Botnets: La amenaza fantasma

Trabajo de Fín de Master

Máster Interuniversitario en Seguridad de las TIC -MISTIC-

UOC

Autor. GUILLERMO CALVO ORTEGA

Directora. ÁNGELA MARÍA GARCÍA VALDÉS

Un trabajo sobre la amenaza BOTNET, su tipología, empleo y formas de mitigarla a través de un caso práctico.

INCIBE

ENERO 2018

LICENSE

Abstract

A través de los medios de comunicación nos hemos hecho eco de algunos ataques perpetrados por botnets, donde la opinión pública se alarma ante esta amenaza por un periodo corto de tiempo para, luego, al poco olvidarlo y volverse a creer a salvo.

Sin embargo, las botnets están con nosotros, en nuestros equipos informáticos y no somos conscientes de si realmente somos o nó esclavos de dicha red. Es como una amenaza “fantasma” de la que solo nos asustamos cuando nos hablan de ella para acabar olvidándola cuando dejamos de “verla”.

Con este trabajo se pretende conocer más en profundidad este tipo de redes, como funcionan, quien las dirige, cuando surgieron y si estamos a salvo o somos rehenes de ellas.

Se realizará una demostración práctica y un análisis en el que se pondrá de manifiesto su potencial, su flexibilidad de uso y su poder de ocultación y que nos proporcionará una aproximación final a esta amenaza real.

Through the media we have echoed some attacks perpetrated by botnets, where public opinion is alarmed by this threat for a short period of time, then, forget it in a short time and return to believe safe.

However, the botnets are with us, in our computer equipment and we are not aware of whether or not we are really slaves of that network. It is like a "ghost" threat that we only scare when they talk about it and forget it when we stop "seeing" it.

With this work we intend to know more in depth this type of networks, how they work, who directs them, when they arose and if we are safe or are hostages of it.

There will be a practical demonstration and an analysis in which its potential, its flexibility of use and its power of concealment will be revealed, which will provide us with a final approach to this real threat.

Dedication and acknowledgements

Quiero dar las gracias a mi familia, amigos y compañeros que durante este tiempo me han apoyado y comprendido para poder finalizar este master que tanto esfuerzo requiere.

Tambień quiero agradecer a todos los profesores de la UOC del Master Interuniversitario en Seguridad de las TIC (MISTIC) por su apoyo y dedicación y en particular a la tutora de este Trabajo de Fin de Master (TFM), Doña Angela María García Valdés (INCIBE), por su ayuda e impulso en la realización de este reto.

Por ultimo, quiero agradecer especialmente a mi mujer, Jessica, su esfuerzo, comprensión, cariño, apoyo y amor con el que me alentado durante este reto y del que he disfrutado desde la conocí. Este trabajo es merito tanto de ella como mío propio.

Table of Contents

List of Tables

Table **Page**

List of Figures

Figure **Page**

GLOSSARY

Adware ..... Cualquier programa que automáticamente muestra u ofrece publicidad no deseada.

APT ..... Advanced Persistent Threat. Procesos informáticos sigilosos y continuos, a menudo orquestados por humanos, dirigidos a penetrar la seguridad informática.

CERT/CSIRT ..... Computer Emergency Response Team/Computer Security Incident Response Team. Centro de respuesta a incidentes de seguridad en tecnologías de la información.

C&C ..... Command and Control. Sistema de gestión que gobierna uno o varios procesos.

DoS/DDoS ..... Distributed/Denial of Service. Ataque a un sistema de computadoras o red que causa que un servicio o recurso sea inaccesible a los usuarios legítimos

Feed RSS ..... Really Simple Syndication. Archivo generado por algunos sitios web que contiene una versión específica de la información publicada en esa web.

FQDN ..... Fully Qualified Domain Name. Nombre que incluye el nombre de la computadora y el nombre de dominio asociado a ese equipo.

HOSTS ..... Archivo de un ordenador es usado por el sistema operativo para guardar la correspondencia entre dominios de internet y direcciones IP.

