MÁSTER UNIVERSITARIO DE INVESTIGACIÓN EN INGENIERÍA DE SOFTWARE Y SISTEMAS INFORMÁTICOS

ITINERARIO DE INGENIERÍA DE SOFTWARE (Código: 31105128)

Detección y bloqueo de botnets mediante la combinación de técnicas basadas en el tráfico de red

Enrique Ripoll Cervera
Director: José Antonio Cerrada Somolinos

Curso 2014/2015
Convocatoria de septiembre

Máster Universitario de Investigación en Ingeniería de Software y Sistemas Informáticos

Itinerario de Ingeniería de Software (Código: 31105128)

Título: *Detección y bloqueo de botnets mediante la combinación de técnicas basadas en el tráfico de red*

Rama de conocimiento: *Desarrollo de software seguro*

Enrique Ripoll Cervera

Director: José Antonio Cerrada Somolinos

Autorización

Autorizo a la Universidad Nacional de Educación a Distancia a difundir y utilizar, con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firma del/los Autor/es

Resumen

Una botnet es un conjunto de ordenadores infectados por un malware, que se denominan zombis, y que son de ejecutar tareas de forma coordinada bajo las ordenes de un operador sin conocimiento del usuario legítimo. Los ordenadores zombis reciben las ordenes y lanzan los ataques a través de la red. En este trabajo se construye un sistema para detectar y bloquear los ataques a otros ordenadores basándose en el tráfico que generan.

Palabras clave

Botnet, bot, virus, troyano, worm, malware, exploit, bloqueo de botnets, detección de botnets, vulnerabilidad, captura y análisis del tráfico de red, DDoS, C&C, Command and Control

Contenido

Tabla de ilustraciones

1. Introducción

En estos últimos años, hemos asistido a un cambio en la forma de conectarnos a Internet. Hemos pasado de necesitar un ordenador enganchado a un cable a tenerlo disponible en el teléfono móvil que llevamos en el bolsillo, permitiéndonos una conexión permanente sin necesidad de estar en un lugar específico.

Esto ha traído consigo el aumento de usuario y dispositivos conectados. En 2014, tres cuartos de la población española entre los 16 y los 74 años ha accedido a Internet en los últimos tres meses [1].

La conexión se ha convertido en casi una necesidad. Por Internet hacemos operaciones bancarias, trámites con la Administración, compras de artículos varios e incluso vida social.

Todo este mundo de servicios online implica que nuestros datos residen en ordenadores conectados a Internet. Datos en algunas ocasiones especialmente delicados, como es el caso de los datos bancarios. Además, lleva aparejado el aumento exponencial en el número de máquinas conectadas.

Los datos bancarios, las identidades, la potencia de cálculo y el ancho de banda son elementos que tienen un valor monetario. Con un usuario y una contraseña podemos hacer transferencias bancarias o consultar el número de las tarjetas de crédito, con el que se pueden hacer compras o crear una tarjeta plástica. Las identidades se pueden utilizar para enviarnos publicidad. La potencia de cálculo permite hacer minería de bitcoins. El ancho de banda permite enviar correos electrónicos a otros usuarios o enviar tráfico hacia objetivos concretos con el objetivo de saturar su ancho de banda.

Es aquí donde surgen las botnets como una herramienta para obtener dinero a partir de los usuarios y máquinas conectadas a Internet. Las botnets, aunque se pueden utilizar como un tipo de arquitectura para un software legítimo, se

asocian a un tipo de malware que puede, aprovechando las redes de comunicaciones, llevar a cabo las anteriores operaciones a gran escala de forma automática. Este malware permite tener a miles de ordenadores trabajando a las órdenes de un individuo.

Los antivirus detectan y eliminan malware, pero su forma de detección, normalmente basada en firmas, impide la detección temprana, llamada de día cero, de muchas amenazas.

Las herramientas de detección basadas en la monitorización del tráfico de red, los llamados Sistemas de Detección de Intrusos, más conocidos por sus siglas en inglés, IDS, también permiten su detección. Es el caso del software de libre distribución SNORT o firewalls de empresas comerciales, como CISCO o f5. Sin embargo, no están exentos tampoco de problemas, bien por su forma de detección basada en firmas, o bien por su elevado precio que dificulta su implantación en un entorno doméstico.

Desde la instalación del malware en el ordenador o dispositivo hasta su detección por el antivirus o el IDS pueden pasar horas o incluso días. Tiempo más que suficiente para que el malware robe información privada o participe en ataques distribuidos de denegación de servicio.

Habitualmente se compara a las botnets con los ejércitos [2]. La potencia de un ordenador doméstico, al igual que un único soldado, es pequeña a escala mundial, sin embargo, cuando se unen miles de ordenadores domésticos, su potencia, al igual que un ejército, se ve multiplicada. Una botnet típicamente está formada por miles de ordenadores conectados a Internet, empresariales y domésticos.

El número de botnets ha ido creciendo con el transcurso de los años y sus ataques siguen produciendo pérdidas de datos y la dificultar en el acceso a los servicios por los usuarios legítimos. Como ejemplo sirven los ataques basados en

la denegación de servicio, lanzados de forma distribuida y conocidos como DDOS [3], como los siguientes:

- En septiembre de 2010 se llevó a cabo la “Operation Payback”. Un ataque de DDOS liderado por el grupo Anonymous y publicitado en los medios, que dejó fuera de juego a los sitios que defendían los derechos de autor y a sitios asociados con los primeros. El ataque se extendió durante varios días.
- Los sitios web gubernamentales y bancarios de Corea del Sur sufrieron, en Marzo de 2011, un ataque de DDOS que los dejó inaccesibles durante 10 días. Según MacAffee, este ataque se produjo desde una botnet concentrada mayormente en la misma Corea del Sur [4].

Estos ataques van a continuar produciéndose, entre otros motivos, porque se puede comprar un ataque de DDOS de una semana de duración por 150$ en la Darknet [5]. Por tanto, no hay que limitarse a la protección de grandes redes locales de empresas. Se requieren mecanismos que permitan instalar en el hogar, de forma sencilla, un sistema de detección de botnets en la red interna que proteja a los usuarios internos y externos de estas amenazas.

2. Objetivos

Los objetivos de este trabajo son:

1. Crear un entorno seguro que permita la ejecución y análisis de botnets.
2. Conseguir muestras de botnets.
3. Analizar las botnets e identificar los patrones el comportamiento.
4. Analizar las técnicas de detección de botnets que han sido propuestas o desarrolladas.
5. Construir un sistema que permita la mejora en la detección de botnets y el bloqueo de sus ataques en tiempo real.

Se tendrá en cuenta el usuario objetivo del sistema, que en algunos casos tiene pocos conocimientos de redes informáticas, por lo que el sistema debe ser sencillo de poner en funcionamiento para permitir su difusión en entornos domésticos.

3. Alcance

El sistema que se construya no detectará malware que se ejecute en las máquinas cliente y que no genere tráfico en la red.

Se deberá de disponer de algún mecanismo en la red local que permita capturar el tráfico de todas las máquinas.

Se estudiarán los patrones de comportamiento de las botnets, no de otro tipo de malware de red como pueden ser los worms.

4. Proyectos relacionados y forma de detección

Existen diversas investigaciones y proyectos relacionados con la detección de botnets. Bothunter [6] es un algoritmo que detecta los bots en base al análisis del tráfico que red que produce un bot y que delata su presencia, como la comunicación con sitios de C&C y la descarga de binarios, aunque requiere de máquinas potentes para funcionar en tiempo real. Botminer [7] detecta la actividad de bots por medio de la clasificación de anomalías y requiere que haya un grupo de máquinas comprometidas, con las que forma un cluster, para su detección. OSSIM, de la empresa AlienVault, es un SIEM[1] de libre distribución que proporciona de forma integrada la monitorización y gestión de eventos de seguridad. Es un software muy completo que requiere conocimientos técnicos para su instalación y explotación. Ourmon [2] es un desarrollo que detecta anomalías en el tráfico de red y realiza la detección según unos parámetros establecidos. Es un motor de análisis y visualización de eventos de red de los logs que se le pasan y requiere la intervención el usuario para aislar la máquina que contiene el bot. Asimismo, existente varias soluciones comerciales pensadas para la protección de redes empresariales, como por ejemplo, DefensePro [8], de la empresa Radware, para protegerse frente a los ataques DDoS.

[1] Security Information and Event Management. OSSIM es un software que proporciona de forma integrada la gestión de la seguridad de la información (SIM) con la gestión de eventos de seguridad (SEM).

5. Descripción de las botnets

5.1.Introducción

Una botnet se puede definir como un conjunto de ordenadores infectados que trabajan bajo las ordenes de un individuo, llamado botmaster [9]. Los ordenadores están conectados a Internet y el usuario no es consciente de que su máquina está ejecutando procesos ajenos.

5.2.Historia de las botnets

Un año después de crearse el protocolo IRC y establecerse como una forma de comunicación entre los usuarios, en 1989, apareció el primer bot [2]. Se llamaba Hunt the Wumpus y permitía interactuar con el juego a los usuarios IRC por medio de comandos en la consola del cliente de IRC.

Los primeros bots se crearon con fines lúdicos y como ayuda a los operadores de IRC en las labores de administración de los canales. Automatizaban tareas de gestión de usuarios y canales y controlaban el buen uso por parte de los usuarios. Los bots fueron aumentando sus funcionalidades, actuando como administradores de los canales con capacidades para añadir y expulsar usuarios. Eran bots que se ejecutaban como servicios en el servidor IRC.

Pretty Park, aparecido en Mayo de 1999, es considerado como el primer bot que se ejecuta en la parte cliente del IRC [10]. Tenía la capacidad de conectarse a un servidor y recoger información sobre el sistema, la posibilidad de actualizarse, subir y bajar archivos y de lanzar ataques DoS. Estas funcionalidades se han heredado en las botnets posteriores.

Posteriormente, fueron aparecieron varios troyanos que explotaban vulnerabilidades de los sistemas. Tal fue el caso de SubSeven, un troyano que se conectaba controlaba remotamente y permitía robar información, como credenciales y pulsaciones de teclas.

Con la aparición del software cliente de IRC mIRC, que permitía la generación de scripts y la creación de sockets UDP y TCP, aparecieron nuevas amenazas. En el 2000 entró en escena GT Bot [2], que se ejecutaba sobre mIRC y permitía escanear puertos, lanzar ataques DDoS, clonar conexiones y hacerlas anónimas. GT Bot no incluye mecanismos para propagarse. Se basa en la ingeniería social, normalmente mediante el envío de un correo electrónico con un link, para que el usuario descargue y ejecute el archivo que lo instala localmente.

En el 2002 apareció SDBot. Su creador, un programador ruso, publicó el código fuente, lo que facilitó su modificación por otros programadores. Tiene varios mecanismos de contagio automático, aprovechando vulnerabilidades de los sistemas operativos y entrando por puertas abiertas por troyanos ya instalados en el cliente. Para infectar a un ordenador primero se instalaba un pequeño ejecutable que se encargaba de descargar el archivo con la funcionalidad completa del bot y ejecutarlo. El fin inicial era lanzar ataques DoS, aunque sus modificaciones posteriores han ido ampliando sus capacidades y variando su forma de infección.

Agobot, también conocido por Gaobot, que apareció en el mismo año, aportó a la escena malware la modularidad. Está formado por tres módulos especializados, que se encargan respectivamente de lanzar el cliente bot y abrir la puerta de acceso remota, matar los procesos del antivirus, e impedir el acceso del usuario a determinados sitios web, como los de las compañías antivirus.

Estas primeras botnets han inspirado las botnets que se han ido desarrollando posteriormente, aprovechando su código fuente o su arquitectura, ampliándolas con nuevas formas de comunicación o explotando nuevas vulnerabilidades que han ido apareciendo.

5.3.Arquitecturas

Las botnets están formadas siempre por una parte cliente, que se instala en los ordenadores a controlar y una parte servidora, desde la que se gestiona a los clientes. A la parte servidora se le llama comúnmente C&C, del inglés Command-

and-Control, y permite, como su nombre indica, controlar a la red de clientes y lanzar las ordenes.

Existen distintas formas de implementar el C&C, pudiéndose clasificar según el protocolo empleado en la comunicación: IRC, HTTP/HTTPS y P2P [11]. Algunos investigadores apuntaron a la posibilidad de utilizar los protocolos de Skype [12] y VoIP [13] y se han llegado a hacer implementaciones del C&C utilizándolos, pero el funcionamiento es muy similar a los anteriores.

5.3.1. Control por canal IRC

Es una topología centralizada en la que el control de la red se lleva a cabo utilizando el protocolo IRC. Cada cliente, tan pronto se activa, se conecta a un canal IRC predefinido de un servidor, público o privado, desde donde recibe las órdenes del botmaster. Al conectarse, informa de su situación. El botmaster tiene a su disposición una lista de órdenes que son entendibles por los clientes.

Agobot utiliza esta técnica. Para lanzar un ataque DDoS mediante ICMP, el botmaster introduciría una frase con el siguiente formato en el canal IRC: 'ddos.phaticmp [host] [time] [delay]'. Con SDBot, sería 'ping [host] [num] [size] [delay] num'. Los clientes conectados al canal, al recibir esta frase, que en realidad es una orden, lanzarían un ataque ICMP a una dirección IP (host) con los parámetros indicados. En ocasiones, la comunicación se lleva a cabo encriptada.

Este sistema de control está muy extendido al permitir los routers la comunicación IRC, ya que es un protocolo que en la actualidad se sigue utilizando en los juegos en línea como forma de comunicación en modo texto entre los jugadores. Además, permite un control rápido de la red de bots, se conoce en tiempo real el número de clientes mirando los usuarios que hay conectados al canal y permite lanzar órdenes dirigidas a un cliente en particular.

5.3.2. Control por HTTP

El control por HTTP o HTTPS se utiliza en una topología centralizada en la que el botmaster publica las ordenes en un servidor web y los clientes, regularmente,

se conectan al servidor y comprueban que tarea tienen que llevar a cabo. La comunicación se suele llevar a cabo encriptada. El protocolo HTTP es el protocolo que permite la navegación web, por lo que los routers lo suelen permitir.

En este caso, el servidor web se convierte en un único punto de fallo. Bloqueando su acceso, se impediría el funcionamiento de la red. Por ello, han surgido técnicas que permiten aumentar la supervivencia de la botnet, como el uso de varios servidores en paralelo, la generación de nombres de dominio mediante un algoritmo o el fast-flux DNS.

El uso de varios servidores permite eliminar el servidor C&C como un único punto de fallo. Actualmente se suele utilizar combinado con las otras técnicas.

El gusano Conficker A utiliza la técnica de generación de nombres de dominio mediante algoritmo para evitar el bloqueo en los firewalls y routers. Entra en un bucle infinito que genera una lista de 250 nombres de dominio. La función de generación de nombres se basa en una función de generación de nombres aleatorios que utiliza como semilla la hora UTC actual del sistema. La misma lista se genera cada 3 horas. Todos los clientes, con los relojes sincronizados en la hora UTC, calcularán la lista de nombres de dominio e intentarán contactar con cada uno de ellos [14].

La técnica de fast-flux establece un mapeo uno-a-muchos entre una entrada DNS y varias direcciones IP. El mapeo cambia a una velocidad muy rápida por lo que la consulta DNS devuelve una IP que cambia dinámicamente en cada consulta. El botmaster asigna un numero de bots fuertes computacionalmente con direcciones IP fijas como flux-agents. En lugar de comunicarse con el servidor C&C directamente, los bots se comunican con estos agentes, que redirigen el tráfico al verdadero servidor C&C [15]. Esta técnica permite enmascarar al servidor, haciendo imposible su localización desde los clientes que se conectan a los flux-agents. Se elimina el punto de fallo cambiando la arquitectura de comunicación, convirtiendo la topología centralizada en una topología híbrida.

Existen botnets que como servidor C&C utilizan redes sociales, como pueden ser Twitter, que lo utiliza TwitterNET [16], y Facebook, utilizado por Whitewell Trojan [17]. En estos casos, el control se realiza mediante la publicación de un nuevo mensaje en la red social con la orden que deben ejecutar los clientes. Esta técnica se combina con la generación aleatoria de nombres de usuario para la red social.

5.3.3. Control por P2P

El control por medio de P2P crea una topología distribuida. La estructura y la forma de comunicación son similares a las de las redes P2P legítimas o en ocasiones, incluso se hace uso de ellas, para utilizar los mismos puertos y que el tráfico no sea sospechoso. Para su control, el botmaster envía un archivo a la red con las ordenes, que se irá propagando a todos los clientes por medio del protocolo P2P.

Nugache es un ejemplo de botnet que utiliza esta técnica de C&C distribuido y que apareció en Mayo de 2006 [18]. Su funcionamiento básico consiste en guardar en los clientes una lista de pares para comunicarse entre ellos y, una vez establecida la comunicación, se envían una clave RSA que utilizan para encriptar los mensajes que intercambia la red [19].

5.3.4. Otras técnicas

Otros mecanismos que también se han utilizado incluyen darknets y servicios en la nube, como el mail de Yahoo, Google Docs y Evernote [20].

5.4.Formas de infección

Para que una botnet se extienda, debe tener una forma de infección eficaz. Existen varias formas de infección, con intervención o sin intervención del usuario. Las que requieren intervención del usuario suelen utilizar técnicas de ingeniería social para lograr infiltrarse en el ordenador.

La infección por vulnerabilidades utiliza errores de programación de aplicaciones o sistemas operativos para instalarse en el sistema sin que el usuario se percate

de ello. La ejecución de código por desbordamiento de buffer es una de las vulnerabilidades más conocidas para la instalación de malware. SDBot utiliza varias vulnerabilidades de software [21] como una de las formas de propagarse.

Cuando la instalación del malware requiere la intervención del usuario para su ejecución, una de las técnicas de propagación que se suele utilizar se basa en el envío masivo de emais. El email recibido contiene un texto y un adjunto que resulta inocente para el usuario, como puede ser la simulación de una factura. Al ejecutar el adjunto, se instala el malware. El mail en ocasiones lleva un enlace, en lugar de un adjunto, que abre la página web que contiene el malware. En estos casos, se instala explotando vulnerabilidades del explorador o alguno de sus componentes o, de forma más simple, descargando un archivo que el usuario deberá ejecutar. La mensajería instantánea también ha sido utilizada como medio para propagar malware [22], actuando de forma similar al correo electrónico.

Los botnets en ocasiones utilizan vulnerabilidades de las aplicaciones para propagarse. En este caso, su uso es muy diverso y está condicionado al lanzamiento de un parche por parte del fabricante que la corrija. El lanzamiento de parches también se utiliza por parte de los desarrolladores de malware para lanzar vectores de infección que exploten esa vulnerabilidad en equipos que no están actualizados. Hay multitud de ejemplos de vulnerabilidades que han sufrido o sufren los distintos sistemas operativos [23], las máquinas virtuales Java [24], Adobe Flash Player [25], Adobe Acrobat Reader [26] y los navegadores de Internet [27], que se corrigen mediante parches y versiones nuevas.

La instalación de software supuestamente legítimo en un ordenador se puede aprovechar para instalar malware sin ser detectado. En esta ocasión, el paquete de software no se descarga del sitio oficial y contiene, además de la funcionalidad correcta, el malware, que habitualmente va unido al instalador del software. De forma transparente para el usuario, el malware se instala de forma silenciosa previo a la instalación del software legítimo. Se utiliza en aplicaciones

freeware, shareware y comerciales obtenidas de sitios web no fiables, así como complementos a estas, como generadores de claves o supuestos crackeadores de aplicaciones.

5.5.Funcionalidades

Las botnets disponen de distinta funcionalidad, adaptada por su desarrollador según sus necesidades.

El ataque DDoS es la funcionalidad más conocida de las botnets. Este ataque aprovecha mensajes del protocolo TCP/IP inocuos de forma individual. Los mensajes lanzados de forma simultánea desde miles de ordenadores, provocan la saturación de los recursos del host atacado, bien por saturación de la red o bien por saturación de los recursos de procesador o memoria. En ocasiones también se le llama ataque por inundación (flood). Hay distintas formas de lanzar un ataque DDoS: SYN, UDP, ICMP (ping) y HTTP son los habituales. SDBot dispone de los ataque DDoS SYN, UDP y ICMP.

El robo de claves es una funcionalidad presente en diversos botnets. El módulo de robo de claves puede estar programado de diversas formas según la funcionalidad deseada aunque su base es un key logger. Se puede programar para detectar la URL a la que se accede mediante el navegador y recoger los campos usuario y contraseña para obtener acceso a servicios de correo electrónico, redes sociales, juegos online o bancos, entre otros. Los servicios de correo electrónico, a su vez, se pueden utilizar para enviar spam. Los key loggers también son capaces de obtener las contraseñas de acceso a zonas restringidas que utilizan imágenes de teclados para introducir los caracteres. Spybot es capaz de capturar una pequeña porción de la imagen sobre la que se hace click [2], a la que luego se le puede pasar un OCR para obtener de forma automática el carácter introducido. El robo de claves no se limita al acceso a zonas restringidas, también se pueden llegar a robar los datos personales del usuario y su tarjeta de crédito cuando los introduce en alguna compra online o al acceder a servicios bancarios.

La publicidad no deseada es una funcionalidad molesta para el usuario, aunque normalmente no dañina. Se basa en mostrar al usuario productos de venta online o servicios de sitios web para que acceda a ellos. Se implementa mostrando ventanas emergentes del navegador o redirigiendo al sitio web en cuestión sin intervención del usuario cuando se abre el navegador, cuando se accede a una URL o de forma aleatoria.

Los ataques por fuerza bruta consisten en lanzar de forma coordinada peticiones de entrada mediante usuario y clave a algún servicio web, como puede ser un servidor de correo electrónico. Los usuarios se obtiene mediante la captura del email o el uso de usuarios predefinidos en el sistema operativo atacado, como root o administrator. Para la generación de las claves se suelen utilizar diccionarios de claves combinados con la generación aleatoria. Una vez obtenida la combinación usuario/clave, se envía a los demás bots que utilizarán la cuenta para el envío de spam.

El envío de spam es un servicio que consiste en utilizar una cuenta de correo electrónico para enviar mensajes de forma indiscriminada a otras cuentas de correo electrónico. Para el envío se utilizan las claves de una cuenta de correo electrónico obtenidas por la botnet mediante alguno de los mecanismos de obtención de claves, como key loggers o fuerza bruta.

La generación de bitcoins consiste en robar potencia de cálculo del ordenador infectado para realizar cálculos que permiten obtener bitcoins[2]. La potencia combinada de miles de ordenadores permite que el tiempo de cálculo para la obtención de un bitcoin disminuya.

Las botnets también se pueden utilizar para extorsionar al usuario. Existen muestras de malware que realizan la captura de imágenes por webcam de forma automática sin percatarse el usuario de ello. Posteriormente, se envía una muestra de esas imágenes al email del usuario, pidiéndole una cantidad de

[2] Al proceso de obtención de bitcoins por medio de cálculos matemáticos se le conoce como minería de bitcoins.

dinero a cambio de que no se publiquen en Internet. En otras ocasiones, realizan la encriptación de los archivos del disco duro y, de igual modo, solicitan una cantidad de dinero a cambio de la clave de desencriptación. A este tipo de malware se le conoce como ransomware.

El fraude de clicks utiliza el mecanismo de cobro por click que emplean algunas empresas para recompensar a las webs que muestran sus anuncios. En el caso de las botnets, los bots lanzan peticiones de forma aleatoria desde todos los clientes, simulando la visita a la web desde cada uno de los clientes y generando tantos clicks con cada ejecución, como bots tenga la botnet, lo que repercute en ingresos para el usuario de la botnet.

