

[7] Cox Kerry J. / Gerg Christopher, Managing Security with Snort and IDS Tools, O'Reilly Agosto 2004, Cap. 10 (Using ACID as a Snort IDS Management Console)

[8] Talabis Ryan, Honeypots 101: A Brief History of Honeypots, The Philippine Honeynet Project <http://www.philippinehoneynet.org> 2005, fecha de captura: 03/07/2010

[9] Babbin Jacob / Biles Simon / Orebaugh Angela D. , Snort Cookbook, O'Reilly Marzo 2005, Cap 1 (Installation and Optimization)

[10] Babbin Jacob / Biles Simon / Orebaugh Angela D. , Snort Cookbook, O'Reilly Marzo 2005, Cap 3 (Rules and Signatures)

[11] Hanna Christopher W., Using Snort to Detect Rogue IRC Bot Programs, GSEC Practical Version 1.4c Oct. 8, 2004, fecha de captura: 03/06/2010

[12] Giménez García María Isabel, Utilización de Sistemas de Detección de Intrusos como Elemento de Seguridad Perimetral, Almería – Universidad de Almería 2008, Cap. 2 (Snort)

[13] Giménez García María Isabel, Utilización de Sistemas de Detección de Intrusos como Elemento de Seguridad Perimetral, Almería – Universidad de Almería 2008, Cap. 3 (Instalación y Configuración)

[14] Devincenzi Juan, Introducción a los Honeypots de baja interacción, Tesis de especialización en Seguridad Informática – UBA Especialización en Seguridad Informática – Marzo 2010, Cap. 1 (Introducción a los Honeypots)

[15] Devincenzi Juan, Introducción a los Honeypots de baja interacción, Tesis de especialización en Seguridad Informática – UBA Especialización en Seguridad Informática – Marzo 2010, Cap. 2 (Nepenthes)

[16] Provos Niels / Holz Thorsten, Virtual Honeypots: From Botnet Tracking to Intrusion Detection, Addison Wesley Professional Julio 16, 2007, Cap. 1 (Honeypot and Networking Background)

[17] Provos Niels / Holz Thorsten, Virtual Honeypots: From Botnet Tracking to Intrusion Detection, Addison Wesley Professional Julio 16, 2007, Cap. 6 (Collecting Malware with Honeypots)

[18] Schiller Craig A. / Binkley Jim / Harley David / Evron Gadi / Bradley Tony / Willems Carsten / Cross Michael, Botnets: The Killer Web App, Syngress Publishing, Inc. 2007, Cap. 6 (Ourmon: Overview and Installation)

[19] Schiller Craig A. / Binkley Jim / Harley David / Evron Gadi / Bradley Tony / Willems Carsten / Cross Michael, Botnets: The Killer Web App, Syngress Publishing, Inc. 2007, Cap. 7 (Ourmon: Anomaly Detection Tools)

[20] Schiller Craig A. / Binkley Jim / Harley David / Evron Gadi / Bradley Tony / Willems Carsten / Cross Michael, Botnets: The Killer Web App, Syngress Publishing, Inc. 2007, Cap. 8 (IRC and Botnets)

[21] GuTi.my Network Security, Ourmon drops packets on 64 bit machine - Fixed, <http://www.gutizz.com/ourmon-drop-packets-on-64-bit-machine/> 4 de Abril, 2008, fecha de captura: 25/05/2010

[22] Linux Management, You have a network worm? » With Ourmon monitoring network (2), <http://www.eastman-watch.cn/you-have-a-network-worm-187-with-ourmon-monitoring-network-2/> 12 de Noviembre, 2006 – 12:00 am, fecha de captura: 25/05/2010

[23] Allen Harper / Shon Harris / Jonathan Ness / Chris Eagle / Gideon Lenkey / Terron Williams, Gray Hat Hacking, The Ethical Hacker's Handbook, Third Edition, The McGraw-Hill Companies 2011, Cap. 28 (Collecting Malware and Initial Analysis) y Cap. 29 (Hacking Malware)