IDS/IPS ..... Intrusion Detection/Prevention System. Software que ejerce el control de acceso en una red informática para proteger a los sistemas computacionales de ataques y abusos.

IoT ..... Internet Of Things. Concepto que se refiere a la interconexión digital de objetos cotidianos con Internet.

IRC ..... Internet Relay Chat. Protocolo de comunicación en tiempo real basado en texto, que permite debates entre dos o más personas.

Keylogger ..... Tipo de software o un dispositivo hardware específico que se encarga de registrar las pulsaciones que se realizan en el teclado, para posteriormente memorizarlas en un fichero o enviarlas a través de internet.

NAT ..... Network Address Traslation. Técnica empleada para que varios dispositivos de una red local puedan conectarse a internet a través de una o varias direcciones IP globales.

P2P ..... Peer-to-Peer. Red de ordenadores en la que todos o algunos aspectos funcionan sin clientes ni servidores fijos, sino una serie de nodos que se comportan como iguales entre sí.

RAT ..... Remote Access/Administration Tool. Aplicación malware que incluye una puerta trasera para tener un control sobre el equipo en el que está instalado.

Rootkit ..... Conjunto de software que permite un acceso de privilegio continuo a una computadora pero que mantiene su presencia activamente oculta al control de los administradores al corromper el funcionamiento normal del sistema operativo o de otras aplicaciones.

SandBox ..... Entorno de pruebas o cerrado que aísla los cambios en el código, fruto de la experimentación, del propio entorno de producción.

Sock ..... Constituyen el mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red a los procesos o hilos apropiados. Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto.

CHAPTER 1

INTRODUCCIÓN

Existe una multitud de usuarios que erróneamente piensan que un malware (los mal llamados virus informáticos) es un programa que tiene autonomía (o conciencia) propia y, como tal, realiza una serie de acciones para conseguir su objetivo. Esa forma de verlo es debido a la incesante manera que tiene el ser humano de establecer una similitud entre un ente y su descripción (virus). Sin embargo, la conciencia que realmente debe de preocuparnos esta siempre detrás de las maquinas, en aquellos que mueven los hilos, pudiendo gobernar un ejército de zombis para tomar el control de todo aquello que creemos tener a salvo.

Igual que una partida de ajedrez en la que nadie sacrifica a su reina en la primera jugada sino, más bien, a sus peones menos valiosos así se opera en el tablero de la ***red de redes****.*

Comencé este trabajo teniendo un concepto teórico de lo que era una botnet y de su existencia "real" en el mundo actual. Cuanto equivocado estaba sobre este tipo de redes y que ingenuo era cuando creía que su existencia seria marginal y sin apenas uso.

Muy al contrario de lo ampliamente divulgado sobre este malware, se estima que 1 de cada 30 equipos esta infectado por un bot y, por tanto, pertenece a una red brindándole sus recursos para actividades maliciosas. A mi entender esta estimación es poco realista pudiendo ser el ratio de equipos zombis mucha mayor que el indicado.

1.1 Escenario actual sobre botnets

Cuando se pretende obtener información sobre las mayores amenazas en internet, entre los primeros puestos aparecen el robo de datos, phishing, DoS/DDoS, ransomware, spaming...etc. En todos estos casos o están o pueden esta involucradas las botnets. Aunque existe una tendencia

a establecer una relación unidireccional entre las botnets y los ataques de DDoS, la verdad es que las botnets se emplean o pueden ser empleadas en casi cualquier tipo de ataque. Las botnets proporcionan, capacidad, flexibilidad, anonimato, resilencia, adaptación y capacidad de replicación o contagio.

Así pues, debemos de asociar la botnets no como un ataque en si, sino como una técnica que nos permite emplear cualquier tipo de ataque a través de una "herramienta" tan eficaz como una botnet. La amenaza que supone las botnets y su "invisibilidad" de cara al usuario hace que sea la herramienta cada vez mas empleada en el cibercrimen, surgiendo mercados que hasta hace años serian impensables, como el arriendo o venta de estas redes para la comisión de delitos por parte de cualquier delincuente incluidos organismos, empresas, gobiernos o particulares. Existe un "mercado negro" donde uno puede adquirir esta forma de ataque.