6. Laboratorio de análisis de botnets

6.1.Introducción

El análisis de botnets es una tarea que conlleva ciertos riesgos. Los botnets se propagan por red, por tanto, cualquier máquina que esté infectada y conectada a una red es una potencial transmisora del malware. Se hace necesario tomar precauciones, que son distintas según como abordemos el análisis:

- Mediante ingeniería inversa desensamblando su código. Se utiliza cuando se quieren obtener detalles de implementación, como puede ser el algoritmo de generación de nombres de dominio.
- Mediante su ejecución y análisis del comportamiento local. Es un complemento del primero que nos facilita la labor de interpretación del código.
- Mediante su ejecución y análisis del comportamiento en red. Permite obtener patrones de comportamiento que delaten su actividad en la red.

Como he centrado la detección de botnets en el análisis del tráfico de red, el laboratorio debe permitir que al ejecutar un cliente de botnet, se pueda capturar el tráfico de red que se genere.

El entorno de análisis ideal debe evitar la propagación del malware que está siendo objeto de estudio pero tiene que permitir que el malware exponga su comportamiento al ejecutarse. Asimismo, incorporará un simulador de la máquina de detección y bloqueo de botnets.

6.2.Diseño

Para analizar tráfico de red, lo primero que hay que hacer es capturarlo. Para ello, existen varios mecanismos:

- SPAN Port: también llamada port mirroring. Consiste en configurar un puerto del switch o router principal de la zona de red de la que queremos capturar el tráfico como SPAN Port. Esta funcionalidad hará que se

replique todo el tráfico de todos los puertos a ese puerto. No todos los switch disponen de esta característica.

- Tap: es un dispositivo especializado en la captura de tráfico que se conecta interceptando el troncal de la red que se quiere analizar y replica el tráfico a un tercer puerto. Podemos simular un Tap de forma rudimentaria mediante un hub, aunque algún autor lo considera un mecanismo independiente de captura de tráfico [28].

El análisis de ambos mecanismo aplicados a un entorno doméstico nos descarta ambas opciones:

- En el caso del SPAN Port, los routers domésticos no suelen incorporar esta funcionalidad, lo que implicaría que el mecanismo diseñado tendría un mercado limitado.
- En el caso de elegir el Tap, implicaría la compra de un dispositivo de red especializado que no es fácil de encontrar.

Se podría utilizar un hub, pero en este caso obligamos al usuario a adquirir un dispositivo adicional, trastear con los cables de red cambiando los cables del router al hub, y no se capturará el tráfico de la red inalámbrica si el router está prestando esta funcionalidad, únicamente se interceptará el tráfico de la red cableada, siendo esta última desventaja la que hace descartar esta opción.

Como se debe capturar todo el tráfico que viaja por el router, un posible mecanismo es estableciendo la funcionalidad de pasarela de la red en el dispositivo de análisis de tráfico, que dirigirá el tráfico recibido al router real una vez analizado. Como actúa de pasarela, recibirá todo el tráfico de salida de la red, tanto de la red cableada como de la inalámbrica.

La configuración física consistirá únicamente en una conexión Ethernet entre el dispositivo y cualquier boca Ethernet de la red, como puede ser una del switch del router. Para la configuración lógica y para evitar cambiar manualmente las pasarelas de los equipos, lo más sencillo es que los equipos obtengan la

configuración de red por DHCP, que es la configuración por defecto de los routers de los proveedores de Internet en España, y deshabilitar la funcionalidad de DHCP en el router principal. El dispositivo analizador de tráfico hará de servidor DHCP, entregando la configuración de red a los equipos que se conecten, con la configuración de pasarela adecuada para que pase el tráfico a través de él. Al analizador de tráfico se le deberá indicar manualmente la dirección del router principal para que sepa por donde debe enviar el tráfico de salida. Cabe la posibilidad de utilizar ARP spoofing con la dirección IP del router, evitando la reconfiguración de la red por DHCP, pero he desechado esta posibilidad porque están apareciendo en el mercado routers con la posibilidad de evitar este tipo de ataques [29]. Con esta configuración se consigue que con una única tarjeta Ethernet, se pueda capturar y bloquear el tráfico.

En la Ilustración 1 se detalla el diagrama de la red una vez conectado el analizador de botnets. En este caso, hay dos portátiles, tres teléfonos móviles y una impresora conectados de forma inalámbrica al router, y el dispositivo de análisis de botnets de forma cableada.

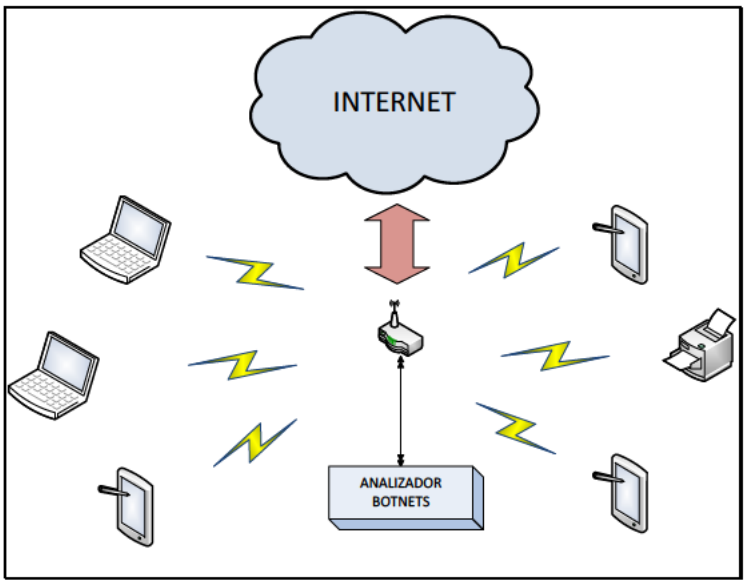


Ilustración 1. Diagrama de red del analizador de botnets

Cuando un nuevo dispositivo se conecte a la red, solicitará su dirección IP por DHCP. El analizador de botnets le dará una dirección IP libre y como pasarela, él mismo. Cuando el nuevo dispositivo envíe un paquete a Internet, primero irá al analizador de botnets, al ser la pasarela, se analizará, y se enviará al router, que a su vez, lo enviará por su conexión de banda ancha. Los paquetes de respuesta seguirán el camino inverso.

El laboratorio virtual debe permitir simular este entorno, por lo que dispondrá de un dispositivo que actuará de analizador de botnets, varios ordenadores conectados a la misma red y un router conectado a Internet.

El hardware para implementar el dispositivo analizador de botnets debe disponer de una tarjeta Ethernet y sería deseable que fuese multinúcleo para permitir analizar el tráfico de red en hilos paralelos de forma simultánea, con el fin de aumentar la velocidad. Como características adicionales, debería ser sencillo de obtener, económico, no muy voluminoso y de bajo consumo energético.

6.3.Implementación

Para montar el laboratorio virtual he utilizado un software de virtualización. Entre las alternativas que hay, he elegido Oracle VM VirtualBox porque es de libre distribución y permite crear una red de máquinas virtuales asiladas de la red del ordenador anfitrión[3] pero con posibilidad de comunicarse entre ellas.

La red virtual está formada por las siguientes máquinas:

- ROUTER: con dos tarjetas de red, una conectada a la misma red que la máquina anfitrión y otra conectada a la red virtual.
- BOTFENCE: simulará la máquina de captura del tráfico de red y su análisis para el bloqueo de ataques de botnets.
- KALI: máquina desde la que se simularán los ataques.

[3] Máquina que ejecuta el software de virtualización sobre el que corren el resto de máquinas.

El diagrama de red se puede ver en la Ilustración 2. Está compuesto por dos redes, la red virtual que crea el software de virtualización, con la máquina que servirá de plataforma de lanzamiento de ataques (KALI), la máquina que bloqueará los ataques (BOTFENCE) y el router (ROUTER VIRTUAL), que encaminará el tráfico hacia la red del ordenador anfitrión, desde donde el router de esa red (ROUTER PRINCIPAL) lo encaminará a Internet.

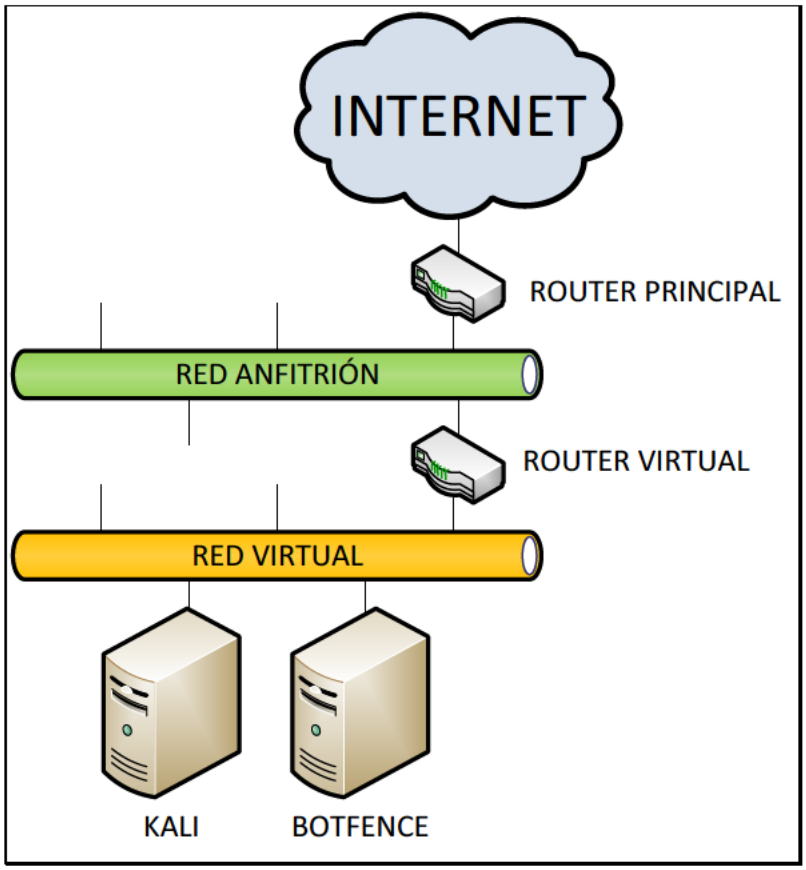


Ilustración 2. Diagrama de red del laboratorio virtual

El laboratorio permite ir añadiendo máquinas con distintos sistemas operativos para estudiar el comportamiento de las botnets en cada uno de ellos o utilizarlas para el análisis de vulnerabilidades. Al estar la salida a Internet localizada en un único punto, resulta sencillo deshabilitarla cuando se quiera evitar la propagación de malware.

El router de salida es una máquina virtual con un sistema operativo que permite hacer de pasarela. Únicamente debe ejercer de router, por lo que he utilizado

una de las minidistribuciones Linux especializadas en esta funcionalidad, OpenWrt y DD-WRT, que básicamente es un fork comercial de OpenWrt.

OpenWrt es una distribución Linux mínima que permite configurar una máquina como router mediante una interface gráfica. Aunque originalmente estaba diseñada y compilada para utilizarse como sustitución del firmware comercial de routers, actualmente posee una versión que puede instalarse en máquinas x86.

Para la máquina que utilizaré para lanzar los ataques he elegido la distribución de Linux llamada Kali Linux, de ahí el nombre utilizado para la máquina. Kali Linux es una distribución utilizada para auditorías de seguridad y hacking ético. Entre otras cosas, viene con las principales aplicaciones de explotación de vulnerabilidades.

En la máquina que controlará el tráfico de la red instalaré la distribución Debian de Linux. Como primera opción había seleccionado OpenWrt, pero finalmente he elegido Debian por los siguientes motivos:

- Es de libre distribución, no hay que adquirir licencias.
- Está gestionado por una comunidad de usuarios y tiene un ciclo rápido de actualizaciones y parches.
- Dispone de los paquetes Snort y Suricata para la detección de intrusos.
- Los pequeños ordenadores de bajo consumo, como la Raspberry PI, Banana PI o BeagleBone Black, disponen de una distribución basada en Debian, por lo que la diferencia con la máquina real debería ser mínima.

La primera máquina que instalo es BOTFENCE. Se podría instalar también primero KALI, pero no así ROUTER, ya que requiere acceder desde la red virtual por medio de un navegador web para finalizar su configuración

La máquina BOTFENCE tendrá 1 GB de RAM, un disco de 10 GB y una tarjeta de red conectada a la red virtual. La red virtual la he llamado lanbots. En esta máquina instalo una versión mínima de Debian. Una vez instalada, le añado los

paquetes que permiten compilar C, el gestor de ventanas Xfce, que es un gestor XWindow ligero, el paquete gdm para iniciar en modo gráfico y el servidor DHCP.

Aunque BOTFENCE dispone de una única tarjeta de red, al actuar de pasarela tiene que encaminar el tráfico en dos redes, una red que será el que lo una al router y otra red que lo unirá al resto de máquinas. Por tanto, físicamente estará conectado a una red, pero lógicamente estará conectado a dos redes. Para conseguir esto le asigno dos direcciones IP de rangos distintos, creando dos red lógicas dentro la red virtual, una de ellas será la red 192.168.216.0/30, que une ROUTER con BOTFENCE y otra la red con dirección 10.0.0.0/24, que he llamado lanbots, donde se conectarán las máquinas del laboratorio. En la Ilustración 3 se muestra el resultado de forma visual.

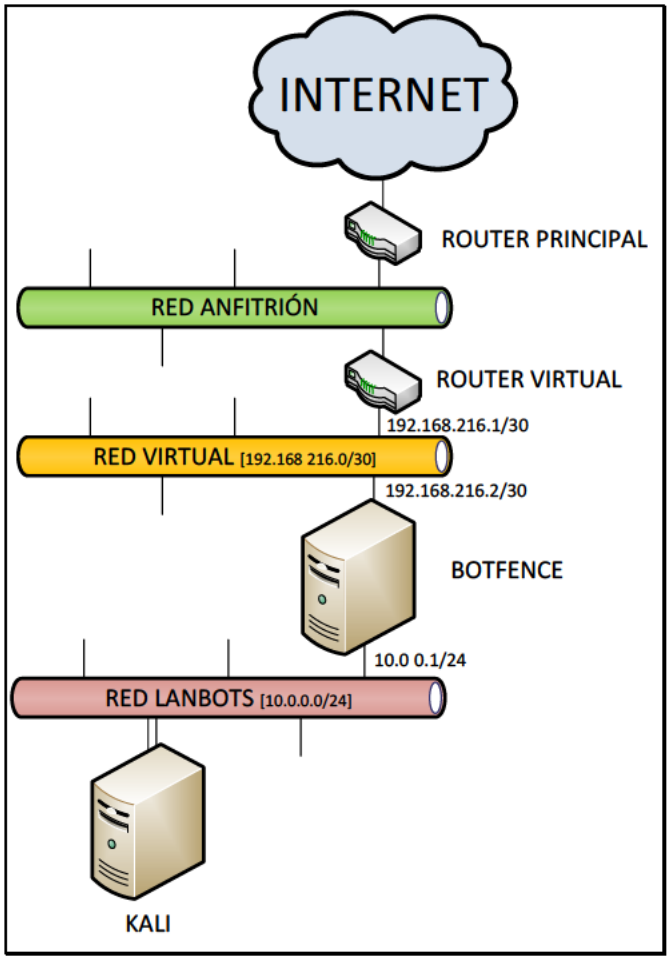


Ilustración 3. Diagrama lógico de red del laboratorio

A KALI le asigno 1 GB de RAM, un disco de 15 GB y una tarjeta conectada a la red virtual. Para instalar Kali Linux, una vez arrancado como LiveCD, tiene la posibilidad de instalarse de forma permanente en el disco. La instalación es automática sin intervención por parte del usuario.

Para ROUTER, como la instalación de OpenWrt requiere pocos recursos hardware, he creado una máquina con 32 MB de RAM, 50 MB de disco duro y dos tarjetas de red. Una de las tarjetas conectada a la tarjeta de red del host anfitrión, que será la eth1, y la otra conectada a una red interna, la eth0, que será la red virtual del diagrama.

El software de virtualización configura automáticamente la tarjeta de red conectada la red externa por DHCP, al seleccionarla como de tipo NAT. La tarjeta conectada a red virtual se conecta a una red lanbots con la IP 192.168.216.1. OpenWRT crea una interface de red llamada br-lan, por medio de las utilidades bridged lan, en la que agrupa a las LAN sobre las que tiene que hacer el enrutado, por lo que la IP 192.168.216.1 se asigna a esta interface, en lugar de a eth0. La interface eth1 se configura como cliente DHCP posteriormente desde BOTFENCE accediendo por web a ROUTER.

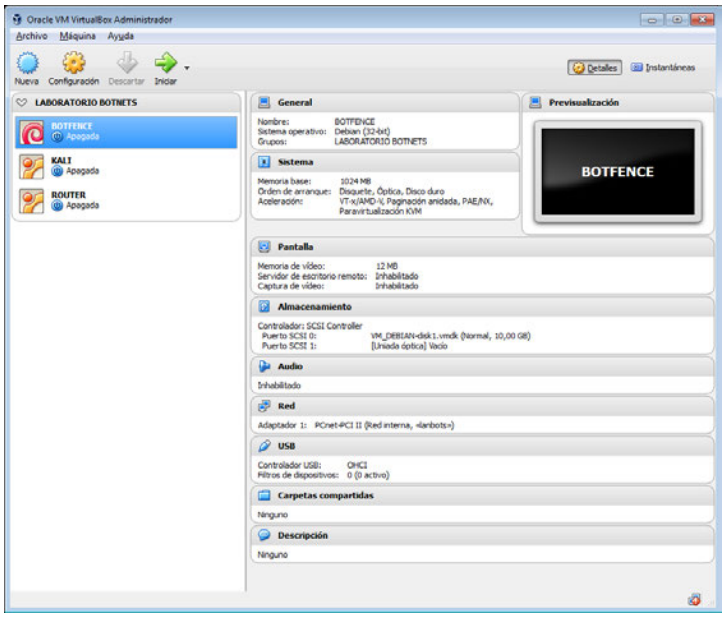


Ilustración 4. Laboratorio virtual

7. Captura y detección de botnets

7.1.Análisis y diseño

Existen varias técnicas para la detección de actividad de bots, que trabajan bien a nivel individual de cada máquina, controlando los procesos que se ejecutan, la memoria y los accesos al disco, bien a nivel de red local, analizando el tráfico que se produce en la red.

Utilizar técnicas de detección por medio del análisis de tráfico presenta algunas ventajas:

- El malware suele emplear técnicas de ocultación que complican la detección mediante un proceso en el mismo host.
- No se requiere la instalación del software de detección en los hosts.
- Los bots necesariamente produce tráfico de red para realizar sus actividades.

Para efectuar el análisis del tráfico de red hay que establecer los datos sobre los que se trabajará, que dependen directamente del detalle con el que se quiera analizar. Los tipos de datos que se pueden recoger clasificados según el tamaño de la captura son [30]:

- Datos de captura completa de paquetes.
- Datos de sesión.
- Datos estadísticos.
- Datos de paquetes de cadena.
- Datos de los log de las aplicaciones.
- Datos de alertas.

El análisis mediante la captura completa de paquetes es el nivel más bajo que permite mayor detalle en el análisis. Sin embargo, en una red grande, se puede volver inabordable y desbordar el almacenamiento por la gran cantidad de datos que se generan.

Por otro lado, los datos de alerta ocupan un espacio mínimo, ya que únicamente se generan ante eventos preestablecidos y permiten un análisis muy rápido a costa de perder parte de la información del paquete.

Los datos se capturan mediante sensores, que pueden ser [30]:

- Solo captura. Este tipo de sensor únicamente se encarga de recoger datos de la red, como paquetes o datos de sesión.
- De medio ciclo. En este caso, además de recoger datos, efectúa tareas de detección mediante un IDS.
- De ciclo completo. Este sensor es el más completo, ya que efectúa la captura, la detección y el análisis de los datos recogidos.

Los sensores de solo captura y de medio ciclo se utilizan para enviar los datos recogidos a otra máquina donde se efectuará el análisis, consiguiendo de esta forma liberar recursos no sobrecargando la máquina con procesos.

La tarea de análisis de los datos habitualmente se separa del sensor de captura por temas de rendimiento. Sin embargo, al tratarse en este caso de un entorno doméstico en el que se intenta que los recursos a emplear sean mínimos, la misma máquina deberá efectuar las tareas de captura, detección y análisis, además de una cuarta que será el bloqueo. Por tanto, estas tareas deberán estar optimizadas para que la red no se vea ralentizada y se produzca una pérdida de paquetes.

Para efectuar la detección se emplean los IDS. Son aplicaciones que están compuestas de dos módulos, un módulo que se encarga de la captura de los paquetes de la red y otro módulo que detecta si los paquetes capturados cumplen alguna regla de un conjunto de reglas preestablecidas. En caso de cumplir alguna regla, emite una alerta. El funcionamiento es similar a las firmas de los antivirus e incluso algunas de las reglas permiten la detección en base a una firma. Es importante que las reglas del IDS estén actualizadas para que detecten las nuevas amenazas y, por ello, se convierte es su debilidad. Una

botnet que no esté contemplada en las reglas se ejecutará en la red mientras el IDS no obtenga la regla que la bloquee.

Los IDS de libre distribución más consolidados en estos momentos son Snort y Suricata. Snort es un IDS que actualmente ha pasado a ser mantenido por una empresa, CISCO, que cobra una suscripción por la actualización de la reglas revisadas. Suricata surgió después de Snort, es compatible con sus reglas y está mantenido por la OISF (Open Information Security Foundation). Entre sus ventajas sobre Snort y por lo que finalmente lo he elegido como IDS, destaca:

- Es multihilo, por lo que aprovecha las arquitecturas de procesadores con varios núcleos.
- Efectúa la identificación de protocolos, permitiendo detectar anomalías basadas en protocolo en lugar de en el puerto esperado, como en el caso de HTTP.
- Permite identificar y extraer archivos, además de efectuar el cálculo de su MD5 para alertar de su paso.

BOTFENCE, como deberá hacer funciones de router, de sensor de ciclo completo y de bloqueador de botnets, ejecutará las siguientes tareas:

1. Recoger los paquetes que le llegan desde la red interna y desviarlos al IDS. Esta tarea se llevará a cabo configurando la máquina para que enmascare los paquetes de los paquetes de la red local y los encamine hacia una cola interna para que los pueda recoger el IDS.
2. El IDS deberá capturar los paquetes y, según las reglas, emitir alertas. Hay que instalar alguno de los IDS disponibles y configurar las reglas para que se emita una alerta en caso de que se detecte actividad de una botnet.
3. Analizar las alertas y bloquear las transmisiones si se detecta actividad de una botnet. Para ello, se procesarán en tiempo real las alertas generadas por el IDS y se bloquearán las transmisiones sospechosas. La interacción del usuario debe ser mínima, por lo que los bloqueos deberán ser

temporales para evitar falsos positivos, es decir, actividades lícitas que por su patrón de comportamiento, se detectan como una botnet.

4. Enviar los paquetes respuesta desde la WAN al host correspondiente. Se deberán encaminar los paquetes desde la WAN a la LAN para que el host origen reciba los datos que ha solicitado.

7.2.Implementación

7.2.1. Instalación de las máquinas del laboratorio

Para la implementación se requiere que el sistema operativo instale la menor cantidad de paquetes posibles, con el fin de limitar los recursos que empleará.

Por ello, para BOTFENCE al principio decido utilizar una distribución Linux específica para routers, sobre la que instalar posteriormente el IDS y desarrollar el analizador de alertas y bloqueador de transmisiones. En concreto, entre las disponibles, elijo OpenWRT por ser de libre distribución.

En el primer diseño, monto una máquina virtual con dos tarjetas de red, instalo OpenWRT 14.07 con una tarjeta de red con el servidor DHCP habilitado y IP 10.0.0.1 y la otra tarjeta de red hacia la máquina que ejercerá de router con IP 192.168.216.2.