Hoy en día el uso de armas para cometer los crímenes no compensa económicamente y tiene mayor rédito económico (o de otro tipo) emplear un "arma" ya preparada para poder realizar la actividad criminal "sin mancharse las manos". Obviamente no solo los creadores de estas redes estarán sujetos a un hipotético castigo por su actividad delictiva, sino aquellos que las adquieren o "alquilan" para conseguir sus fines.

1.2 Objetivos de la investigación

El TFM que se quiere desarrollar, versa sobre ***Botnets: La amenaza fantasma***.

Al ser un tema amplio y algo desconocido quiero abordar este trabajo desde una perspectiva del conocimiento y entendimiento de esta amenaza (seria la parte de investigación documental) y desde la perspectiva de un ámbito práctico (seria la parte experimental) realizando un caso de estudio práctico sobre la botnet Zeus.

Este trabajo se basará en documentos, guías, normativas y publicaciones de diferentes fuentes, recogidas y referenciadas para que sirvan de base en el estudio documental inicial.

En la investigación experimental se tratará de vincular los resultados sobre la protección de esta amenaza en las guías y publicaciones existentes, ampliando (en su caso) las recomendaciones necesarias en virtud de los resultados empíricos obtenidos.

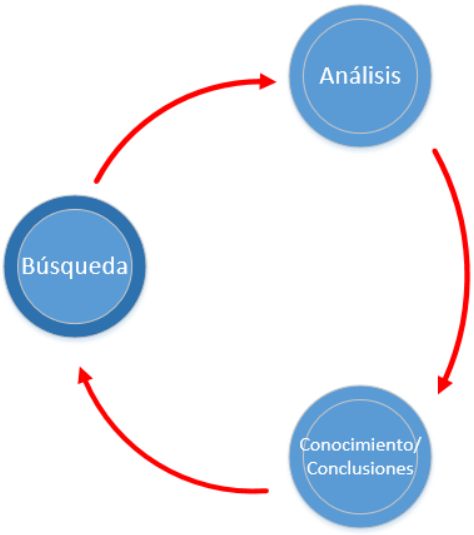


Figure 1.1: Ciclo de metodología documental.

En este trabajo se pretende alcanzar cuatro objetivos principales:

1. Explicar la taxonomía de infección, empleo y gestión de las botnets, así como su impacto en internet.

2. Realizar un ejemplo práctico (a través de la implementación de un escenario) verificando su potencial, forma de ejecución y posibles medios de protección a través de datos empíricos.

3. Obtener unas conclusiones finales en las que se puedan reseñar los tipos de defensas que se podrían establecer ante este tipo de ataques y/o empleos de métodos de detección.

4. Por último, obtener vías de estudio para un trabajo futuro vinculado a los botnets.

Para la conclusión de este TFM se pretende seguir un proceso deductivo a través de una metodología procedimental documental (o bibliográfica) y experimental operativa.

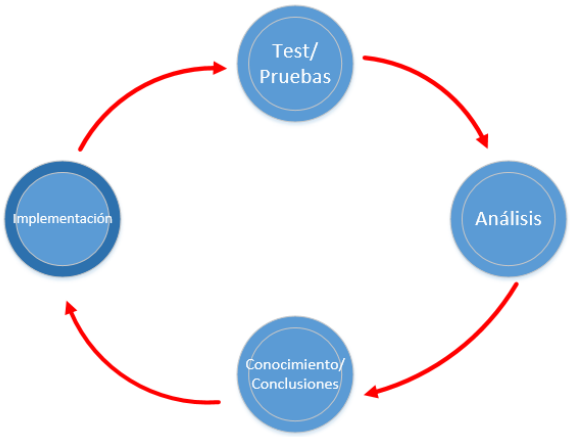


Figure 1.2: Ciclo de metodología experimental.

La metodología documental (búsqueda – > análisis - > conocimiento extraído - > nueva búsqueda) es la que se muestra en la siguiente imagen 1.1 y se realizará a través de la investigación en fuentes sobre el TFM a desarrollar y nos dará una visión inicial del tema tratado, búsqueda de normativa, guías y otra documentación asociada a los botnets. Esta metodología no solo se va a emplear para la fase inicial del TFM, sino durante todo el desarrollo del trabajo como por ejemplo en la selección de las herramientas más adecuadas para la implementación del escenario práctico.

La metodología experimental (implementación – > test/pruebas – > análisis - > conocimiento extraído - > nueva implementación) es la que se muestra en la siguiente imagen 1.2 y se basará en el establecimiento de unas fases o pasos determinados en la parte práctica de este trabajo. Gran parte de esta experimentación será analizada y grabada en un vídeo sobre el caso concreto que se implemente. Ambas metodologías se basan en los dos criterios siguientes:

1. La resolución final de los objetivos propuestos.

2. Las adquisiciones de nuevas competencias en todos los procesos anteriores antes de pasar al siguiente escenario.

1.3 Justificación de dichos objetivos

El establecimiento de los anteriores objetivos se ha basado en la consecución del entendimiento del trabajo planteado de una manera secuencial. Así, se estipula que primero se debe conocer

todos los conceptos relacionados con las botnets y sus tipos antes de poder proponer un escenario práctico.

Una vez asimilados los conocimientos necesarios se seleccionará un escenario práctico en forma de un laboratorio en el que se pueda representar un caso de una botnet en la que se den varios tipos de elementos adquiridos en la fase teórica para poder aplicar un estudio de tipo deductivo que nos aporte una visión práctica y del que podamos extraer unos conocimientos específicos y empíricos de la materia.

Los últimos objetivos se cumplirían en la realización de un análisis de los resultados obtenidos del laboratorio implementado confrontándolos con la teoría conocida y del que poder extraer una nuevas (o confirmar) unas conclusiones. Además de dichas conclusiones, esta fase de análisis final nos permitirá establecer una nueva base teórica o práctica para poder ser ampliada en futuros trabajos de investigación en algún campo más especifico y relacionado.

1.4 Recursos y limitaciones existentes

En este apartado se valorará aspectos materiales de personal y administrativos, así como sus limitaciones. Los recursos empleados para el desarrollo de este trabajo de fin de master (TFM) son los siguientes: Recursos Materiales:

- Ordenador Portátil MacBook Pro (2,7 GHz i7 / 8 Gb RAM)
- Ordenador de sobremesa MSI Cubi (2,4 GHz i7 / 16 Gb RAM)
- Red LAN propia de mi hogar.

Recursos Humanos

- El propio alumno.

Recursos documentales

- Los accesibles a través de internet.
- Los accesibles a través de la biblioteca de la UOC.

Recursos Software

- VMWare Workstation Pro 12
- Software gratuito disponible en UOC o en internet (W7, W10, Visio...etc.)
- WireShark
- XAMPP

- GANTT Project
- ShowMore
- HitmanPro
- Malwarebytes
- Dependency Walker
- Volatility
- Belkasoft
- Blue Botnet (ataques DDoS)
- Máquinas virtuales gratuitas (OpenSource).
- Versión de Zeus 2.1.0.1 http (Existen otras muchas como Zeus P2P -GameOver-, Floki-Bot...etc).

Recursos Temporales:

- Media jornada compartida con 2 asignaturas de la UOC (Previsible 2/3 horas diarias aumentanándolas los fines de semana). Se explicará mas adelante.