Sin embargo, a lo hora de elegir el IDS, descubro que solo tiene disponible SNORT. Aunque la implementación es compatible con ambos IDS, también me doy cuenta que el ciclo de actualizaciones es menor que en otras distribuciones, como Debian.

Como el laboratorio virtual requiere de un router, reutilizo la instalación para que la máquina virtual haga las funciones de router en la red virtual, configurando una de las tarjetas de red en modo Bridge y la otra tarjeta de red con la IP fija 192.168.216.1. Deshabilito el servidor DHCP, ya que esta función la hará BOTFENCE.

El diseño con dos tarjetas de red para BOTFENCE, actuando intercalado entre la LAN y el router, presentaba desventajas, ya que no permitía interceptar las señales WiFi si el router ejercía de punto de acceso según la configuración de red que había diseñado y obligaba a adquirir un switch adicional si había más de un dispositivo cableado conectado al router. Finalmente decido cambiar el diseño, instalando una única tarjeta de red que funcionaría como dos tarjetas virtuales. Para el sistema operativo, desestimada una distribución específica de router, elijo una distribución de dominio público mantenida por la comunidad y con actualizaciones constantes para evitar agujeros de seguridad. Entre CentOS y Debian, escojo Debian, ya que CentOS deriva de Red Hat y está sujeta a sus actualizaciones, además de que Debian dispone de versión ARM y, por tanto, permite su instalación en computadores basado en este tipo de procesador, como la Rasberry PI, lo que debería facilitar la migración de la configuración establecida en la máquina virtual a la máquina real.

Instalo la última versión de Debian disponible, la 8.1, en una nueva máquina virtual con una única tarjeta de red. En esta tarjeta de red se crearán dos redes virtuales. Una que irá al router y otra que irá a la red privada, que hará las veces de servidor DHCP.

Para configurar las dos redes, hay que editar el archivo /etc/network/interfaces, con la primera interface en la misma red que el router con IP 192.168.216.2/30, y la segunda en la red interna con IP fija 10.0.0.1/24. Para ello, se le añaden las siguientes líneas al archivo /etc/network/interfaces [31]:

```
allow-hotplug eth0
iface eth0 inet static
 address 192.168.216.2
 netmask 255.255.255.252
 gateway 192.168.216.1

auto eth0
iface eth0 inet static
 address 10.0.0.1
```

```
 netmask 255.255.255.0
```

Como Botfonce debe hacer funciones de servidor DHCP, hay que instalar el paquete dhcp3-server. Para configurarlo, hay que editar el archivo /etc/default/isc-dhcp-server con la siguiente línea que indica a que interfaces se conectará el servidor DHCP [32]:

```
INTERFACES="eth0"
```

Como la tarjeta eth0 se configura con dos direcciones IP, hay que establecer una configuración de red compartida en la configuración del servidor DHCP para que funcione correctamente, ya que las peticiones DHCP son broadcast, por lo tanto, irán a la misma interface y se obliga a definir necesariamente todas las redes que tiene la interface a la que está enganchada el servidor DHCP. Para que únicamente entregue direcciones en la red 10.0.0.0, hay que editar el archivo /etc/dhcp/dhcp.conf, añadiendo las siguientes líneas:

```
option domain-name-servers 8.8.8.8;
shared-network botnet {
 subnet 10.0.0.0 netmask 255.255.255.0 {
  range 10.0.0.2 10.0.0.254;
  option subnet-mask 255.255.255.0;
  option routers 10.0.0.1;
 }
 subnet 192.168.216.0 netmask 255.255.255.252 {
 }
}
```

Con esta configuración, se le indica al servidor DHCP que se dispone de dos redes con las direcciones 192.168.216.0/30 y 10.0.0.0/24. Deberá entregar únicamente direcciones en el rango que va desde 10.0.0.2 a 10.0.0.254, con máscara 255.255.255.0, con la pasarela 10.0.0.1, que es BOTFENCE y el DNS 8.8.8.8, que es uno de los DNS de Google.

BOTFENCE hace las funciones de router, por lo que hay que configurar los parámetros adecuados en el sistema operativo para que se permita el encaminamiento y enmascaramiento de los paquetes de la red local hacia el router original.

Para que se puedan encaminar los paquetes, se establecer los siguientes parámetros en el archivo /etc/sysctl.conf:

```
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
```

Para habilitar los cambios, se lanza la siguiente orden que permite modificar los parámetros del kernel en tiempo de ejecución:

```
sysctl –p /etc/sysctl.conf
```

Para el encaminamiento y enmascaramiento de paquetes en Linux, se utiliza 'iptables'. Como la máquina debe ejercer de router para la red la red 10.0.0.0 en eth0, creo que archivo /etc/init.d/router con el siguiente contenido [33]:

```
#!/bin/bash
### BEGIN INIT INFO
# Provides: router
# Required-Start: $syslog
# Required-Stop: $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: router_debian
# Description:
### END INIT INFO
##FLUSH DE REGLAS
iptables –Z
iptables -t nat -F
# Enmascaramiento de la red 10.0.0.0 para que tenga salida a
través de la tarjeta de red eth0.
iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -o eth0 -j
```

```
MASQUERADE
```

Este archivo que se ejecuta cuando se inicia el sistema operativo e indica que se borren las entradas de iptables y se cree una regla que enmascare los paquetes de la red 10.0.0.0/24 después de encaminarlos, cambiándoles la cabecera para que el origen sea BOTFENCE y los envíe por la tarjeta de red eth0. Para que la configuración se ejecute cada vez que se enciende la máquina, se lanza la siguiente orden:

```
etc/init.d/update-rc.d router defaults
```

Para comprobar que todo funciona correctamente, hay que instalar una tercera máquina en la red interna que funcione por DHCP. Creo una nueva máquina virtual e instalo la distribución Kali Linux. Esta distribución, diseñada para realizar auditorías de seguridad, me permitirá simular ataques desde la red [34]. Es una distribución Live, pero decido instalarla en disco para poder instalar posteriormente aplicaciones adicionales y que no se elimine la configuración con cada reinicio.

7.2.2. Prueba del servidor DHCP y la salida a Internet

Al encender la máquina Kali, obtiene la IP 10.0.0.2, con la máscara 255.255.255.0, la pasarela 10.0.0.1 y el DNS 8.8.8.8, por lo que el servidor DHCP está funcionando correctamente.

Para probar la salida a Internet y que los paquetes viajan a través de los routers BOTFENCE y ROUTER, ejecuto los siguientes comandos:

```
ping www.google.es
traceroute www.google.es
```

Obtengo respuesta en ambos. En el caso del trazado de ruta, me confirma que pasa por 10.0.0.1 y 192.168.216.1, que son respectivamente BOTFENCE y ROUTER.

Para tener un laboratorio de pruebas más completo, decido instalar una cuarta máquina con Microsoft Windows XP SP2, que es un sistema operativo que no dispone de los últimos parches, por lo que se considera vulnerable.

Al finalizar la instalación, ejecuto las ordenes anteriores, y obtengo la misma salida que en el caso anterior, que es correcta, tal como se muestra en la Ilustración 5.

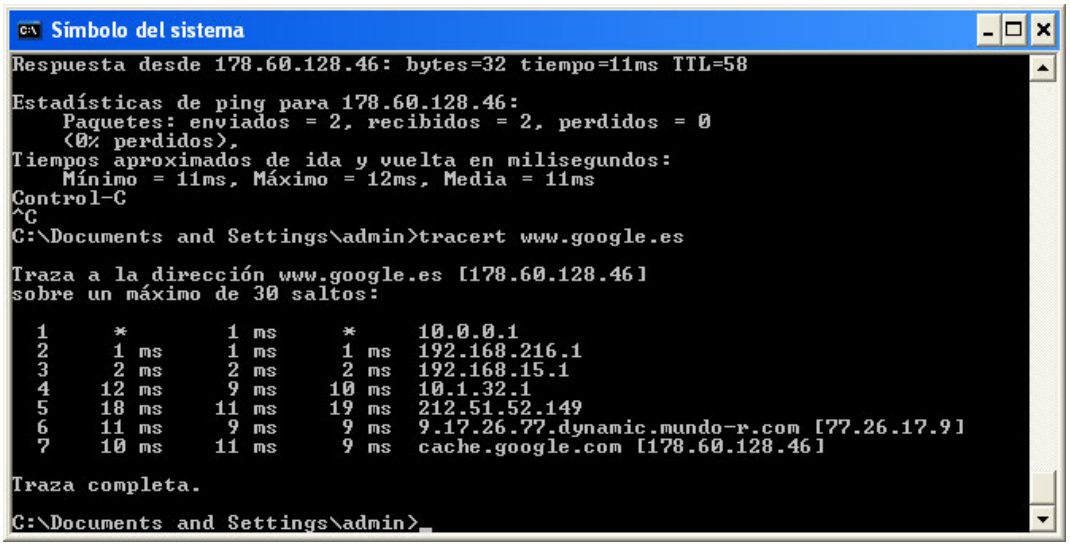


Ilustración 5. Prueba de conectividad

7.2.3. Instalación del IDS

El IDS Suricata no viene por defecto con la instalación, por lo que hay que instalar el paquete y sus dependencias con la siguiente orden:

```
apt-get install suricata
```

Para su configuración, en el caso de la distribución Debian, hay que editar el archivo /etc/suricata/suricata-debian.yaml.

El primer paso es configurar los rangos por defecto de los sistemas operativos. Se configura para que el análisis de los paquetes por defecto los considere en primera instancia del sistema operativo Microsoft Windows, ya que según las estadísticas en Julio de 2015 [35], está instalado en aproximadamente el 85% de los ordenadores de escritorio. Aunque es la política por defecto, este parámetro no impide que se analicen de forma correcta el resto de sistema operativos. En la Ilustración 6 se muestra el resultado de la edición, estableciendo también que la IP 10.0.0.1, que es BOTFENCE, tiene el sistema operativo Linux.

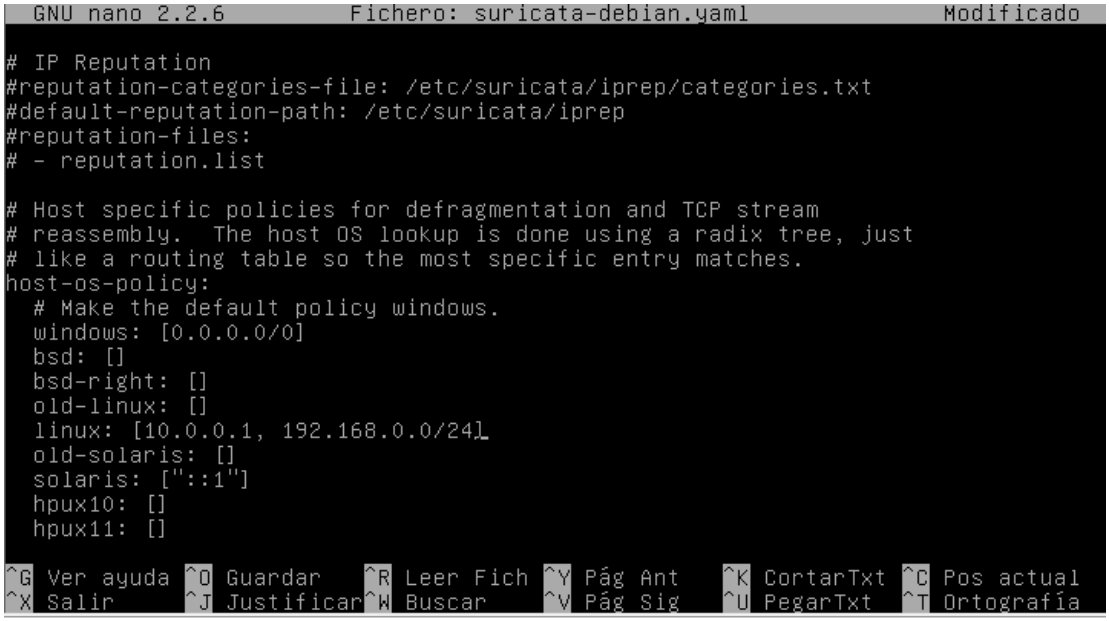


Ilustración 6. Política de defragmentación por defecto

La detección de Suricata se basa en firmas, por lo que debe ser actualizado constantemente para la detección de nuevas amenazas. Para su actualización, se debe instalar el paquete oinkmaster con la siguiente orden:

```
apt-get install oinkmaster
```

Este paquete se encarga de descargar las últimas reglas publicadas y copiarlas en la carpeta de reglas de Suricata. Una vez finalizada su instalación, hay que editar el archivo/etc/oinkmaster.conf, indicándole la URL donde obtener las reglas con el siguiente parámetro: 'url:

http://rules.emergingthreats.net/open/suricata/emerging.rules.tar.gz' e indicar que no modifique el archivo 'botfence.rules' con 'skipfile botfence.rules', ya que es el archivo que se utilizará para las reglas de bloqueo de ataques de botnets.

Finalmente, para descargar las reglas, se ejecuta la siguiente orden:

```
oinkmaster –o /etc/suricata/rules
```

En la carpeta indicada en el parámetro –o se descomprimirán todas las reglas descargadas, además de los archivos classification.config y reference.config. Como estos dos últimos archivos están en el instalación por defecto de Suricata en la carpeta /etc/suricata, hay que establecer en el archivo /etc/suricata-debian.yaml la nueva localización.

7.2.4. Pruebas del IDS

Para probar que el IDS, se pone en funcionamiento con la siguiente orden en BOTFENCE, donde se indica la localización del archivo de configuración y la interface sobre la que se debe ejecutar la captura y detección:

```
suricata –c /etc/suricata/suricata-debian.yaml –i eth0
```

Desde la máquina con Windows, se hace ping y se navega a www.google.es. Se comprueba que en el archivo /var/log/suricata/eve.json se han registrado eventos de DNS y HTTP hacia máquinas de Google.

Para comprobar si funcionan las alarmas, se miran dominios que estén en las reglas, en concreto, veo que están todos los dominios terminados en .co.cc. Buscando por Internet dominios con ese dominio superior, obtengo [http://plandicardyu9.co.cc/index.php?k=Spun], por lo que navego a esa página y se visualiza correctamente en el navegador. Se registra la alarma en /var/log/suricata/unified2.alert, aunque se accede a la página. Esperaba que la página se bloquease, pero leyendo el manual veo que el comportamiento es correcto, ya que Suricata está funcionando en modo IDS y no es modo IPS. En el

modo IDS, Suricata visualiza los paquetes y alerta sobre anomalías. En el modo IPS, funciona como un router y tiene la capacidad de bloquearlos. Hay que tener en cuenta que la regla también tiene que estar habilitada para que los bloquee. Para configurar el modo IPS, hay que habilitar NFQ en IPTABLES con la siguiente orden.

```
iptables -I FORWARD -j NFQUEUE
```

Este modo de funcionamiento hará que los paquetes que se reenvían (FORWARD) en BOTFENCE, se manden a una cola, que por defecto es la cola número 0, desde donde los recogerá Suricata.

Para que Suricata capture los paquetes desde la cola, se lanza la siguiente orden:

```
suricata -c /etc/suricata/suricata-debian.yaml -q 0
```

Se realizan las mismas pruebas de antes, ping y navegación a www.google.es y funciona correctamente. Se accede a http://plandicardyu9.co.cc/index.php?k=Spun, y muestra la página. Es debido a que la regla está configurada para que alerte del acceso. Para que descarte paquetes, hay que establecer la regla como de tipo DROP.

Para comprobar que funciona correctamente, se modifica el archivo /etc/suricata/rules/drop.rules para que descarte los paquetes ICMP (ping) añadiendo la regla:

```
drop icmp $HOME_NET any -> $EXTERNAL_NET any (msg: "PING detected"; sid:2;rev1;)
```

Se pone de nuevo en ejecución Suricata y se hace ping a www.google.es desde la máquina Windows. En este caso, no se obtiene respuesta, da timeout y se registran los eventos en /var/log/suricata/eve.json y /var/log/suricata/drop.log, lo que indica que Suricata está funcionando correctamente en modo IPS.

8. Análisis y bloqueo de botnets

8.1. Análisis y diseño

Entre las funcionalidades que puede proporcionar una botnet, la que más afecta a agentes externos son los ataques DDoS. Los ataques producidos por botnets están basados en volumen de datos[4] y se pueden clasificar en los siguientes tipos [36]:

- **Ataques por volumen de tráfico a servidores web:** se basan en lanzar gran cantidad de peticiones a un servidor web que provocan un consumo elevado de recursos de procesamiento en la víctima, imposibilitando la atención de las peticiones legítimas con normalidad.
- **Ataques por enlentecimiento HTTP a servidores web:** se realizan peticiones HTTP al servidor que se mantienen abiertas, bloqueando los servicios de procesamiento de peticiones al dejarlos atendiendo peticiones pendientes de cierre.
- **TCP SYN:** se lanza un gran volumen de paquetes TCP SYN a la víctima utilizando direcciones IP con el origen falso.
- **TCP Full Connect:** es un ataque que conserva la IP de origen y que realiza todo el proceso completo del TCP hand-shake, provocando el consumo de recursos de la víctima por la carga de procesamiento del elevado número de peticiones.
- **TCP ACK/FIN:** los atacantes envían un gran volumen de paquetes TCP ACK/FIN a la víctima que provocan el consumo de su ancho de banda.
- **TCP RST:** se envían paquetes TCP RST que la víctima tiene que recibir, chequear y descartar, elevando el consumo de recursos y llevando al bloqueo cuando se incrementa el número de peticiones.

[4] Los ataques de botnets se basan en volumen porque aprovechan la cantidad de bots para producir de forma coordinada un gran número de paquetes que bloqueen a la víctima. El otro tipo está basado en exploits, que aprovecha vulnerabilidades de los sistemas y los protocolos.

- **Inundación DNS:** este ataque está dirigido a servidores DNS. Los clientes de la botnet bloquean el servidor enviando gran cantidad de paquetes de petición de resolución de nombres de dominio.
- **Inundación UDP:** se envían gran cantidad de paquetes UDP a un puerto aleatorio de la víctima, forzando la devolución de paquetes ICMP con 'Destino Inalcanzable' y provocando el bloqueo por consumo de recursos.
- **Inundación ICMP:** los bots envían paquetes del tipo ICMP Echo Request y la víctima responde con paquetes ICMP Reply, inundando la red de paquetes.
- **Non TCP/UDP/ICMP Flood:** se envían paquetes que de otro tipo distinto a TCP, UDP o ICMP, como por ejemplo paquetes IP malformados.
- **Inundación a nivel de aplicación:** los atacantes aprovechan vulnerabilidades de aplicación o peticiones que consuman un elevado nivel de recursos para agotar los recursos de la víctima.

Todos los ataques descritos tienen un elemento en común: provocan un tráfico de red elevado de un tipo en concreto. En cada caso, el tráfico de red será distinto, por lo que habrá que individualizarlos para que se puedan detectar todos ellos.

El último caso, que es un ataque a nivel de aplicación, debería ser el software de la víctima quién detectase y evitase estas situaciones, bien corrigiendo la vulnerabilidad o bien descartando las peticiones que provocan un elevado consumo de recursos.

Para simular cada uno de los ataques, voy a utilizar la aplicación de libre distribución 'bonesi' [37], desarrollada por Markus Goldstein. Permite la simulación de cada uno de los ataques por volumen provocados por botnets. Utilizaré la aplicación de captura de paquetes Wireshark para visualizar el tráfico que se genera con el fin de crear los patrones de comportamiento que permitan discernir el tipo de bloqueo que hay que establecer para detener el ataque. Los ataques se simularán asociados al protocolo utilizado. El IPS deberá detectar los

ataques mediante la generación de reglas específicas que indiquen que se está produciendo un ataque desde una máquina interna.

El problema de los bloqueos son los falsos positivos. Para evitar que se provoque un bloqueo de una comunicación legítima, el software de análisis y bloqueo deberá crear bloqueos temporales, pero incrementando el tiempo de bloqueo forma lineal si se repite la situación. Con esto, se consigue que el sistema sea autónomo, lo que evitará la intervención del usuario en los falsos positivos y cuando se elimine de la red el malware.

8.2.Implementación

La implementación consta de dos partes diferenciadas, por un lado las reglas para que el IPS detecte los ataques, y por otro, el software de análisis y gestión de los bloqueos, al que llamaré Botfence.

Cada ataque genera en la red un patrón distinto, por lo que las reglas normalmente serán específicas para cada uno de ellos. Para que se puedan identificar las regla que controlan los ataques de botnets, el mensaje de descripción comenzará por la palabra “BOTNET”. La simulación del ataque se llevará a cabo mediante el software ‘bonesi’ que se invoca con la siguiente orden:

```
bonesi [opciones] <IP_destino:puerto>
```

Se utilizarán las siguiente opciones:

-p: protocolo

-d: interface de red sobre la que se lanzan los paquetes

--ips: archivo con listado de direcciones IP. Se utilizará para simular con ataques con IP spoofing pasándole un archivo con 50000 direcciones IP llamado ‘50k-bots’ que viene incluido en el software.