A pesar de existir limitaciones temporales y de personal se querría destacar los dos principales obstáculos que se ha encontrado en la realización del laboratorio práctico:

- La disponibilidad un entorno físico con mejores especificaciones técnicas que el empleado.
  - Ya que la simulación con maquinas virtuales (aunque el host anfitrión contaba con 16 Gb) consume muchos recursos y nos hemos limitado inicialmente a 3 maquinas "zombis" (más adelante se empleará una más) para que el laboratorio tuviese la suficiente fluidez.
  - Debido a esta limitación no se ha podido elaborar un escenario mucho mas ambicioso con varios servidores DNS, 8 o 10 bots, una maquina objetivo, 2 C&C servers, un BotMaster y varias redes filtradas a través de un firewall (DMZ, Inside y Outside).
  - La alternativa de emplear un entorno real a través de internet se ha descartado en aras de tener un entorno controlado que pudiese no levantar alarmás externas ni crear trafico de malware visible.
- La dificultad obtener las versiones de Zeus.

- Si bien es relativamente fácil obtener versiones de Zeus antiguas (por debajo de la 1.9), la descarga de versiones superiores se complica y es necesario estar registrado en alguna web para su obtención.
- La diferencia entre versiones es crítica ya que algunas dan fallo en su funcionamiento y ciertos comandos y scripts no funcionan adecuadamente.
- La infinidad de versiones (más o menos especificas) de Zeus es también otro dilema difícil de dilucidar.
- La documentación de este tipo de malware es o limitada o incompleta y difícil (en ocasiones) en ocasiones de entender.
- Por último, reseñar que en la descarga de este tipo de malware nunca sabremos si estamos siendo egañados e infectandos con otro malware oculto, ya que no existen versiones oficiales con un hash y una seguridad que certifique la "limpieza" de la herramienta descargada. Este es otro de los motivos por lo que se creo un escenario "aislado" de la red internet.

Nombre	Fecha de inicio	Fecha de fin
TFM	10/07/17	19/01/18
Propuesta TFM "Ad-hoc"	10/07/17	24/07/17
Estudio del caso (Guión-tutora)	26/09/17	27/09/17
PEC_1	5/10/17	7/10/17
Entendimiento del problema TFM)	5/10/17	7/10/17
PEC_2	9/10/17	18/10/17
Busqueda de Información (Botnets)	9/10/17	15/10/17
Análisis de requisitos escenario práctico	16/10/17	18/10/17
Generación de documentación (TFM)	8/10/17	11/01/18
PEC_3	19/10/17	6/12/17
Diseño del escenario práctico	19/10/17	20/10/17
Preparación del escenario práctico	21/10/17	7/11/17
Implementación del escenario práctico	8/11/17	13/11/17
Ejecución de pruebas y obtención de datos	14/11/17	29/11/17
Obtención de conclusiones	30/11/17	6/12/17
PEC_4 (Memoria Final)	7/12/17	3/01/18
PEC_5 (Presentación/Video)	4/01/18	10/01/18
Preparación defensa del TFM	11/01/18	16/01/18
Defensa final del TFM	17/01/18	21/01/18

Figure 1.3: Calendario del TFM.

La planificación temporal se confeccionará con un diagrama de GANTT 1.4 en el que se muestran las tareas a realizar con su calendario detallado y sus inter-dependencias.

1.5 Desarrollo del trabajo -Diagrama de Gantt-

Las tareas desarrolladas se muestran en la imagen 1.3, donde se pueden ver por orden cronológico. Aunque las tareas se dividen en fases secuenciales, alguna de ellas podrá realizarse durante todo

el tiempo de desarrollo del TFM (por ejemplo, la documentación).

En la imagen 1.4 se puede ver el diagrama de GANTT con diferentes colores siendo los que son iguales los relativos a tareas dentro del mismo bloque de trabajo y representándose en color azul las PECś que se han entregado.