8.2.1. Detección de ataque ICMP Flood con IP interna

Este ataque envía paquetes ICMP a un puerto de la víctima. La IP de origen es la IP del ordenador atacante. Para simularlo, se ejecuta la siguiente orden:

```
bonesi –p icmp 192.168.1.1:80
```

Se genera un número inusual de paquetes de tipo ICMP en la red, por lo que se puede detectar estableciendo un número máximo de paquetes ICMP cada cierto tiempo, con independencia del puerto destino. Si se sobrepasa ese umbral, se bloquean los paquetes. La regla para detectarlo con un umbral de 100 paquetes ICMP en 20 segundos a un mismo destino es:

```
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:”BOTNET: PING threshold limit”; flow:to_server, threshold: type threshold, track by_dst, count 100, seconds 20; sid:50001; rev:1;)
```

8.2.2. Detección de ataque ICMP con IP spoofing

Este ataque envía paquetes ICMP a un puerto de la víctima, pero la IP de origen es falsa. Para simularlo, se ejecuta la siguiente orden:

```
bonesi –p icmp –ips 50k-bots 192.168.1.1:80
```

La detección en este caso se hace comprobando que no pasa ningún paquetes ICMP con IP origen externa e IP destino externa, independientemente del puerto destino, con la siguiente regla:

```
alert icmp $EXTERNAL_NET any -> $EXTERNAL_NET any (msg:”BOTNET: ICMP IP spoofing”;sid:50002;rev:1;)
```

8.2.3. Detección de ataque UDP Flood con IP interna

Este ataque envía paquetes UDP a un puerto de la víctima. La IP de origen es la IP del ordenador atacante. Para simularlo, se ejecuta la siguiente orden:

```
bonesi -p udp 192.168.1.1:80
```

Se genera un número inusual de paquetes de tipo UDP en la red, por lo que se puede detectar estableciendo un número máximo de paquetes UDP cada cierto tiempo, con independencia del puerto destino. Si se sobrepasa ese umbral, se bloquean los paquetes. La regla para detectarlo con un umbral de 500 paquetes en 10 segundos a un mismo destino es:

```
alert udp $HOME_NET any -> $EXTERNAL_NET any (msg:" BOTNET: UDP threshold limit"; flow:to_server; threshold: type threshold, track by_dst, count 500, seconds 10; sid:50003; rev:1;)
```

8.2.4. Detección de ataque UDP con IP spoofing

En este caso, se inunda la red con paquetes UDP falseando la IP de origen de los paquetes. Se simula con la siguiente orden:

```
bonesi -p udp  -ips 50k-bots 192.168.1.1:80
```

Para detectarlo, se comprueba que no existan paquetes UDP con origen IP externa y destino IP externa con la siguiente regla:

```
alert udp $EXTERNAL_NET any -> $EXTERNAL_NET any (msg:" BOTNET: UDP IP spoofing";sid:50004;rev:1;)
```

8.2.5. Detección de ataque web con IP interna

Los ataques a servidores web se realizan mediante paquetes TCP que transportan la solicitud HTTP. Se pueden simular mediante la siguiente orden:

```
bonesi -p tcp -d eth0  192.168.1.1:80
```

Las comunicaciones legítimas con servicios web generan un número elevado de paquetes, por lo que el umbral de detección se debe elevar. He fijado el umbral

en 500 paquetes en 10 segundos al mismo destino. Con la siguiente regla, como no se han fijado los puertos destino, se detectarán todos los ataques TCP:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:" BOTNET: TCP
threshold limit"; flow:to_server; threshold: type threshold,
track by_dst, count 500, seconds 10; sid:50005; rev:1;)
```

Se podría limitar a la detección de ataques HTTP estableciendo los puertos con la variable $HTTP_PORTS que viene ya definida en los archivos de configuración de Suricata.

8.2.6. Detección de ataque web con IP spoofing

En este caso, se inunda la red con paquetes TCP falseando la IP de origen de los paquetes. Se simula con la siguiente orden:

```
bonesi -p tcp -d eth0  -ips 50k-bots 192.168.1.1:80
```

Para detectarlo, se comprueba que no existan paquetes TCP con origen IP externa y destino IP externa. Con la siguiente regla, como no se han fijado los puertos destino, se detectarán todos los ataques TCP:

```
alert udp $EXTERNAL_NET any -> $EXTERNAL_NET any (msg:" BOTNET:
TCP IP spoofing";sid:50006;rev:1;)
```

8.2.7. Detección de ataque DNS Flood

Los paquetes de petición de resolución de nombre son de tipo UDP. Al lanzar un ataque DNS Flood, se lleva a cabo una generación de paquetes UDP similar a la que se produce con el ataque UDP Flood, por lo que se puede considerar que es el mismo ataque y se bloquean con las reglas establecidas para los ataques UDP Flood.

8.2.8. Detección de comunicación con C&C

La detección de las comunicaciones con C&C se puede hacer con las reglas de Suricata que se descargan. Es importante que estas reglas estén actualizadas

para que se detecten las amenazas más recientes, ya que su forma de detección se basa en la IP del C&C.

Por defecto, estas reglas se descargan para que generen una alerta en el registro de eventos. Sin embargo, ya que el equipo de detección y bloqueo va a actuar de forma autónoma, he modificado el archivo /etc/oinkmaster.conf para que cambie las alertas de detección de paquetes con destino un C&C por el bloqueo de paquetes mediante las siguientes líneas:

```
modifysid botcc.rules "alert" | "drop"
modifysid botcc.portgrouped.rules "alert" | "drop"
modifysid emerging-trojan.rules "alert" | "drop"
```

Esta configuración establecerá las reglas de tipo 'alert' a reglas de tipo 'drop', que descartan los paquetes cuando se cumplen. Se ejecutan sobre los archivos botcc.rules y botcc.portgrouped.rules porque ambos contienen las reglas de detección de servidores C&C conocidos.

8.2.9. Detección de envío masivo de correos electrónico

El envío de correo masivo a un servidor SMTP se puede llevar a cabo de dos formas:

1. Mediante el envío de mensajes independientes, especificando en cada envío el origen y el destino. La conversación SMTP de envío de dos mensajes independientes tiene el siguiente formato:

```
220 mail.victima.org ESMTP server ready Tue, 20 Jan 2004
22:33:36 +0200
HELO malware.botnet.com
250 mail.victima.org
MAIL FROM: <conocido@victima.org>
250 Sender <conocido@victima.org> Ok
RCPT TO: <incauto1@victima.org>
250 Recipient <incauto1@victima.org> Ok
DATA
```

```
354 Ok Send data ending with <CRLF>.<CRLF>
FROM: conocido@victima.org
TO: incauto1@victima.org
Subject: Prueba de mensaje

Esto es una prueba.
.
250 Message
received:154.4382.mail.victima.org@malware.botnet.com
MAIL FROM: <conocido@victima.org>
250 Sender <conocido@victima.org> Ok
RCPT TO: <incauto2@victima.org>
250 Recipient <incauto2@victima.org> Ok
DATA
354 Ok Send data ending with <CRLF>.<CRLF>
FROM: conocido@victima.org
TO: incauto2@victima.org
Subject: Prueba de mensaje

Esto es una prueba.
.
250 Message
received:155.4382.mail.victima.org@malware.botnet.com
QUIT
221 mail.victima.org ESMTP server closing connection
```

2. Enviando a varios destinatarios el mismo mensaje conservando el origen del mensaje. En este caso, la conversación SMTP para enviar a dos destinarios tiene el siguiente formato:

```
220 mail.victima.org ESMTP server ready Tue, 20 Jan 2004
22:33:36 +0200
HELO malware.botnet.com
250 mail.victima.org
MAIL FROM: <conocido@victima.org>
250 Sender <conocido@victima.org> Ok
RCPT TO: <incauto1@victima.org>
250 Recipient <incauto1@victima.org> Ok
```

```
RCPT TO: <incauto2@victima.org>
250 Recipient <incauto2@victima.org> Ok
DATA
354 Ok Send data ending with <CRLF>.<CRLF>
FROM: conocido@victima.org
TO: incauto1@victima.org; incauto2@victima.org
Subject: Prueba de mensaje

Esto es una prueba.
.
250 Message
received:154.4382.mail.victima.org@malware.botnet.com
QUIT
221 mail.victima.org ESMTP server closing connection
```

Ambas situaciones se pueden detectar con la siguiente regla, que salta si se envían más de 25 mensajes en 60 segundos:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 25 (msg:"BOTNET:
SMTP threshold limit"; flow:to_server,no_stream;
content:"rcpt to|3a|"; nocase; threshold: type threshold,
track by_src, count 25, seconds 60; sid:50007; rev:1;)
```

8.2.10. Análisis y bloqueo de los ataques

El software encargado del análisis y bloqueo se debe ejecutar rápido y empleando pocos recursos, ya que los ataques generan gran cantidad de tráfico en muy poco tiempo y la máquina sobre la que se realizará la implementación posiblemente no disponga de mucha potencia de proceso. Con el fin de optimizar lo máximo posible el código, he decidido hacer la implementación utilizando el lenguaje C.

La aplicación lee de forma continua los eventos del archivo de registro de Suricata que están en formato JSON. Si el evento se ha generado después de la hora de ejecución de la aplicación y tiene la palabra BOTNET, lo analiza. En caso de que sea la primera vez que se produce, bloquea los paquetes el tiempo de

bloqueo establecido en los parámetros de configuración. Si el mismo evento se vuelve a repetir de nuevo no habiendo transcurrido el tiempo entre repeticiones de eventos, que también se establece en los parámetros de configuración, se bloquean de nuevo los paquetes, pero en esta ocasión el tiempo de bloqueo se multiplicará por el número de veces que se ha bloqueado anteriormente el paquete, consiguiendo un tiempo de bloqueo directamente relacionado con la duración del ataque. Para llevarlo a cabo, la aplicación mantiene una lista de eventos activos y eventos que están dentro del tiempo mínimo establecido entre repeticiones. Si un evento no se vuelve a repetir transcurrido ese tiempo, se elimina de la lista. La estructura de los elementos se muestra en el siguiente listado:

```
struct net_event {
      time_t timestamp; /* hora del evento */
      char event_type[20]; /* tipo de evento */
      char src_ip[16]; /* dirección origen */
      unsigned int src_port; /* puerto origen */
      char dest_ip[16]; /* dirección destino */
      unsigned int dest_port; /* puerto destino */
      char proto[10]; /* protocolo */
      time_t fence_time; /* hora de bloqueo */
      char signature[2048]; /* firma que identifica el paquete */
      unsigned int count; /* num. veces que se ha repetido */
      int status; /* 1-> bloqueado. 0-> desactivado */
      struct net_event *next; /* puntero al siguiente elemento */
};
```

Para realizar el bloqueo, se utiliza ‘iptables’. Los paquetes se bloquearán antes de entrar en la cola, para evitar trafico innecesario y que Suricata los tenga que procesar de nuevo.

El bloqueo de los ataques ICMP se lleva a cabo con la siguiente orden, que indica que se bloquearán todos los paquetes ICMP que se dirijan a la IP destino indicada:

```
iptables -I FORWARD 1 -d <IP_destino> -p ICMP -j DROP
```

Mientras que para bloquear el resto de ataques, se utiliza la siguiente orden, en la que se establece la IP destino, el protocolo y el puerto destino:

```
iptables -I FORWARD 1 -d <IP_destino> -p <protocolo> --
destination-port <puerto_destino> -j DROP
```

Transcurrido el tiempo de bloqueo establecido, se lanzará una nueva orden con el siguiente formato en caso de ser un ataque ICMP:

```
iptables -D FORWARD -d <IP_destino> -p ICMP -j DROP
```

Y con este formato en los otros casos:

```
iptables -D FORWARD -d <IP_destino> -p <protocolo> --destination-
port <puerto_destino> -j DROP
```

Una vez desbloqueado el evento, deberá transcurrir el tiempo establecido de no repetición en los parámetros de configuración de botfence para que se elimine definitivamente de la lista de eventos. Si se repite el ataque estando en la lista, se multiplicará el tiempo de bloqueo por el número de veces que se ha repetido.

Al iniciarse la aplicación, realizará una limpieza de las tablas de 'iptables', introducirá la redirección a la cola para que los mensajes pasen a través de Suricata y funcione como IPS y, por último, ejecutará Suricata en background.

9. Pruebas

Para realizar las pruebas se simulan los ataques mediante la aplicación 'bonesi', ejecutándola con los parámetros adecuados según el tipo de ataque. Las pruebas se hacen con todas las máquinas del laboratorio y utilizando la máquina ROUTER como víctima para no enlentecer la red externa.

9.1.Patrones simulados

9.1.1. ICMP Flood

Se comienza la prueba lanzando en BOTFENCE la aplicación Botfence con el IPS Suricata. Se comprueba que en 'iptables' únicamente figura el encaminamiento de los paquetes FORWARD a la cola número 0, desde donde los cogerá Suricata. Se pone en una terminal el log de eventos de Suricata con 'tail –f /var/log/suricata/eve.json'.

Para simular los ataques ICMP Flood, se ejecuta la siguiente orden desde una terminal de KALI:

```
bonesi -p icmp 192.168.1.1:80
```

Inmediatamente se observa como en el log de Suricata que hay en BOTFENCE se registran eventos de ataque ICMP. Botfence analiza los logs, lanza el bloqueo y el log se detiene. En la Ilustración 7, numerando las terminales de izquierda a derecha y de arriba a abajo, se muestra la ejecución de Suricata en la terminal 1, el registro del ataque ICMP Flood detectado por Suricata en la terminal 2, la ejecución y bloqueo del ataque por Botfence en la terminal 3 y el estado de las 'iptables' antes y después del ataque, con el bloqueo de los ataques ICMP desde cualquier origen a 192.168.1.1.

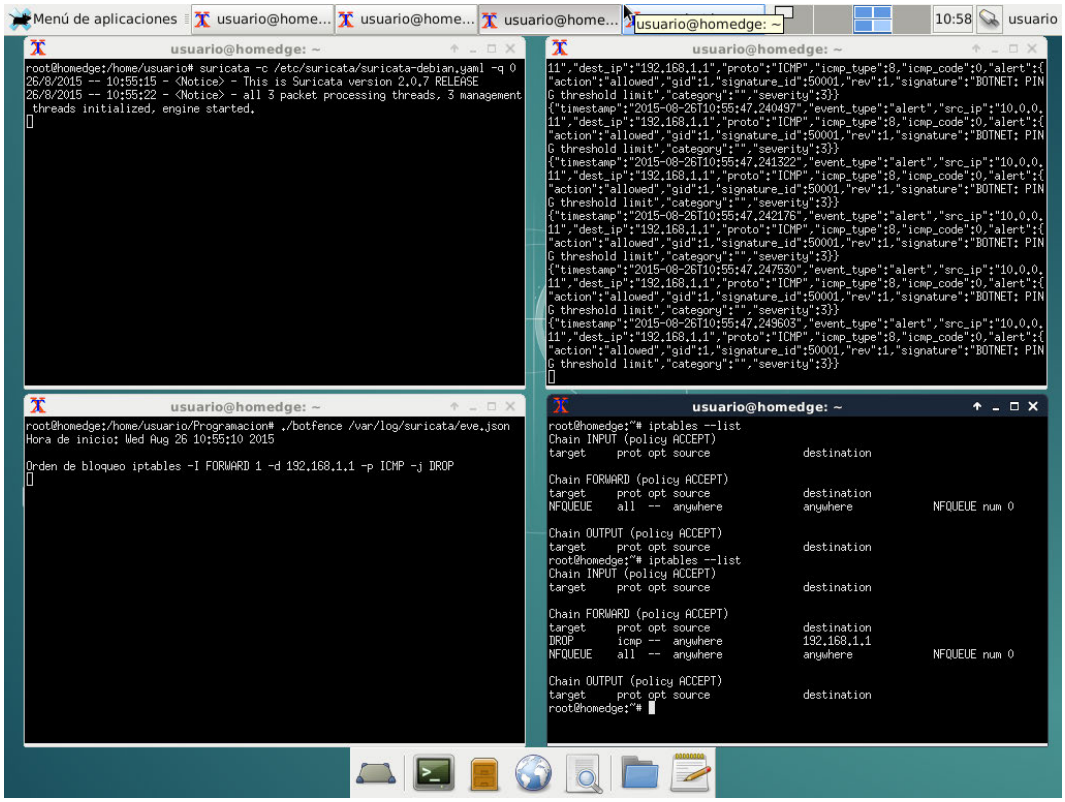


Ilustración 7. Prueba de ICMP Flood

Se repite la prueba con IP Spoofing y, en esta ocasión, no se registran eventos por Suricata. Esto es debido a que Botfence bloquea los ataques ICMP para todos los orígenes, por lo que se detiene el ataque antes de llegar a Suricata al haber una regla por el ataque anterior.

9.1.2. UDP Flood

Se continúan las pruebas partiendo del entorno de la prueba anterior, por lo que en BOTFENCE ya se está ejecutando Botfence y Suricata.

Para simular los ataques UDP Flood, se lanza la siguiente orden desde una terminal de KALI:

```
bonesi -p UDP 192.168.1.1:80
```

En el log de Suricata se comienzan a registrar los eventos del ataque UDP. En el momento en que Botfence analiza los logs, lanza el bloqueo y el log se detiene.

En la Ilustración 8, numerando las terminales de izquierda a derecha y de arriba a abajo, se muestra la ejecución de Suricata en la terminal 1, el registro del ataque UDP Flood detectado por Suricata en la terminal 2, la ejecución y bloqueo del ataque por Botfence en la terminal 3 y el estado de las 'iptables' antes y después del ataque, con el bloqueo de los paquetes UDP enviados al host 192.168.1.1. En la terminal 3, además, se muestra como se desbloquea el envío de paquetes ICMP, que se había hecho en la prueba anterior, al haber transcurrido el tiempo de bloqueo establecido en los parámetros de configuración de Botfence.

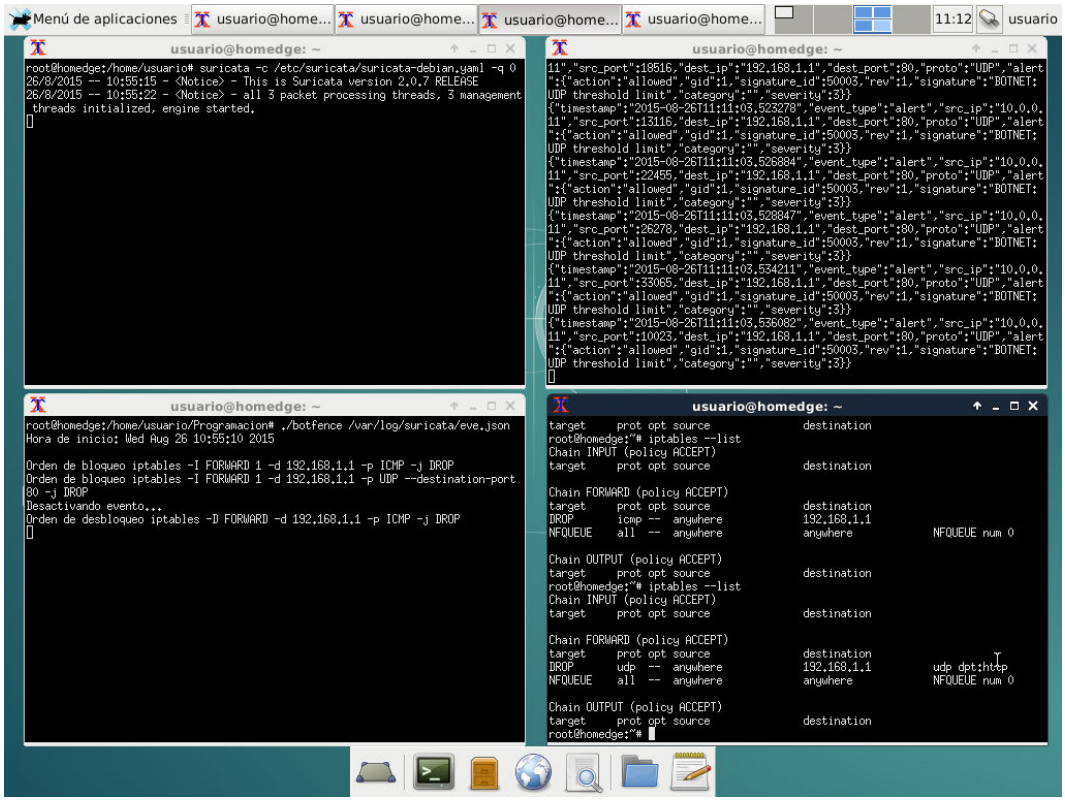


Ilustración 8. Prueba de UDP Flood

Se repite de nuevo la prueba pero con IP Spoofing. Suricata no registra ningún evento porque la regla de bloqueo que ha introducido Botfence está establecida por equipo destino, por lo que sirve igualmente para bloquear ataques que hagan IP Spoofing a la misma víctima.

9.1.3. HTTP Flood

Se comienzan las pruebas arrancando Botfence y Suricata en BOTFENCE y se visualiza en un terminal el log de Suricata y en otro el contenido de las 'iptables'.

Para simular los ataques TCP o HTTP Flood, se lanza la siguiente orden desde una terminal de KALI:

```
bonesi -p TCP -d eth0 192.168.1.1:80
```

En el log de Suricata se comienzan a registrar los eventos del ataque TCP. En el momento en que Botfence analiza los logs, lanza el bloqueo y el log se detiene. En la Ilustración 9, numerando las terminales de izquierda a derecha y de arriba a abajo, se muestra la ejecución de Suricata en la terminal 1, el registro del ataque UDP Flood detectado por Suricata en la terminal 2, la ejecución y bloqueo del ataque por Botfence en la terminal 3 y el estado de las 'iptables' antes y después del ataque, con el bloqueo de los paquetes TCP enviados al host 192.168.1.1.

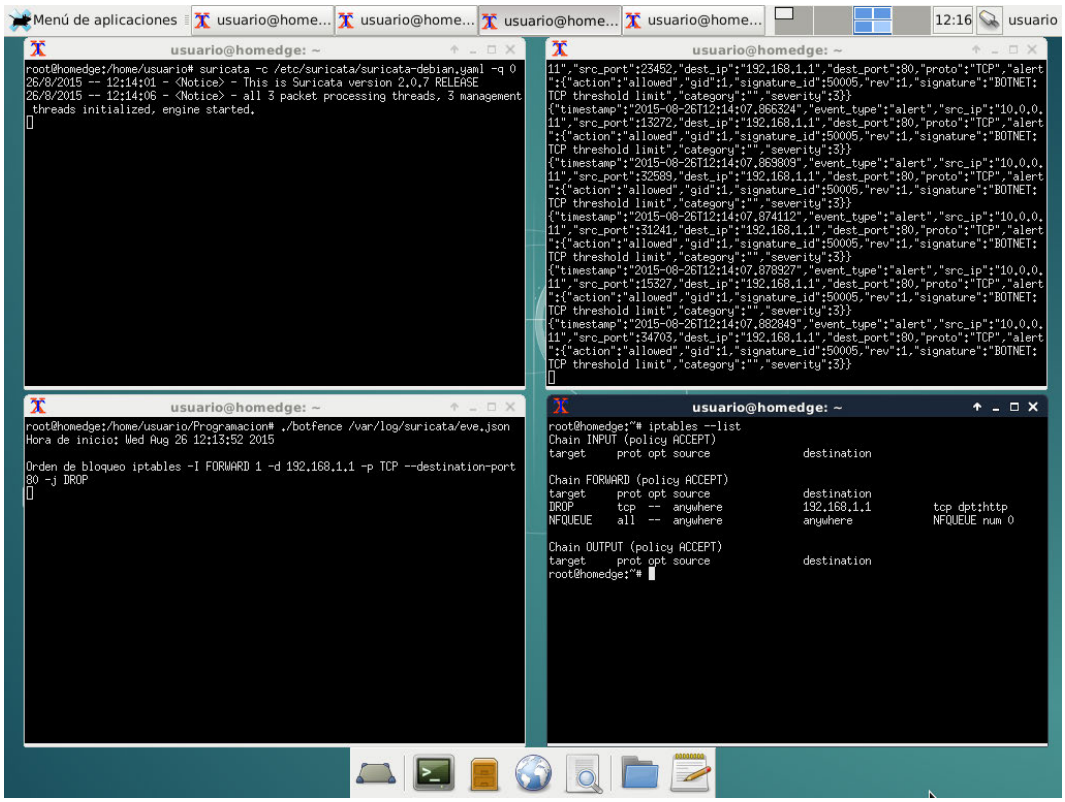


Ilustración 9. Prueba de HTTP Flood

Se repite de nuevo la prueba pero con IP Spoofing. Suricata no registra ningún evento porque, al igual que ocurre con los otros protocolos, la regla de bloqueo que ha introducido Botfence está establecida por equipo destino, por lo que sirve igualmente para bloquear ataques que hagan IP Spoofing a la misma víctima.

9.1.4. DNS Flood

El ataque DNS Flood produce paquetes UDP que son interceptados mediante la regla de bloqueo que evita el UDP Flood, por tanto, las pruebas que se han realizado para UDP Flood confirman que los ataques DNS Flood también se bloquean.

9.1.5. Envío masivo de correos electrónico

Para simular el envío masivo de correo, he creado el siguiente script en Perl que realiza el envío de 25 correos a un servidor:

```
#!perl
use warnings;
use strict;
use Net::SMTP;
use Net::SMTP_auth;

my $smtpserver = "mail.victima.org";
my $smtpport = 25;
my $smtpuser = "usuario";
my $smtppwd = "password";

my $smtp = Net::SMTP_auth->new($smtpserver, Port=>$smtpport, Timeout=>30, Debug=>1) or die "No se pudo conectar al servidor\n";
$smtp->auth('LOGIN', $smtpuser, $smtppwd) or die "Error en autentificación";

for(my $i=0; $i <= 25; $i++) {
    $smtp->mail("usuario\@victima.org");
    $smtp->to("incauto\@victima.org");
```

```
    $smtp->data();
    $smtp->datasend("To: incauto\@victima.org\n");
    $smtp->datasend("Esto es una prueba de envío masivo\n");
    $smtp->dataend();
}
$smtp->quit;
```

Lanzo el script en Perl desde KALI a un servidor web real y se detecta el ataque en BOTFENCE, tal como muestra la Ilustración 10, siendo la primera terminal la ejecución de Botfence, que indica que se ha bloqueado el envío y la terminal inferior el log de Suricata con la alerta:

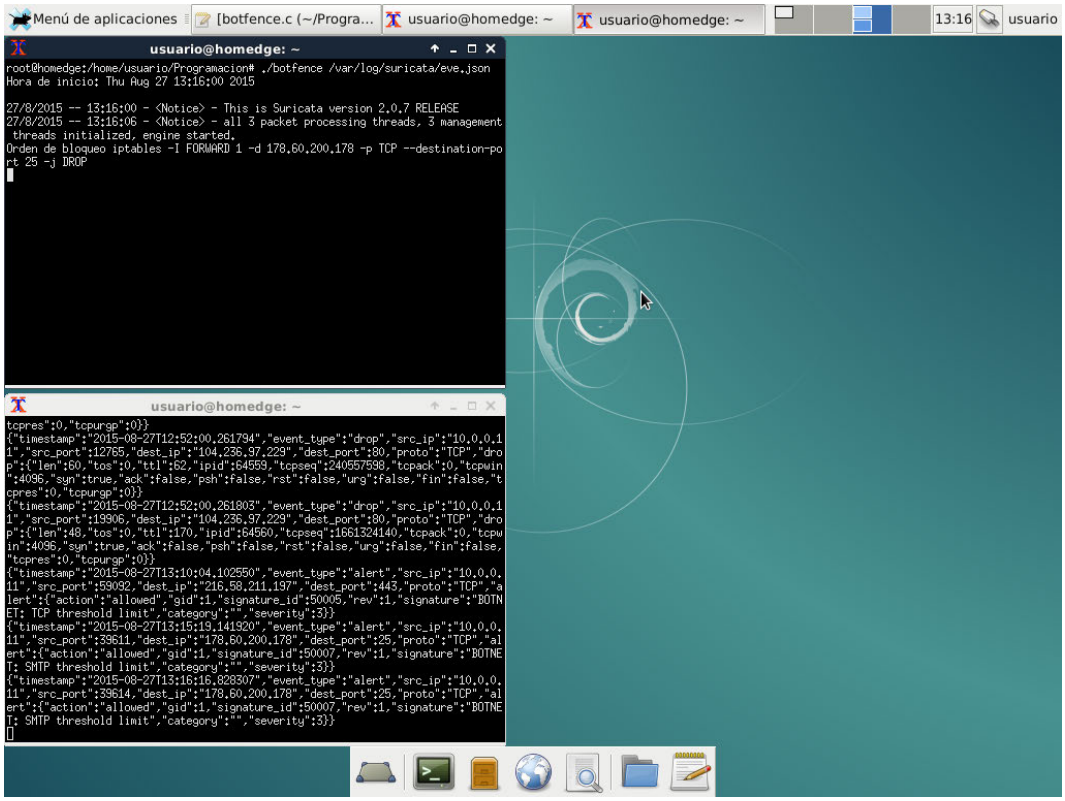


Ilustración 10. Bloqueo de envío masivo de correo

9.1.6. Comunicación con C&C

Para realizar la prueba de bloqueo de las comunicaciones con servidores C&C, escojo una de las direcciones IP que hay en el archivo /etc/suricata/rules/botcc.rules. En concreto, la IP 118.219.232.134 y pruebo con

el navegador de la máquina anfitrión a acceder a esa dirección, mostrando la página de la Ilustración 11.

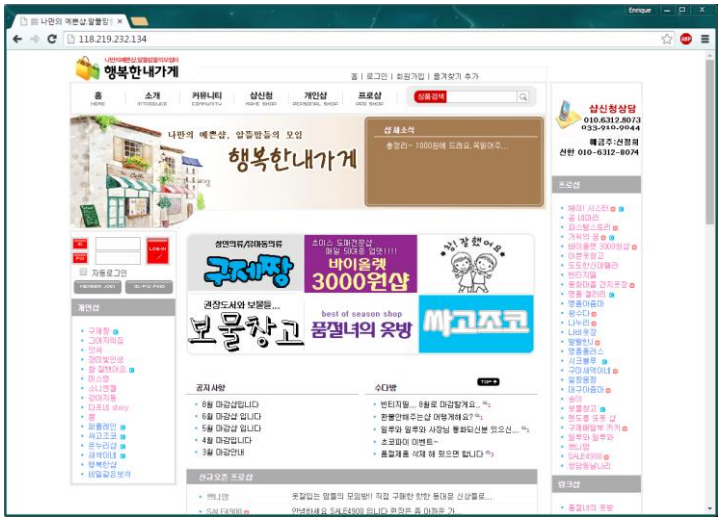


Ilustración 11. Navegación a dirección de servidor C&C

Hago la misma prueba desde KALI, con Botfence y Suricata funcionando. No se recibe respuesta desde el servidor y en el log de Suricata aparece un evento indicando que se ha bloqueado el acceso por ser un servidor que se ha reportado como C&C, tal como muestra la Ilustración 12, en el terminal de arriba a la derecha.

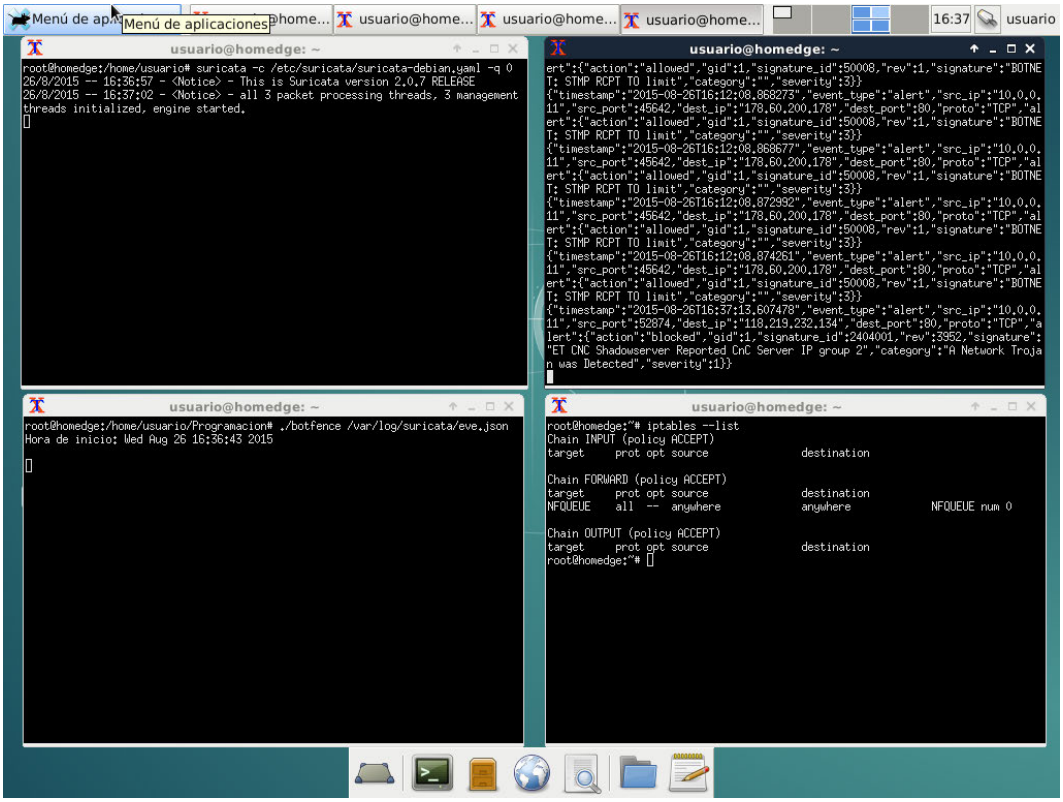


Ilustración 12. Bloqueo de acceso a servidor C&C

9.2.Ataques con SDBot

SDBot es una botnet desarrollada en C++ con Visual Studio 6. El control es por canal IRC, por lo que el servidor C&C deberá tener instalado un servidor IRC. Las primeras versiones están diseñadas para instalarse mediante ingeniería social con campañas de email o descarga de archivos y posterior ejecución, no incorpora técnicas de instalación por exploit.

Lleva insertados en el ejecutable los parámetros de configuración, que se deben establecer en el código fuente previamente a su compilación. En la Ilustración 13 se muestran los parámetros que indican el nombre de la botnet, el password de control, el servidor IRC, el puerto de conexión y el nombre del canal.

El bot hace las siguientes acciones al ejecutarse:

- Copiarse en una carpeta del sistema operativo con el nombre que le hayamos indicado en los parámetros.
- Ejecutar una instancia del ejecutable copiado.

- Cambiar la configuración de arranque para iniciarse cuando se inicie el sistema operativo.
- Conectarse al canal del servidor IRC y esperar ordenes.

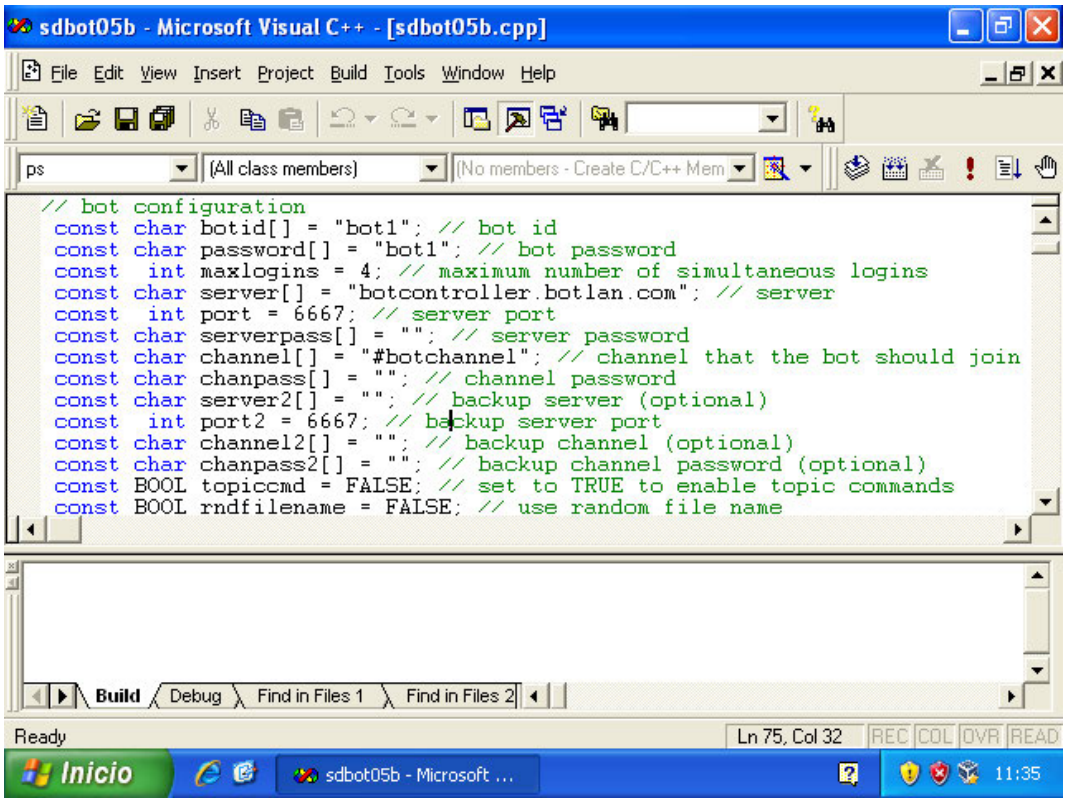


Ilustración 13. Parámetros de configuración de SDBot

Para efectuar las pruebas, he montado el entorno de la Ilustración 14 en el laboratorio virtual. Simula una red compuesta por dos máquinas infectadas situados en la misma red local que BOTFENCE, VIRULENTO1 y VIRULENO2, y dos máquinas externas. Una de ellas es la máquina que recibirá los ataques, llamada VICTIMA. La otra máquina, BOTCONTROLLER, hace las funciones de servidor de C&C, para lo que le he instalado el software UnrealIRCd, que es un servidor IRC. He compilado SDBOT para que tenga a BOTCONTROLLER como servidor IRC. Para visualizar las conexiones al IRC y los ataques, ejecutaré la aplicación Wireshark en VICTIMA. Como VICTIMA está en la misma red que BOTCONTROLLER, será capaz de ver todo el tráfico que va hacia BOTCONTROLLER al poner la tarjeta de red en modo promiscuo con Wireshark.

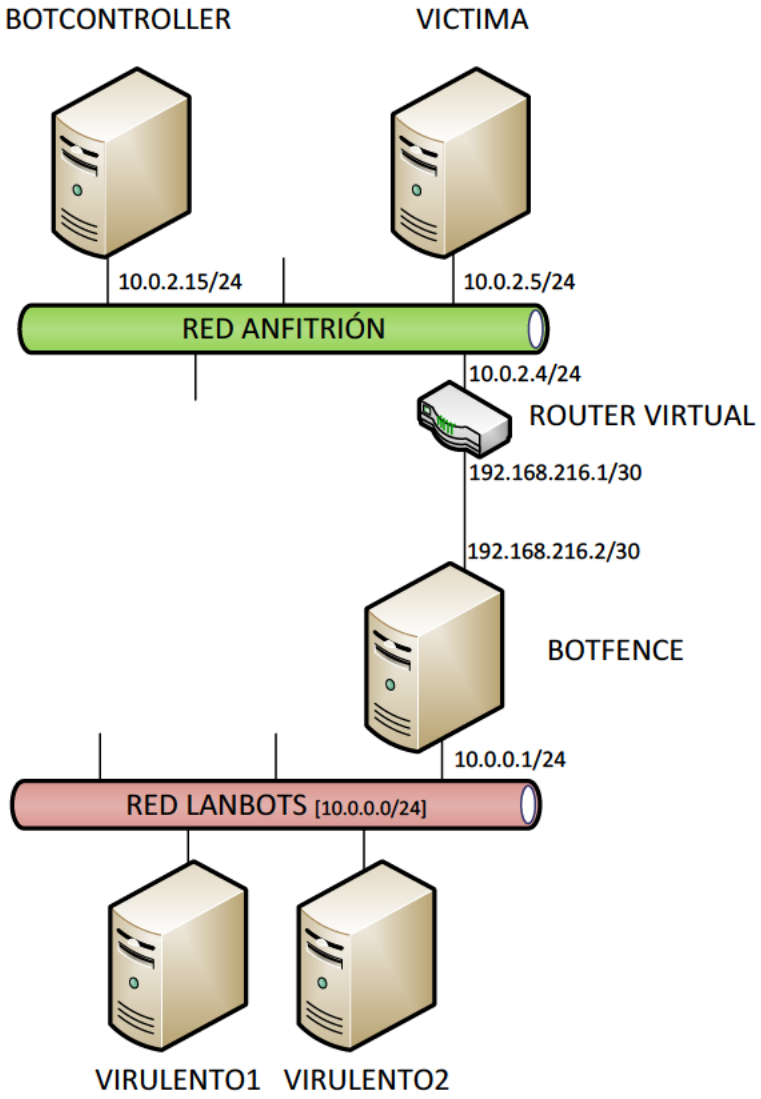


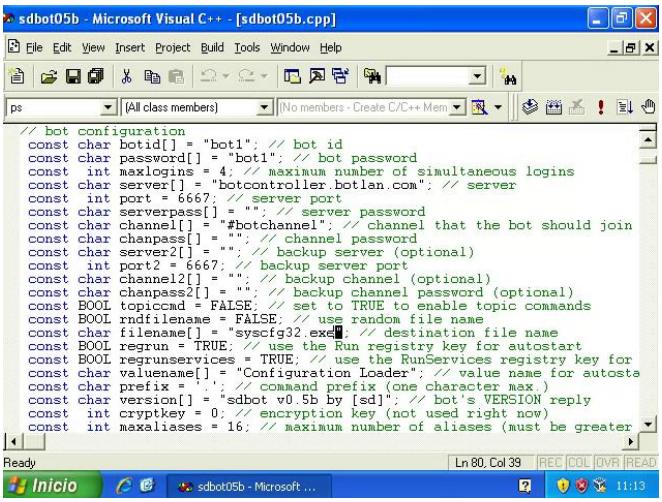
Ilustración 14. Entorno ejecución SDBot

9.2.1. Ataque sin bloqueo

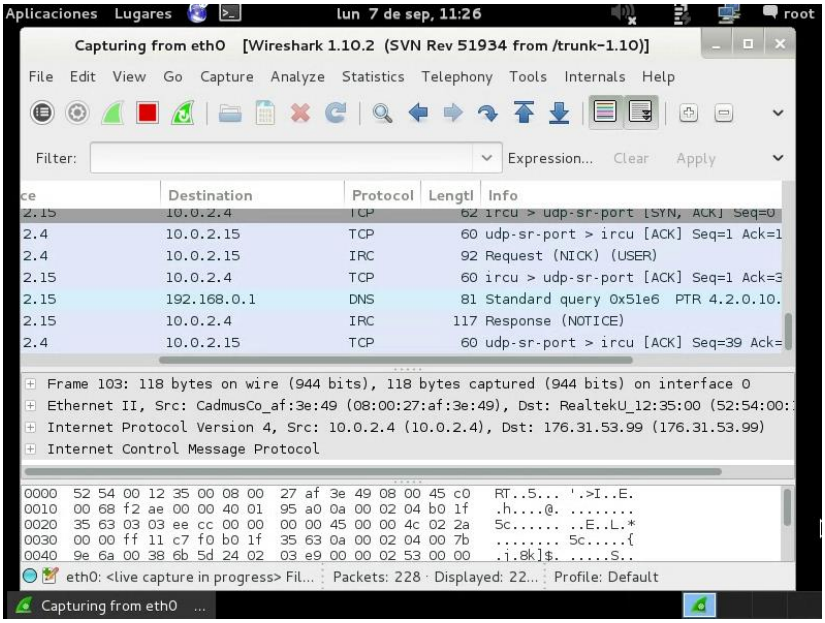
En esta prueba lanzo un ataque sin tener en funcionamiento la detección y bloqueo de botnets, por lo que llegará a la víctima íntegramente.

Efectúo los siguientes pasos:

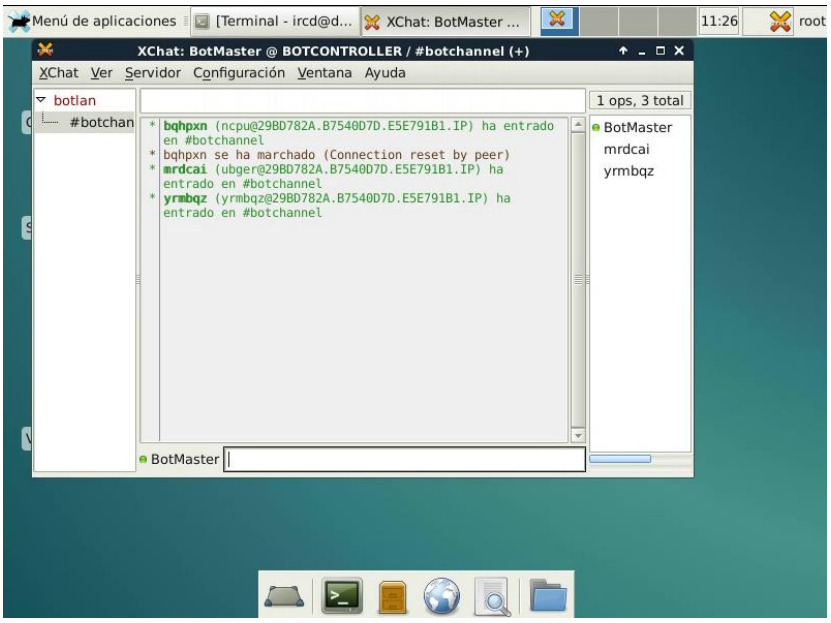
1. Compilo SDBot para que se conecte al canal #botchannel del servidor C&C BOTCONTROLLER. El password de control será password ‘bot1’.



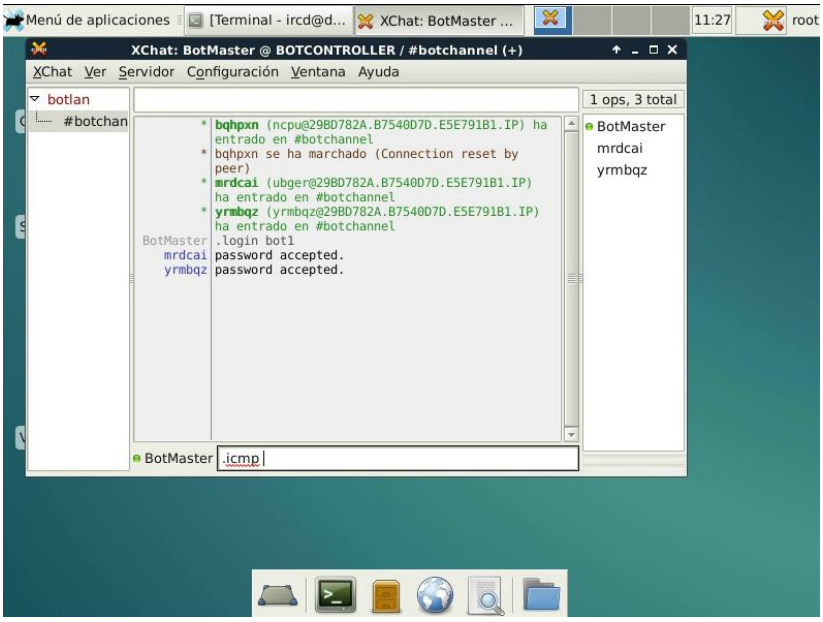
2. Me conecto al canal #botchannel desde BOTCONTROLLER desde el cliente de IRC XChat, con nombre de usuario ‘BotMaster’.
3. Ejecuto en VICTIMA la aplicación Wireshark.
4. Instalo el ejecutable en VIRULENTO1 y VIRULENTO2.
5. En Wireshark se ve el tráfico IRC que se genera desde la red LANBOTS, que es la IP 10.0.2.4 al estar detrás de ROUTER, hacia 10.0.2.15, que es BOTCONTROLLER. Son los dos bots conectándose al servidor C&C.



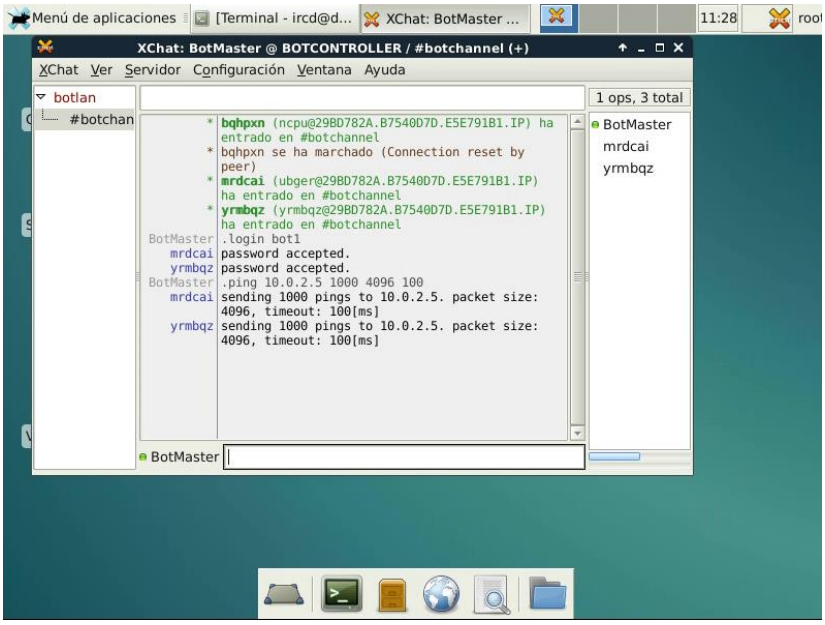
6. Se conectan los dos bots al canal de IRC.