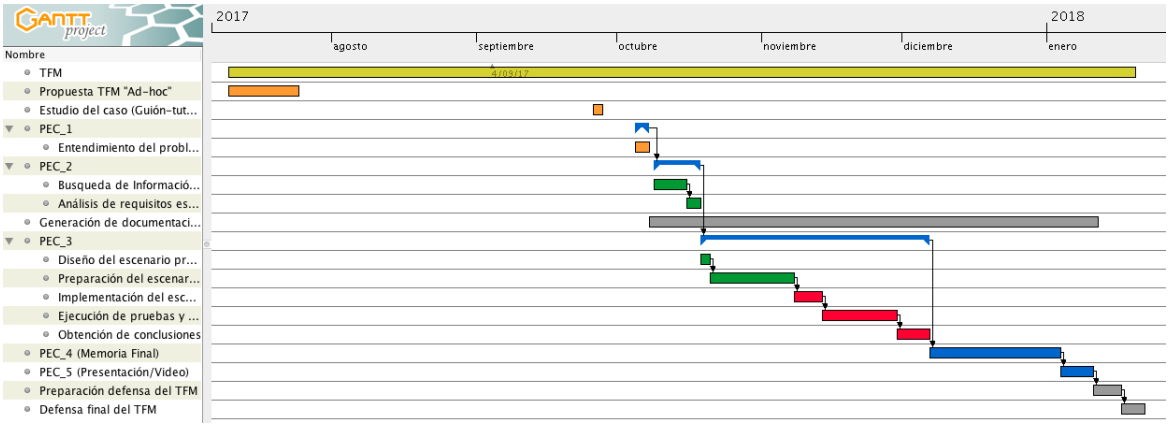


Figure 1.4: Diagrama de GANTT del TFM.

1.5.1 Disposición de tiempo a las tareas

La planificación temporal de las tareas a desarrollar ha sido definida teniendo en cuenta la disposición de tiempo del alumno debido a la compatibilidad con su actividad profesional y al hecho de tener que realizar dos asignaturas más de este mismo Master (MISTIC).

La distribución en cuanto a las horas dedicadas semanalmente se muestra en la tabla 1.1.

En dicha tabla se observa que se emplea más tiempo durante los fines de semana. Así mismo, se quiere indicar que este alumno empleó una semana de vacaciones para poder documentar el proyecto empleando una media de entre 8 a 10 horas diarias durante esos días.

Por último es importante destacar que existen casos en los que los imprevistos han hecho que el alumno empleé más tiempo unos días en detrimento de otros.

	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
Horas dedicadas	2 h	2h	2h	2h	2h	5h	5h

Table 1.1: Distribución de horas semanalmente.

CHAPTER 2

MARCO DE REFERENCIA

Cualquier tecnología lo suficientemente avanzada es indistinguible de la magia. Arthur C. Clarke (1917-2008)

2.1 Fundamentos teóricos

Parecería absurdo cometer un delito sin proteger nuestra identidad o con medios menores a los que la protección de nuestro objetivo ha empleado para protegerse (Piénsese en ir a atracar un banco a cara descubierta y con un simple periódico como arma). Es por ello que los que cometen delitos informáticos utilizan sus recursos para cometer estas acciones siempre intentando obtener **anonimato**, **ventaja tecnológica** y un **beneficio** para lucrarse con ello. *Las Botnets nos proporcionan todas estas ventajas*.

Botnet es una palabra formada por la unión de "BOT" referente a robot y "NET" referente a la red informática. Simplemente describe una red (o si queremos un "ejército") de robots. Aunque el termino "robot" nos lleve a pensar más bien en una idea futurista de máquinas de metal con apariencia humana, aquí se refiere, sin embargo, a artefactos software que se ejecutan en las máquinas víctima tomando el control sobre ellas. Estas máquinas controladas se las suele denominar "zombis". El disponer de un ejército de máquinas zombis a nuestra disposición nos permite realizar ataque coordinados o conjuntos sobre las víctimas elegidas. Con este potencial y con el enmascaramiento que obtiene el que gobierna (o amo) este "ejercito de zombis" permite poder realizar unos ataques exitosos y anónimos.

Podemos concluir que un bot (robot) nos es más que un software o programa informático que, una vez instalado en un ordenador, realizará las tareas para las cuales ha sido programado y con total autonomía respecto al usuario de esa máquina, el cual ni se percatará de ello.