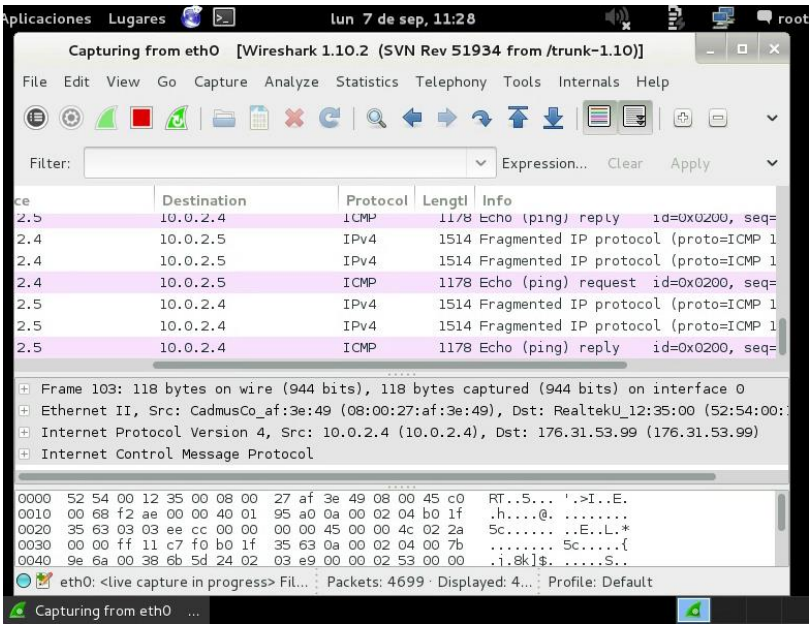
7. Pongo los bots bajo control con la siguiente orden en el IRC '.login bot1', que es el password establecido en la configuración. Los dos bots contestan aceptando el password.



8. Lanzo un ataque DDoS consistente en 1000 pings a 10.0.2.5, que es VICTIMA, con un tamaño de 4096 bytes y un tiempo entre envíos de 100ms.



9. En VICTIMA se visualiza el ataque con Wireshark.

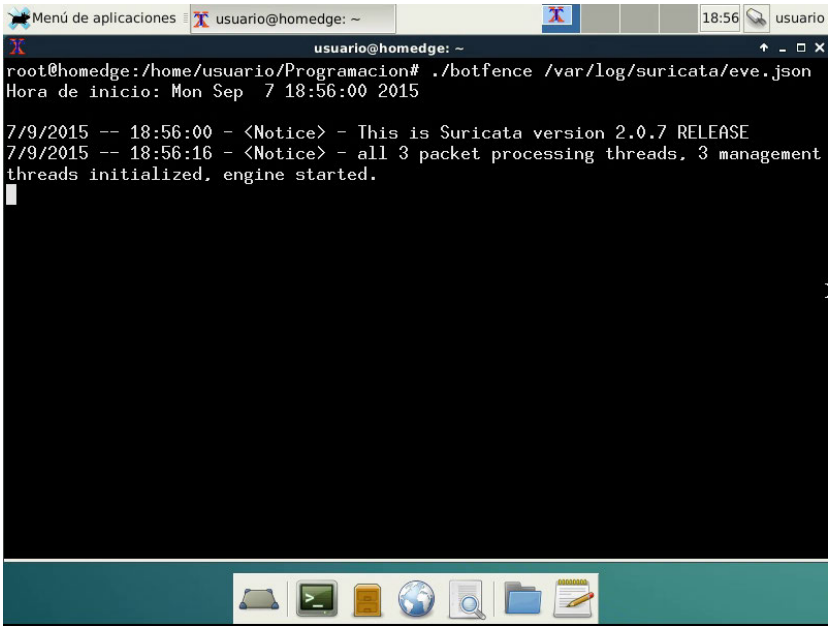


Doy por finalizada la prueba. Se ha conseguido realizar un ataque DDoS desde la botnet SDBot que ha sido visualizado con la aplicación Wireshark.

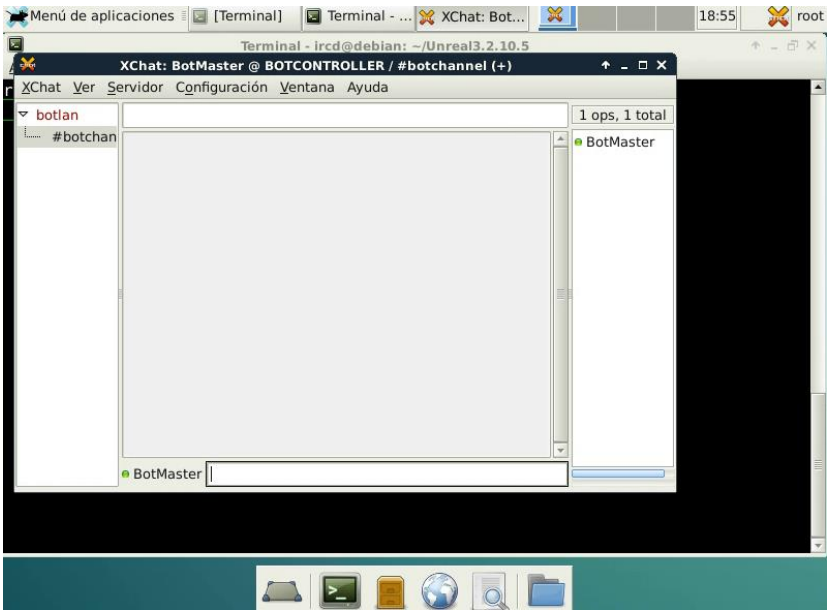
9.2.2. Ataque con bloqueo

En esta prueba lanzo un ataque con la máquina de detección y bloqueo de botnets activada, por lo que al detectarse el ataque, debería pararse.

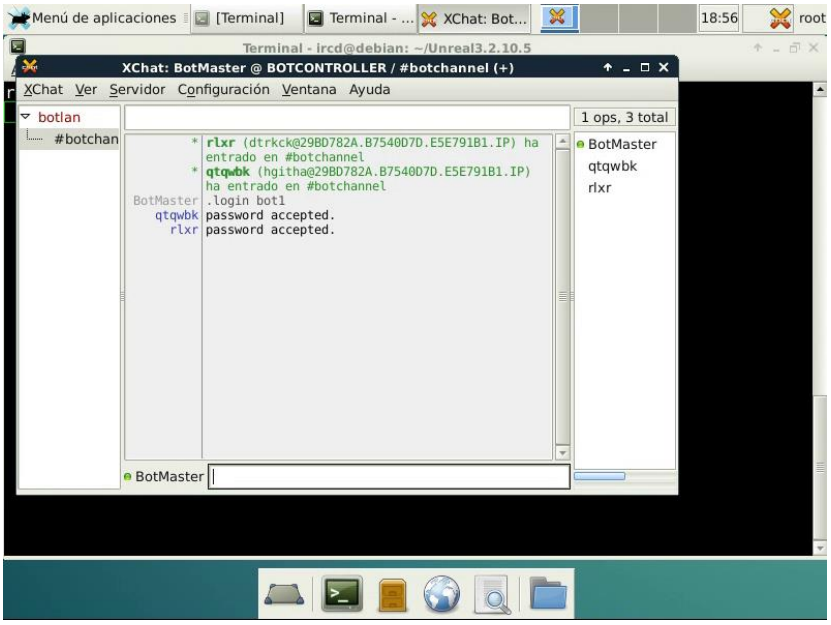
1. Lanzo la detección y bloqueo de botnets en BOTFENCE.



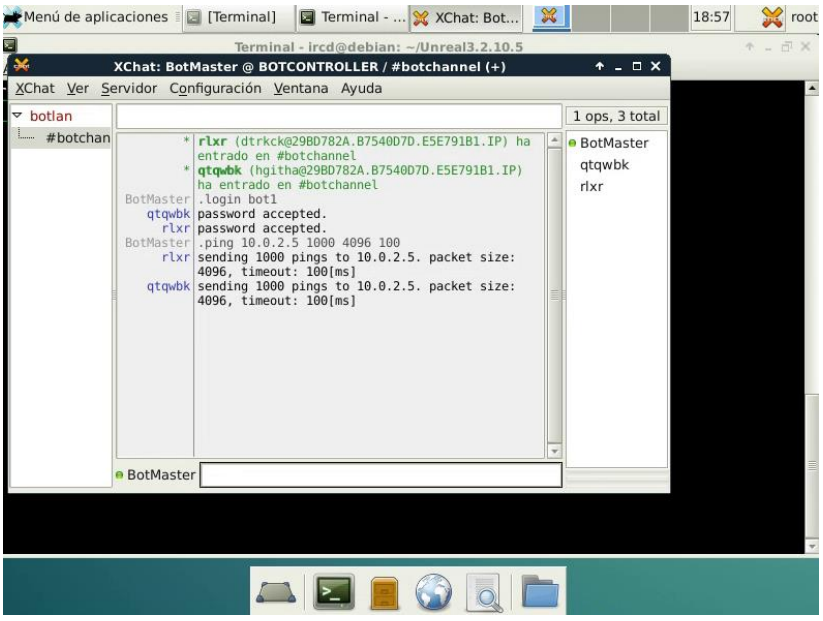
2. Inicio una sesión de chat en el canal #botchannel en el servidor C&C BOTCONTROLLER.



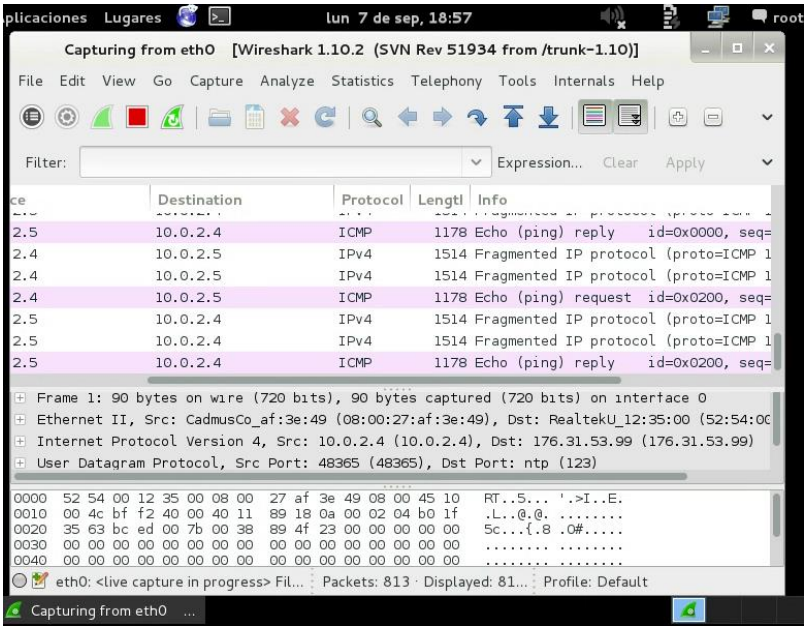
3. Inicio Wireshark en el ordenador VICTIMA para visualizar el tráfico de la red.
4. Arranco las máquinas zombies VIRULENTO1 y VIRULENTO2.
5. Los dos bots se conectan a la sesión de chat. Les indico el password de control para poder darles órdenes.



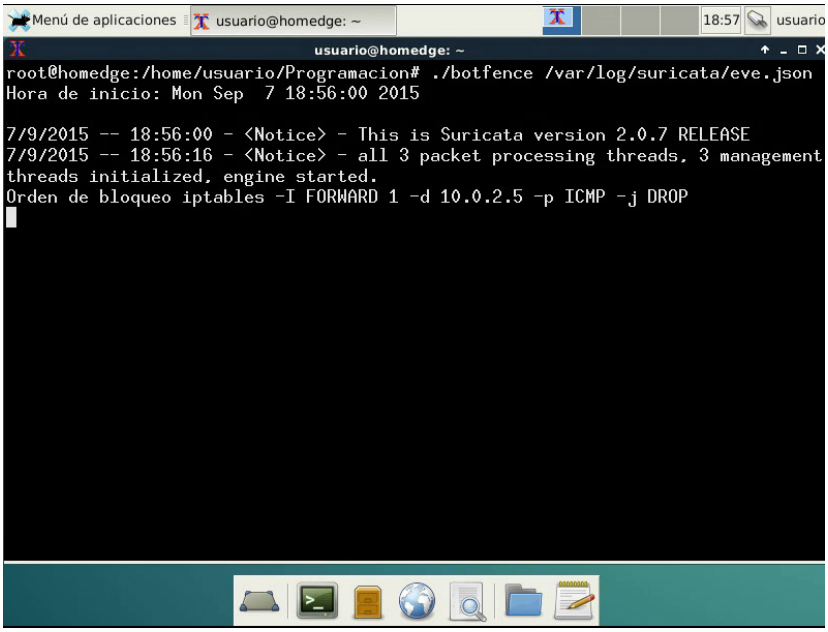
6. Lanzo un ataque DDoS contra VICTIMA. Es un ataque de 1000 paquetes ICMP de ping de 4096 bytes con 100 ms entre ellos.



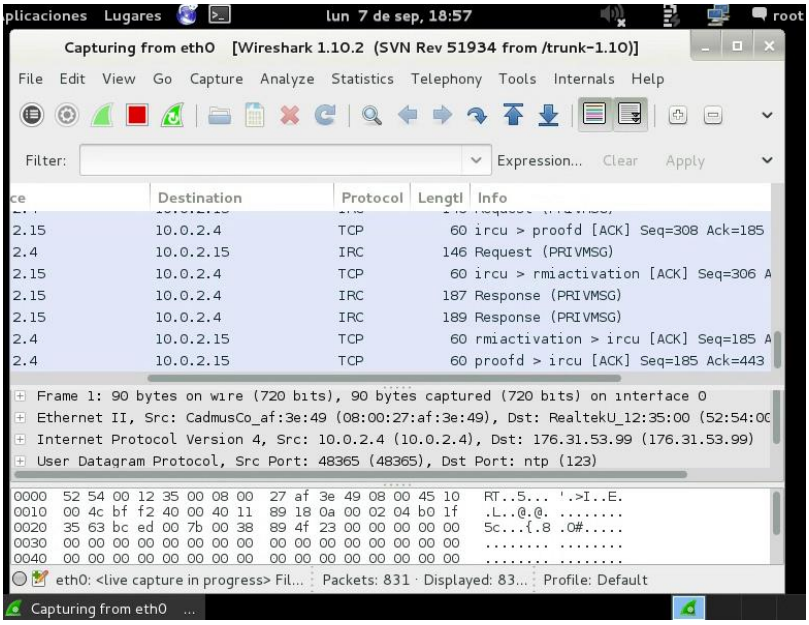
7. Llegan algunos pings a VICTIMA.



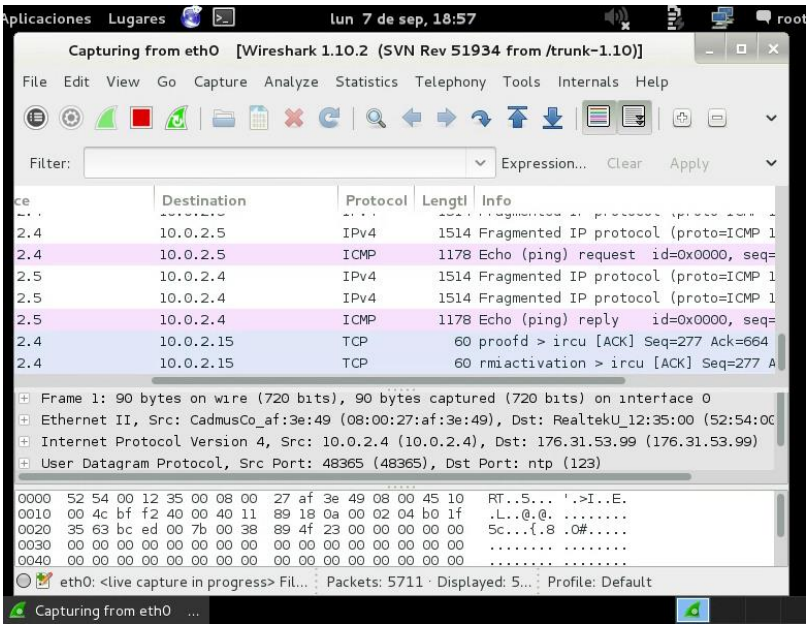
8. BOTFENCE detecta el ataque y lo bloquea. Indica en pantalla la orden de bloqueo.



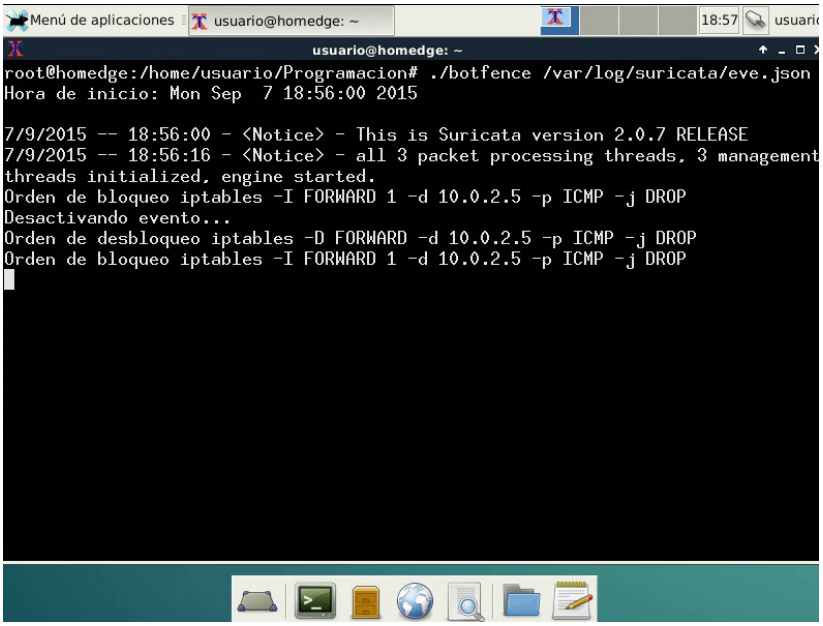
9. Se vuelve a lanzar el ataque. No llega ningún ping a VICTIMA. En Wireshark se ve el tráfico de los mensajes del IRC entre el servidor C&C y los bots, pero no llega ningún paquete ICMP porque BOTFENCE los está bloqueando.



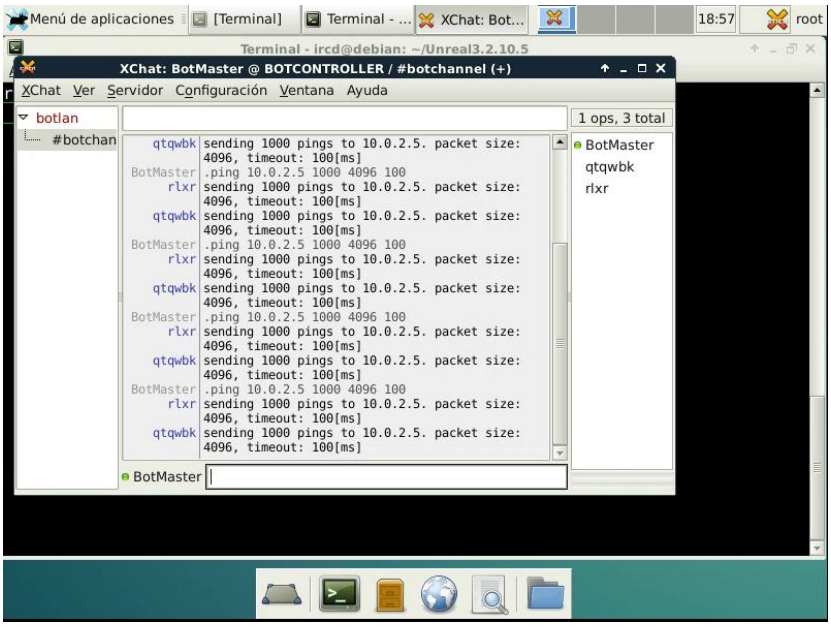
10. Se vuelve a lanzar el ataque. Llega algún ping a VICTIMA.



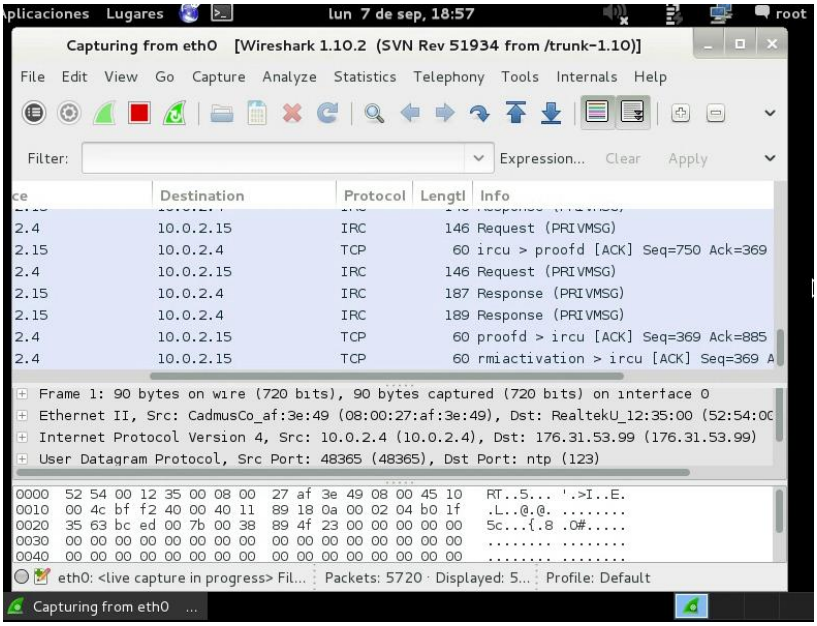
11. Inmediatamente se bloquea el ataque por BOTFENCE. Esto es debido a que se ha cumplido el tiempo de bloqueo configurado en los parámetros de BOTFENCE pero como el ataque se repite, se vuelve a bloquear de nuevo el doble de tiempo, según se ha establecido en el algoritmo de bloqueo para que su funcionamiento sea autónomo sin requerir intervención del usuario. BOTFENCE indica en pantalla que se ha bloqueado un ataque, se ha desbloqueado y se ha vuelto a bloquear.



12. Se lanzan tres ataques seguidos.



13. No llega ningún paquete a VICTIMA.



Doy por finalizada la prueba. El funcionamiento de la detección y el bloqueo de botnets ha sido satisfactorio. Se han bloqueado los ataques en el momento de dar comienzo. Este sería un comportamiento típico de bloqueo de botnets de día 0 en el que aún no se reconocen por las firmas.

9.3.Ataques con ZBot/Zeus Botnet

ZBot es una botnet con comunicación con el servidor C&C por HTTP. La comunicación entre los bots y el C&C está encriptada por una clave común. Es una botnet especializada en el robo de información, aunque existen muchas variantes, que permiten incluso la encriptación de la información para pedir una recompensa (ransomware).

El código fuente se compone de una parte en servidor, desarrollada en PHP y que se tiene que instalar en un servidor web accesible por los bots, y una aplicación de generación de bots, que está desarrollada en Visual C++ sobre Visual Studio 2010. La aplicación de generación de bots permite crear el ejecutable del bot combinando los parámetros de configuración, que especifican, entre otras cosas, la localización del C&C, y el archivo con las definiciones de inyección de código HTML en las páginas web, que tienen las reglas de modificación de las páginas que se visualizan en el navegador de la máquina infectada.

Para compilar el código, hay que ajustar los parámetros del archivo '/make/buildconfig.inc.php', estableciendo las carpetas donde se encuentran instalados Visual Studio y las librerías, según muestra la Ilustración 15.

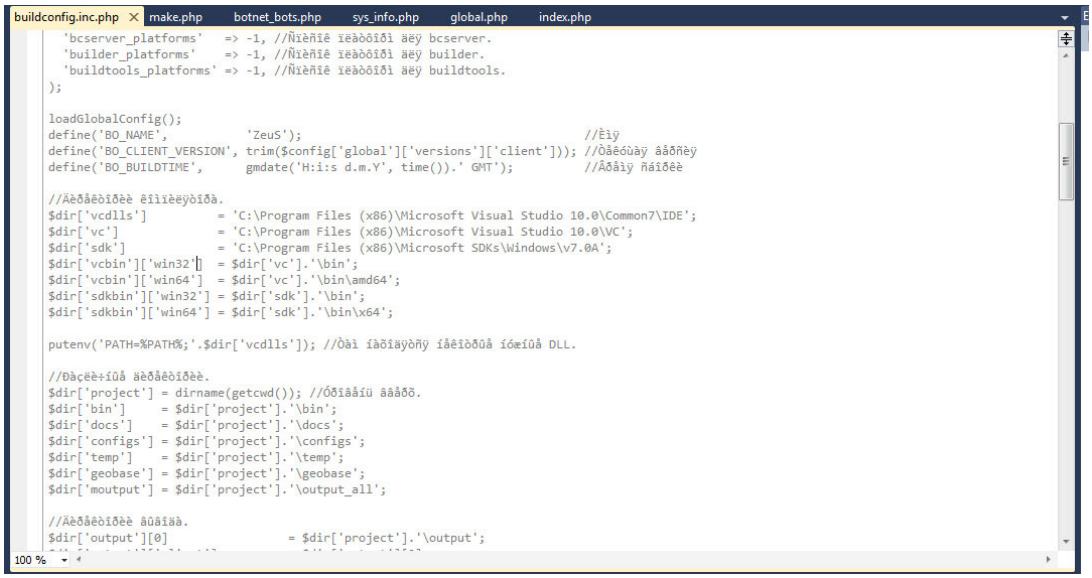


Ilustración 15. Configuración para compilar ZBot

Los ejecutables '/bin/7z.exe' y '/bin/upx.exe' que vienen en el código fuente no funcionan correctamente en Windows 7 64 bits, por lo que se deben de obtener desde otras fuentes y sustituirlos.

Para lanzar la compilación, se debe ejecutar el archivo 'make_full.cmd', que se encuentra en la raíz. Cuando finalice, se habrán creado dos archivos dentro de la carpeta 'output'. Uno de ellos es un archivo comprimido y otro es el password para descomprimirlo.

El archivo comprimido contiene cuatro carpetas. Las páginas del servidor C&C se encuentran en la carpeta 'server[php]'. Hay que instalarlas en un servidor web con el módulo de PHP habilitado y la base de datos MySQL. Para configurarlo, hay que editar los archivos '/install/index.php' y '/system/global.php', acceder con el navegador web a 'http://<servidor_C&C>/install/index.php', Ilustración 16, comprobar que los parámetros son correctos y pulsar el botón 'Install', que creará las tablas en la base de datos y el archivo '/system/config.php'.

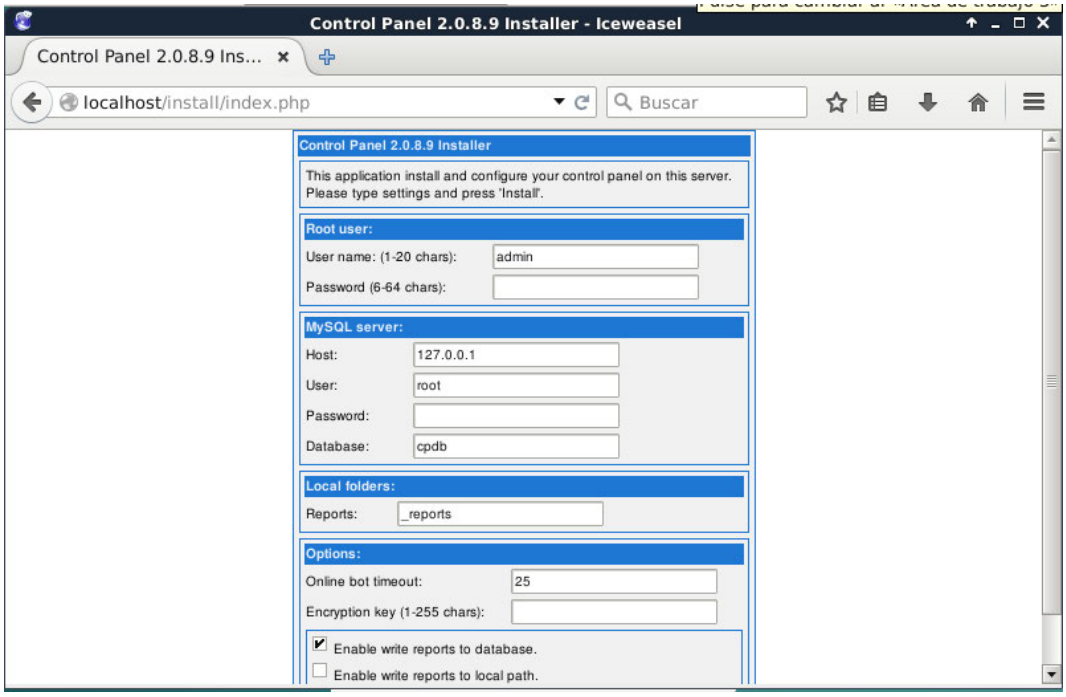


Ilustración 16. Configuración del C&C de ZBot

La generación del bot se lleva a cabo ejecutando el archivo '\builder\zsb.exe'. Los parámetros de configuración se establecen en '\builder\config.txt'. Permite generar nuevas configuraciones del bot para ponerlas en el servidor C&C y que los bots se actualicen automáticamente, y generar el ejecutable del bot con la configuración establecida, según el botón que se pulse, tal como se observa en la Ilustración 17.

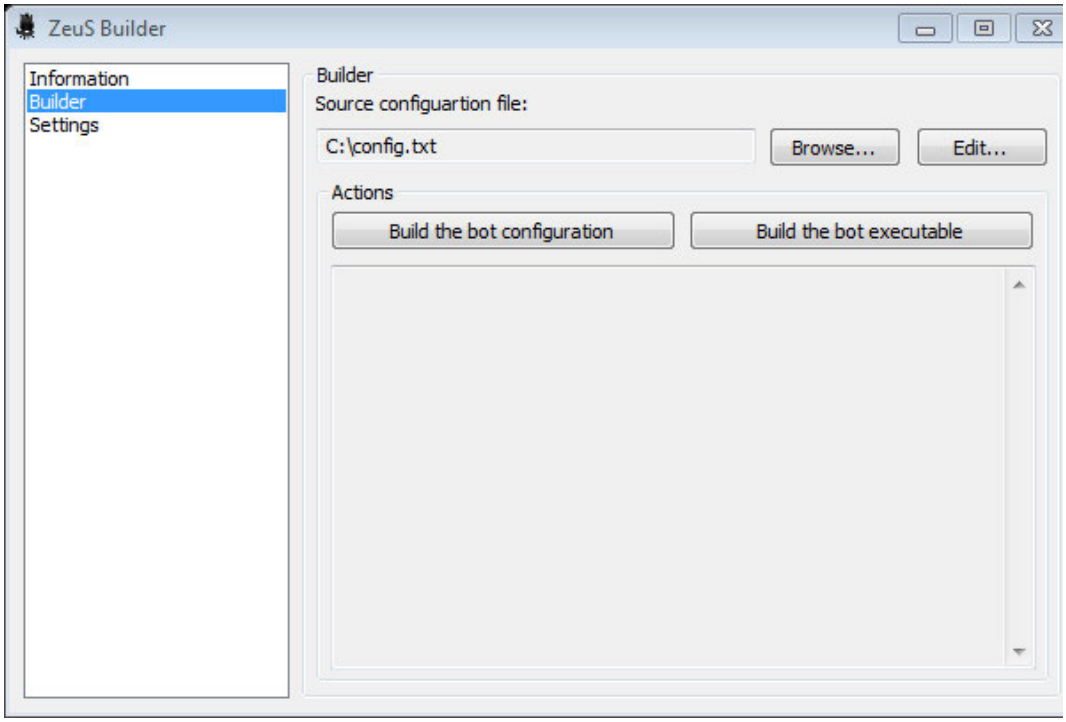


Ilustración 17. Generación del bot de ZBot

La infección de las máquinas se debe llevar a cabo mediante ingeniería social y, una vez infectada, se pondrá en contacto con el servidor C&C configurado.

Para efectuar la prueba, he simplificado la red de la prueba anterior en el laboratorio virtual. En este caso, se dispone de un ordenador infectado, VIRULENTO1, el servidor C&C al que se conectará el bot, BOTCONTROLLER, y BOTFENCE, que tiene como objetivo detectar la actividad del bot y bloquear sus comunicaciones. En la Ilustración 18 se muestra de forma gráfica. La prueba consistirá primero en demostrar que se está robando información privada desde VIRULENTO1 para, posteriormente, detener el robo de información por medio de BOTFENCE.

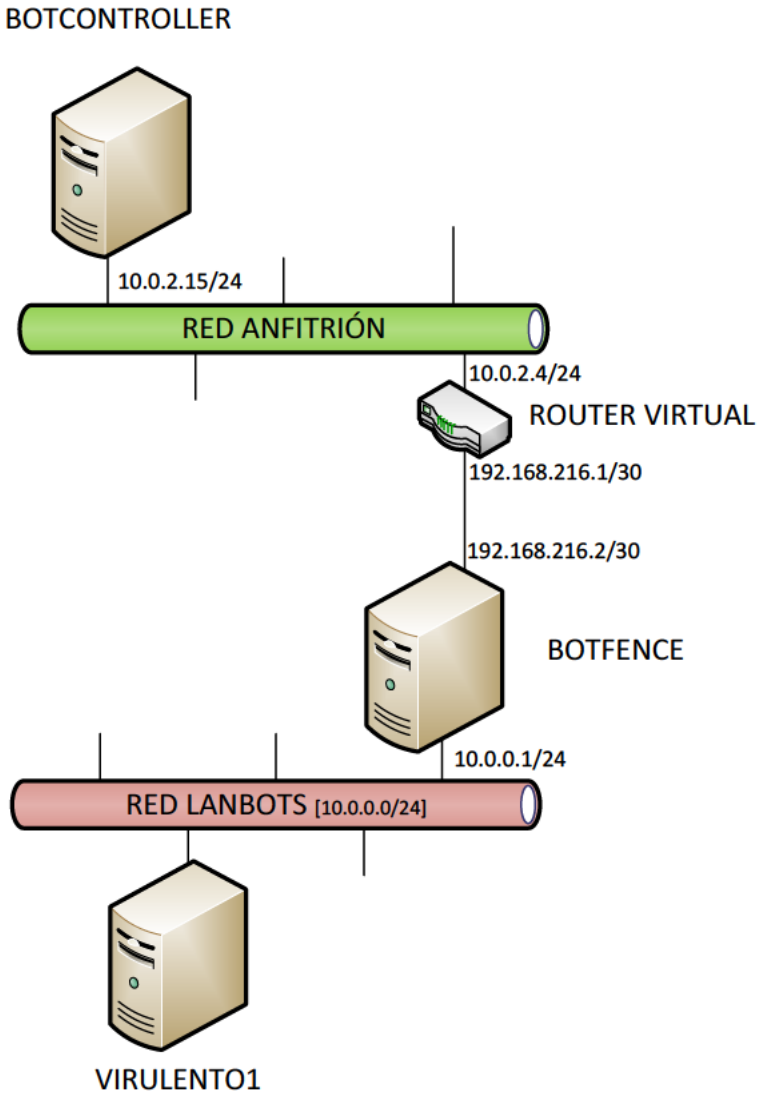
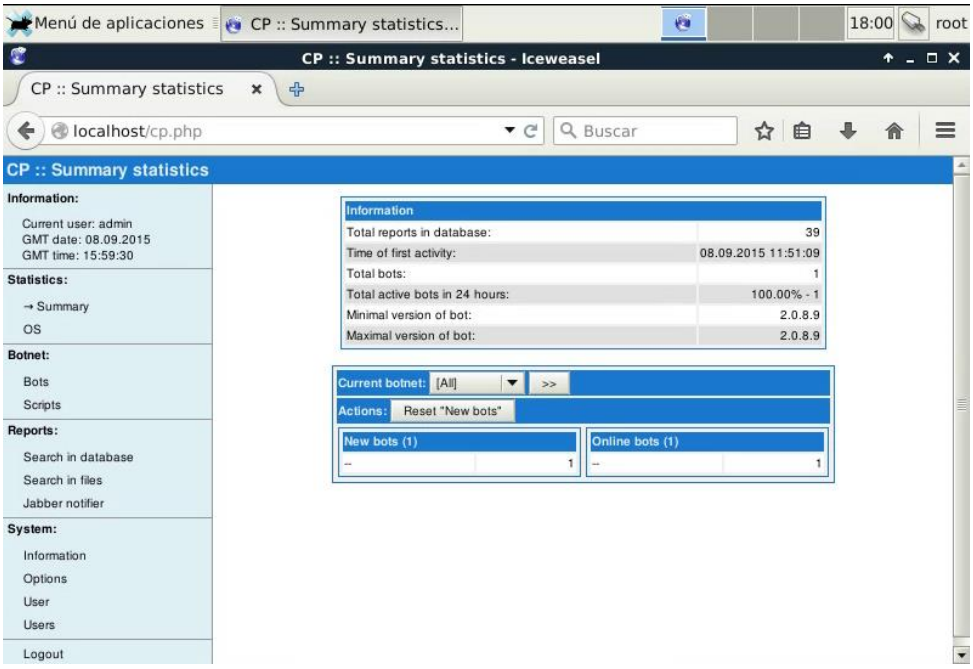


Ilustración 18. Entorno ejecución ZBot

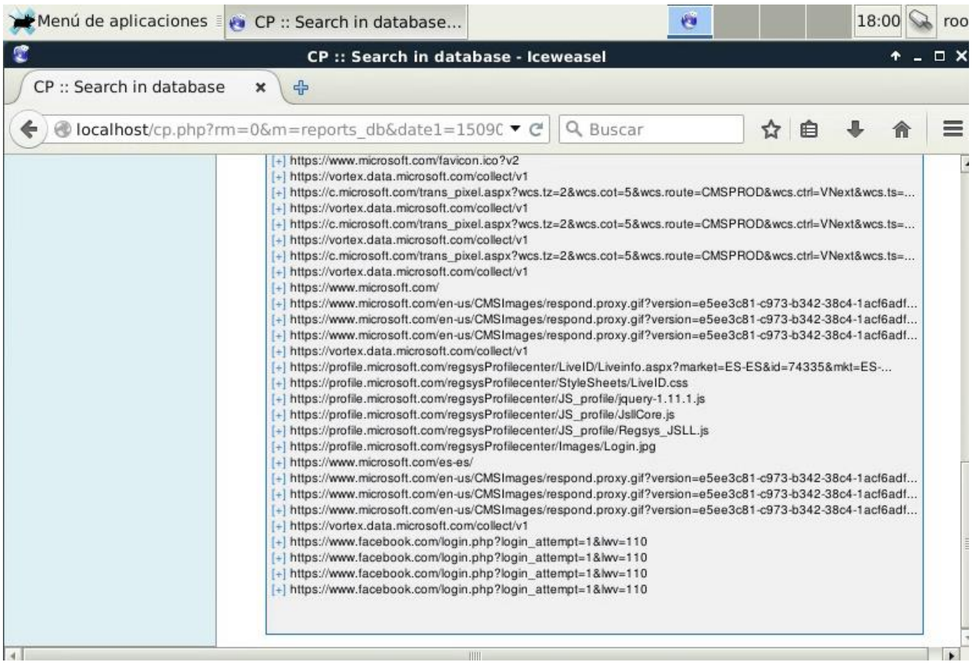
9.3.1. Ataque sin bloqueo

En esta prueba, me conecto a Facebook en VIRULENTO1 para demostrar como ZBot roba la información y la envía al servidor C&C, BOTCONTROLLER. Los pasos de la prueba son:

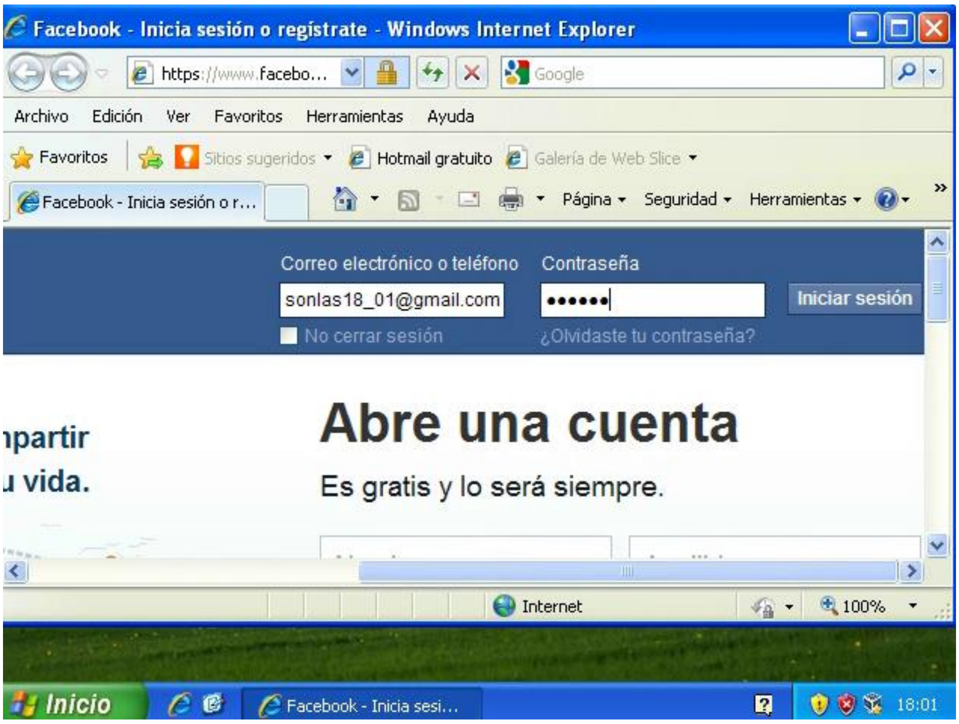
1. Arranco las máquinas que formarán parte de la prueba: ROUTER VIRTULA, BOTFENCE, VIRULENTO1 Y BOTCONTROLLER.
2. Abro un navegador en BOTCONTROLLER apuntando a la interface de control de los bots. La pantalla de resumen indica que hay un bot activo, que es VIRULENTO1.



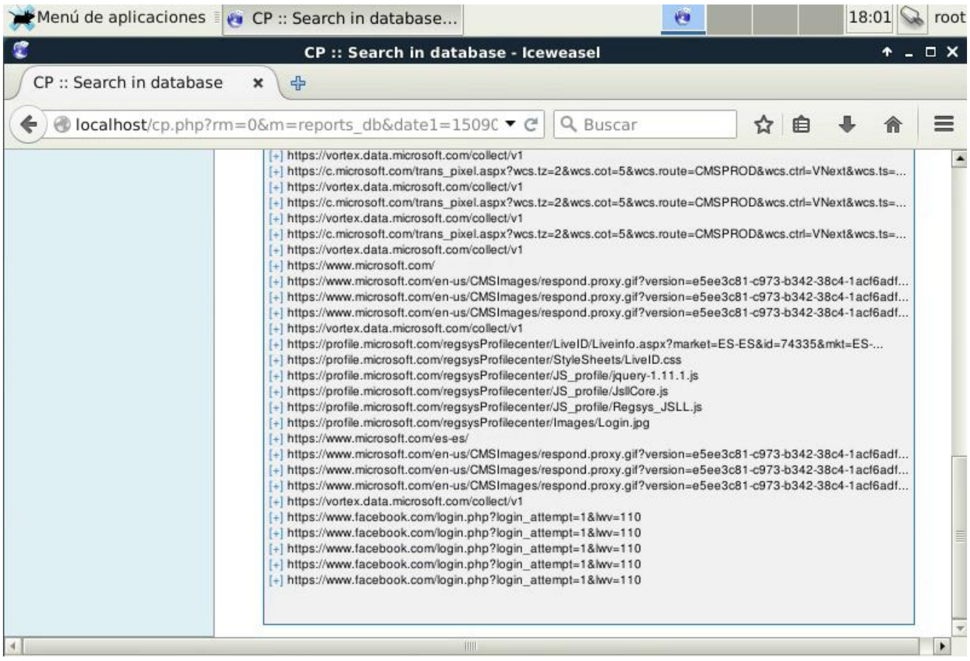
3. En BOTCONTROLLER, compruebo el registro de informes recibidos de los bots. Los últimos cuatro son de acceso a Facebook.



4. Abro un navegador en VIRULENTO1 apuntando a Facebook y hago login con el usuario ‘sonlas18_01@gmail.com’ y la contraseña ‘sesamo’.

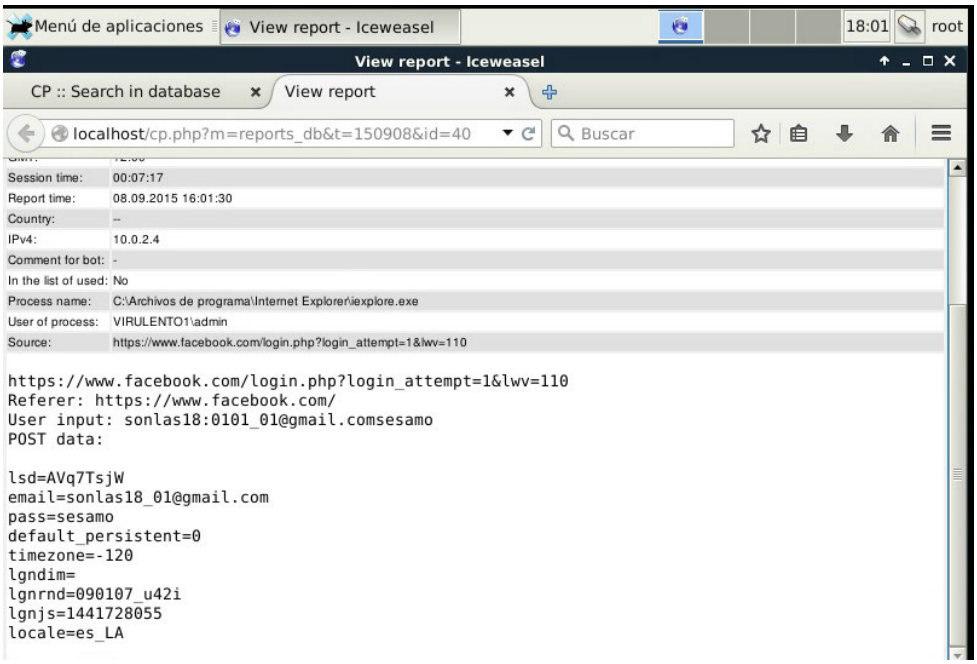


5. En el menú de control de los bots de BOTCONTROLLER, vuelvo a solicitar el registro de informes de los bots. En esta ocasión, aparecen cinco informes de Facebook.



6. Selecciono el último informe para visualizarlo y muestra los datos que se han utilizado para entrar en Facebook en VIRULENTO1. Se confirma que

el bot está funcionando, ha robado las credenciales de entrada del sitio web y las ha enviado al servidor C&C.