Un zombi (a nivel informático) es una máquina infectada en la que su propio usuario (y las aplicaciones que la protegen, como el antivirus) no tiene "conciencia" de que está infectada y que, desde el punto de vista del usuario externo, funciona con toda normalidad. Además, esta máquina infectada puede ser el objetivo del atacante o un medio (o plataforma) desde donde realizar ataques a otros objetivos. Una de las funciones que se le puede asignar (entre otras) seria la infección de otras máquinas.

De esta manera, un solo atacante (o grupo de ellos) podría disponer de miles de equipos a su servicio desde la que poder lanzar sus ataques coordinados a objetivos más jugosos. La topología clásica de una botnet es la de una arquitectura jerárquica donde ciertas máquinas (equípos informáticos) son utilizados como servidores de mando y control (servidores C&C) para dar órdenes y poder recibir información de sus bots o máquinas zombis. Este esquema de funcionamiento puede verse en la figura 2.1.

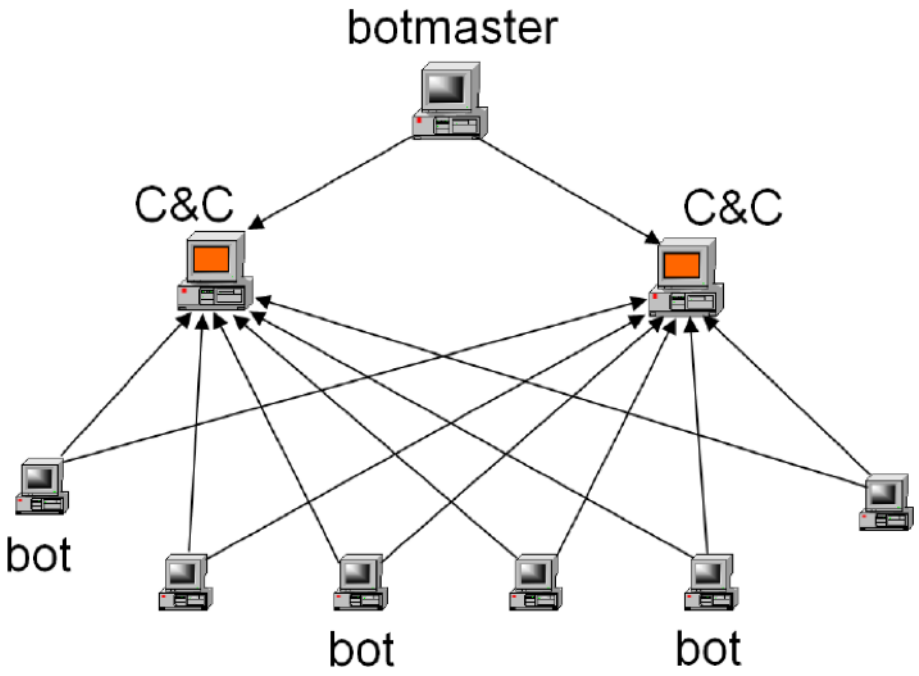


Figure 2.1: Tolopología clasíca botnet [2].

2.2 Antecedentes

2.2.1 Inicios y cambios tecnológicos

Se dice que las botnets nacieron con el primer gusano conocido (Morris's Worm 1988) aunque habría que esperar hasta 1999 para considerar los primeros malwares que podría ser descritos como botnets. Los primeros fueron "Sub7" y "Pretty Park" - un troyano y un gusano respectivamente. Ambos introdujeron el concepto de máquina víctima que se conectaba a un canal de IRC para recibir los comandos maliciosos.

El siguiente hito reseñable [17] en cuanto a este tipo de amenazas se produce en el año 2000 con la aparición del bot de amenaza global (GTbo). GTbot se basó en el cliente mIRC, lo que significaba que podría ejecutar scripts personalizados en respuesta a eventos IRC a través de sockets TCP/UDP, lo que lo hacía idóneo para ataques de denegación de servicio (DoS).