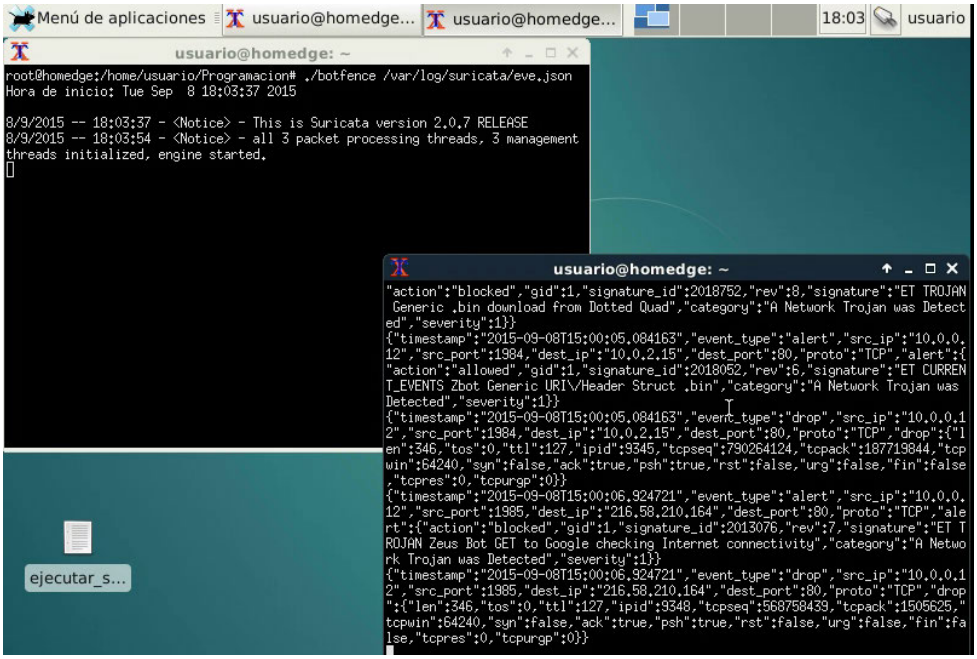


Doy por finalizada la prueba. Se ha conseguido robar las credenciales de acceso a un sitio web desde una máquina infectada y obtenerlas en el servidor C&C.

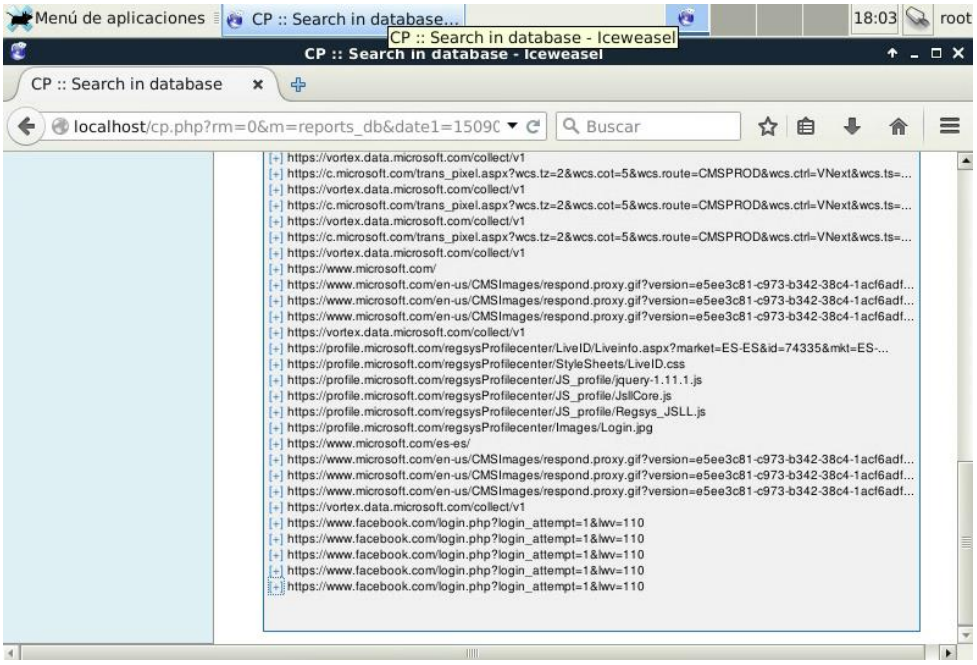
9.3.2. Ataque con bloqueo

En esta prueba, continúo a partir de la prueba anterior asegurando que si la detección y bloqueo de botnets está activada, no se roban credenciales en la red.

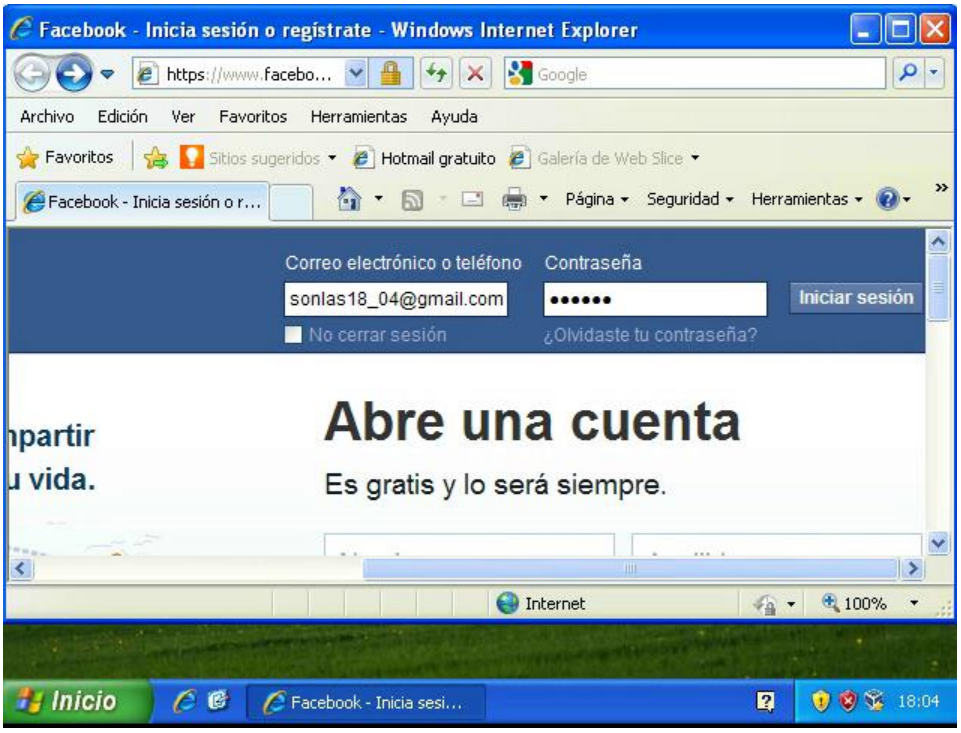
1. Activo en BOTFENCE la detección y bloqueo de botnets.



2. Compruebo en BOTCONTROLLER el registro de informes de los bots. Lo último que se ha recibido son cinco informes de acceso de Facebook.



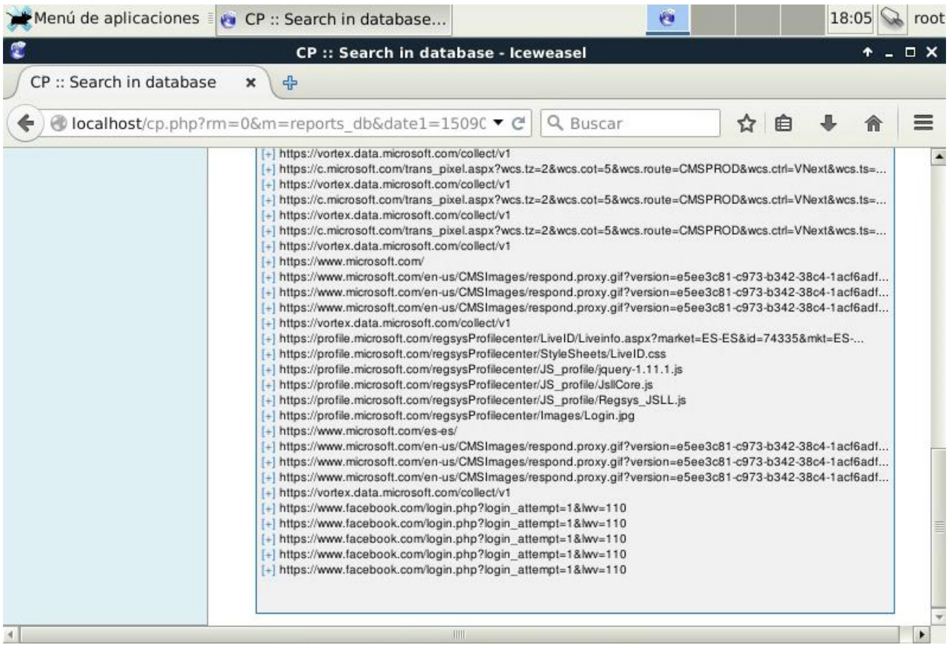
3. En VIRULENTO1, accedo de nuevo a Facebook con el usuario 'sonlas18_04@gmail.com' y la contraseña 'sesamo'.



4. En BOTFENCE se detecta el envío de información al C&C y se bloquea. No deberían de haber llegado las credenciales de entrada al Facebook que ha enviado VIRULENTO1 a BOTCONTROLLER ya que los últimos tres registros del log de BOTFENCE indican que se han descartado paquetes de actividad de troyano enviados por la IP 10.0.0.12, que es la IP de VIRULENTO1. El segundo registro concreta que el troyano es ZBot.

{"timestamp":"2015-09-08T18:04:46.524479","event_type":"alert","src_ip":"10.0.0.12","src_port":1081,"dest_ip":"10.0.2.15","dest_port":80,"proto":"TCP","alert":{"action":"blocked","gid":1,"signature_id":2017930,"rev":9,"signature":"ET TROJAN Trojan Generic - POST To gate.php with no referer","category":"A Network Trojan was Detected","severity":1}}
{"timestamp":"2015-09-08T18:04:46.524479","event_type":"alert","src_ip":"10.0.0.12","src_port":1081,"dest_ip":"10.0.2.15","dest_port":80,"proto":"TCP","alert":{"action":"blocked","gid":1,"signature_id":2019141,"rev":3,"signature":"ET TROJAN Zbot POST Request to C2","category":"A Network Trojan was Detected","severity":1}}
{"timestamp":"2015-09-08T18:04:46.524479","event_type":"drop","src_ip":"10.0.0.12","src_port":1081,"dest_ip":"10.0.2.15","dest_port":80,"proto":"TCP","drop":{"len":371,"tos":0,"ttl":127,"ipid":796,"tcpseq":1217761393,"tcpack":499159827,"tcpwin":64240,"syn":false,"ack":true,"psh":true,"rst":false,"urg":false,"fin":false,"tcpres":0,"tcpurgp":0}}

5. Accedo al registro de informes de bots en BOTCONTROLLER y no se han recibo nuevos informes. El último informe es el de la prueba anterior, por lo que el envío de las credenciales desde VIRULENTO1 ha sido bloqueado.



Doy por finalizada la prueba. El funcionamiento de la detección y el bloqueo de botnets ha sido satisfactorio. Se ha bloqueado el envío de información confidencial por un bot de la red a un servidor C&C. Este sería un comportamiento típico de bloqueo de botnets por firma.

10. Conclusiones y trabajo futuro

Las botnets son un tipo de malware que, debido a sus características de funcionamiento, se vuelve más peligroso conforme va aumentando el número de máquinas a las que infecta. Los ataques producidos por botnets grandes son capaces de dejar fuera de línea cualquier sistema conectado a Internet colapsando su ancho de banda.

Los ataques de botnets producidos por volumen del tipo DDoS consumen todos los recursos de la víctima, imposibilitando la prestación del servicio a los usuarios legítimos. Pueden llegar a dejar fuera de línea completamente a un sistema durante todo el tiempo que dure el ataque.

La fuerza del ataque es mayor cuanto más cerca se está de la víctima y, por tanto, es más débil en el origen, por lo que es más sencillo detener el ataque cuánto más cerca de los bots que forman la red se actúe.

En este trabajo he desarrollado una sistema que permite detectar el ataque en la red local del bot, justo antes de que salga a Internet. Es uno de los puntos donde el ataque es más débil y aún no ha alcanzado el volumen suficiente como para saturar el ancho de banda o la capacidad de proceso de los dispositivos de la red.

El sistema de detección y bloqueo de botnets obtenido permite su integración en cualquier tipo de instalación y puede utilizarse por usuarios noveles en informática, ya que una vez puesto en marcha, es autónomo, totalmente transparente en su funcionamiento y no impide el uso normal de las máquinas infectadas por alguna botnet. La arquitectura software funciona en máquinas con pocos recursos, no requiere un hardware de última generación, lo que permite utilizar un ordenador de pequeñas dimensiones que ocupe poco espacio y su consumo eléctrico sea mínimo.

La configuración se realiza mediante archivos de texto, lo que puede dificultar su instalación, por lo que se debería crear una interface que funcionase con una pantalla táctil para facilitar su configuración.

Como el sistema se basa en una distribución completa de Linux y debido a su colocación como router de la red, permite instalar otros servicios añadidos, como proxy web con cache, dns local y control parental.

El laboratorio virtual construido es un entorno que permite ejecutar muestras de malware de forma aislada, sin poner en riesgo otras máquinas de la red. Aunque inicialmente estaba compuesto de cuatro máquinas, permite su ampliación para disponer de una red completa de máquinas con distintos sistemas operativos y versiones de software, que pueden llegar a simular una red doméstica grande, dependiendo de la potencia que tenga la máquina anfitrión sobre la que se ejecute, por lo que es aprovechable para investigaciones posteriores. Se ha utilizado esta capacidad de ampliación y reconfiguración para infectar con dos botnets las máquinas y hacer pruebas de detección y bloqueo que han permitido verificar el correcto funcionamiento del sistema.

La detección de botnets desarrollada se basa en tres técnicas: la detección de las comunicaciones con los servidores que controlan la botnet (C&C), el descubrimiento de direcciones IP falsas y la ejecución de ataques. El primer sistema de detección, el sistema de detección de la comunicación con servidores C&C, tiene el problema de que se basa en firmas y, al igual que los antivirus, el malware de día 0, es decir, aquel que aún no ha sido detectado en la red, no lo detecta. El bot se descubriría posteriormente cuando comenzase un ataque.

En las pruebas se ha replicado estos tipos de bloqueo con dos botnets reales que tiene distintas características, SDBot y ZBot. SDBot es una botnet con control por canal IRC y que permite realizar ataques DDoS. En este caso, se ha lanzado un ataque que se ha detectado en inmediatamente y se ha bloqueado. Con ZBot, que está especializado en el robo de credenciales, se ha utilizado la técnica

basada en firmas y se ha producido el bloqueo de los paquetes con credenciales de acceso a sitios web que enviaba el bot al C&C.

Existen otros patrones de comportamiento de las botnets, como el envío de información a servidores de forma regular, el uso de redes P2P con archivos de pequeño tamaño o el intercambio de mensajes encriptados o carentes de sentido en redes sociales, que pueden llegar a utilizarse en botnets y que habría que investigar e integrar con el fin de aumentar la fiabilidad del sistema y detener los ataques antes de producirse.

11. Bibliografía

[1] F. Telefónica, «La Sociedad de la Información en España 2014», Ariel, Barcelona, 2015.

[2] C. A. Schiller, J. Binkley, D. Harley, G. Evron, T. Bradley, C. Willems y M. Cross, Botnets. The Killer Web App, Syngress, 2007.

[3] CISCO, «Detecting Botnet Traffic with the Cisco Cyber Threat Defense Solution 1.0», CISCO, 2012.

[4] McAfee, «Ten Days of Rain. Expert analysis of distributed denial-of-service attacks targeting», McAfee, 2011.

[5] M. Goncharov, «Russian Underground 101», Trend Micro Incorporated, 2012.

[6] G. Gu, P. Porras, Y. Vinod, M. Fong y W. Lee, «BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation», de *16th USENIX Security Symposium*, 2007.

[7] G. Gu, R. Perdisci, J. Zhang y W. Lee, «BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection», de *17th USENIX Security Symposium*, 2008.

[8] Radware, «DDoS Protection and Attack Mitigation», Radware, [En línea]. Available: http://www.radware.com/Products/DefensePro/. [Último acceso: 09 08 2015].

[9] M. A. Rajab, J. Zarfoss, F. Monrose y A. Terzis, «A Multifaceted Approach to Understanding the Botnet Phenomenon», de *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, ACM New York, 2006, pp.

41-52.

[10] J. Canavan, «The Evolution of Malicious IRC Bots», Symantec, 2005.

[11] S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed y S. A. Khayam, «A Taxonomy of Botnet Behavior, Detection, and Defense», *IEEE Communications Surveys & Tutorials,* vol. 16, nº 2, pp. 898-924, Second Quarter 2014.

[12] A. Nappa, A. Fattori, M. Balduzzi, M. Dell'Amico y L. Cavallaro, «Take a Deep Breath: A Stealthy, Resilient and Cost-Effective», de *Detection of Intrusions and Malware & Vulnerability Assessment*, Berlin, 2006.

[13] A. Berger y M. Hefeeda, «Exploiting SIP for botnet communication», de *Secure Network Protocols, 2009. NPSec 2009. 5th IEEE Workshop on*, Princeton, NJ, 2009.

[14] P. Porras, H. Saidi y V. Yegneswaran, «An analysis of Conficker's logic and rendezvous points», SRI International, 2009.

[15] X. Yu, B. Zhang, L. Kang y J. Chen, «Fast-Flux Botnet Detection Based on Weighted SVM», *Information Technology Journal,* vol. 8, nº 11, pp. 1048-1055, 2012.

[16] T. Brewster, «Twitter botnet creation made simple», IT Pro, 14 5 2010. [En línea]. Available: http://www.itpro.co.uk/623323/twitter-botnet-creation-made-simple. [Último acceso: 3 08 2015].

[17] A. Lelli, «Trojan.whitewell: What's your (bot) facebook status today?», Symantec, 31 10 2009. [En línea]. Available: http://www.symantec.com/connect/blogs/trojanwhitewell-what-s-your-bot-facebook-status-today. [Último acceso: 03 08 2015].

[18] D. Dittrich y S. Dietrich, «P2P as botnet command and control: a deeper insight», de *Malicious and Unwanted Software, 2008. MALWARE 2008.*,

2008.

[19] S. Dietrich, D. Dittrich, J. Hernandez y S. Stover, «Analysis of the Storm and Nugache trojans: P2P is here», *; L O G I N :,* vol. 32, nº 6, pp. 18-27, 2007.

[20] S. T. Ali, P. McCorry, P. H.-J. Lee y F. Hao, «ZombieCoin: Powering Next-Generation Botnets with Bitcoin», de *2nd Workshop on Bitcoin Research*, 2015.

[21] Trend Micro, «Trend Micro. Thread Encyclopedia. SDBOT.», 09 10 2012. [En línea]. Available: http://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/sdbot. [Último acceso: 26 07 2015].

[22] R. Tokazowski, «Disrupting an Adware-serving Skype Botnet», 03 06 2015. [En línea]. Available: http://phishme.com/disrupting-an-adware-serving-skype-botnet/. [Último acceso: 15 08 2015].

[23] M. Noga, E. Carter y M. Lee, «MS14-063 A Potential XP Exploit», 01 12 2014. [En línea]. Available: http://blogs.cisco.com/security/talos/ms14-063-a-potential-xp-exploit. [Último acceso: 13 08 2015].

[24] Trend Micro, «Pawn Storm Update: Trend Micro Discovers New Java Zero-Day Exploit», 11 07 2015. [En línea]. Available: http://blog.trendmicro.com/trendlabs-security-intelligence/pawn-storm-update-trend-micro-discovers-new-java-zero-day-exploit/. [Último acceso: 12 08 2015].

[25] E. Eng y D. Caselden, «Operation Clandestine Wolf – Adobe Flash Zero-Day in APT3 Phishing Campaign», 23 06 2015. [En línea]. Available: https://www.fireeye.com/blog/threat-research/2015/06/operation-clandestine-wolf-adobe-flash-zero-day.html. [Último acceso: 16 08 2016].

[26] L. Constantin, «Adobe confirms zero-day exploit bypasses Adobe Reader

sandbox», 14 02 2013. [En línea]. Available: http://www.pcworld.com/article/2028163/adobe-confirms-zeroday-exploit-bypasses-adobe-reader-sandbox.html. [Último acceso: 12 08 2015].

[27] L. Constantin, «Dangerous Internet Explorer vulnerability opens door to powerful phishing attacks», 03 02 2015. [En línea]. Available: http://www.pcworld.com/article/2879372/dangerous-ie-vulnerability-opens-door-to-powerful-phishing-attacks.html. [Último acceso: 12 08 2015].

[28] C. Sanders, Practical packet analysis, 2nd edition, No Starch Press, Inc., 2011.

[29] HUAWEI, ARP Security Technology White, HUAWEI TECHNOLOGIES CO., LTD, 2012.

[30] C. Sanders y J. Smith, Applied Network Security Monitoring, Syngress, 2014.

[31] Debian, «Debian Network Configuration», 22 01 2015. [En línea]. Available: https://wiki.debian.org/es/NetworkConfiguration. [Último acceso: 10 04 2015].

[32] Debian, «DHCP Server», 25 11 2013. [En línea]. Available: https://wiki.debian.org/DHCP_Server. [Último acceso: 10 04 2015].

[33] R. Russell, «Linux 2.4 NAT HOWTO», 14 01 2002. [En línea]. Available: http://www.netfilter.org/documentation/HOWTO//NAT-HOWTO.html. [Último acceso: 14 04 2015].

[34] J. Hutchen, Kali Linux Network Scanning Cookbook, Packt Publishing, 2014.

[35] Netmarketshare, «Desktop Operating System Market Share», 1 07 2015. [En línea]. Available: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0. [Último acceso: 2 08 2015].

[36] McAfee Inc., «McAfee® Network Security Platform [formerly IntruShield]

Denial-of-Service [DoS] Prevention Techniques», McAfee, 2013.

[37] M. Goldstein, «BoNeSi - the DDoS Botnet Simulator», Markus Goldstein, 21 05 2015. [En línea]. Available: https://github.com/markus-go/bonesi. [Último acceso: 12 07 2015].

[38] The Honeynet Project, «Dionaea», The Honeynet Project, [En línea]. Available: http://dionaea.carnivore.it/. [Último acceso: 3 04 2015].

12. Siglas, abreviaturas y acrónimos

Botnet: red de ordenadores infectados por un malware común bajo el control de un operador y que permite realizar acciones de forma coordinada, por ejemplo lanzar ataques DDoS.

Bot: ordenador zombi infectado por un malware que pertenece a una botnet.

C&C: del inglés Command and Control, se refiere a la forma de control centralizada de una botnet.

DDoS: del inglés Distributed Denial of Service, es un ataque realizado por varias máquinas de forma coordinada sobre un sistema cuyo fin es impedir que preste sus servicios con normalidad.

ICMP: del inglés Internet Control Message Protocol, es un protocolo utilizado para el control y notificación de errores. Está definido en la RFC 792. La orden ping lo utiliza para obtener el tiempo que tarda un paquete en llegar al destino.

IDS: Del inglés Intrusion Detection System, es un software que permite detectar que el intrusiones en el sistema monitorizando la red.

IPS: De inglés Intrusion Prevention System, es un IDS que añade la funcionalidad de bloquear las intrusiones.

SEM: del inglés Security Event Management. Software o servicio que monitoriza y gestiona los eventos de seguridad que se producen en la red.

SIEM: del inglés Security Information and Event Management. Software o servicio que monitoriza, analiza, proporciona informes y gestiona los eventos de seguridad que se producen en la red.

SIM: del inglés Security Information Management. Software o servicio que analiza y proporciona informes de los eventos de seguridad que se producen en la red, proporcionando almacenamiento a largo plazo.