En el año 2002 se vieron dos notables evoluciones en la tecnología botnet con el lanzamiento de SDBot y Agobot. SDBot estaba escrito en C ++ y, debido a que su creador puso a disposición el código fuente, muchos de los bots posteriores incluían partes de ese código en sus desarrollos. En ese mismo año, Agobot introdujo el concepto de ataque modular y escalonado, ya que las cargas útiles (payloads) se entregaban secuencialmente.

El refinamiento de los bots con Agobot alcanzó una nuevo nivel al ser cpaz de instalar una puerta trasera en su víctima para, luego, deshabilitar el software antivirus y bloquear el acceso a los sitios web de los proveedores de seguridad. Ambos robots estaban dirigidos al uso del control remoto y al robo de información, pero el movimiento hacia la modularización y el código abierto comenzó hizo que comenzase el aumento en las variantes y la expansión de sus funcionalidades.

Otro de los hitos importantes fue el lanzamiento de Spybot en 2003 que era una evolución del anterior SDbot, pero introduciendo algunas nuevas funcionalidades importantes como el keylogging, data mining, SPIM (Instant Messaging Spam). En ese mismo año también vimos la aparción de Rbot que introdujo el proxy de SOCKS, e incluyó funcionalidades de DDoS y herramientas de robo de información. Rbot fue también la primera familia de robots en utilizar algoritmos de compresión y encriptación para tratar de evadir su detección.

En 2003, también, se vio la primera aparición de una botnet "peer-to-peer" con el nombre de Sinit. Más tarde se desarrollaron módulos Agobot para incorporar esta funcionalidad peer-to-peer.

Al año siguiente, otro derivado de Agobot, conocido como Polybot introdujo el polimorfismo para tratar de evadir la detección cambiando su aspecto.

Las siguientes evoluciones fueron encaminadas a ser más efectivas en obtener acceso a través de firewalls que, normalmente, bloquean los puertos de IRC pero si habilitan los de HTTP, ICMP o SSL.

2.2.2 El crimen organizado

Alrededor del año 2003 cuando surgió un gran interes sobre las posibilidades ofrecidas por botnets. Al principio de la década, el correo spam todavía era en gran parte una de las grandes tareas de los bots. Bagle y Bobax fueron los primeros botnets de spam y el malware Mytob fue esencialmente una mezcla de un gusano de correo masivo. Los delincuentes evolucionaron las botnets para distribuir sus actividades de spam en todas sus máquinas víctima, dándoles agilidad, flexibilidad y ayudándoles a evitar la actividad de cumplimiento legal sobre el spam que ya se aplicaba.

En 2006 aparece RuStock como una botnet de spam y Zeus (2007) que se consideran como una herramienta de robo de información. Desde ese año, Zeus probablemente se ha convertido en la herramienta criminal más utilizada en robar información. Los creadores de Zeus lo han actualizado regularmente lanzando nuevas versiones del rootkit y añadiendo o mejorando su funcionalidad.

Como estas nuevas versiones se han puesto a la venta a precios muy elevados, las versiones anteriores se distribuyeron gratuitamente. A menudo, estas versiones más antiguas son infectadas por los criminales, lo que significa que el "ladrón novato" también se convierte en la víctima de esa botnet. Este exceso de herramientas criminales de libre disposición ha reducido la barrera de los costos de la ciberdelincuencia y alentó a más criminales aspirantes a la delincuencia en línea. Zeus ha sido diseñado para realizar una infección efectiva en internet.

Durante esta época ya se vieron casos de botnets enormes. Por ejemplo, la Fundación Shadowserver [6] ha detectado más 6000 servidores C&C y esa cifra podría incluso ser baja en comparación a la que se considera real. Trend Micro está siguiendo a decenas de millones de PCs infectados que se están utilizando para enviar Spam y esa cifra no incluye todos las demás PC's infectados con bots que están siendo utilizados con fines de robo de información, DDoS o cualquiera de los otros crímenes.

2.2.3 Alarma social

Es a partir del 2003 cuando la comunidad científica se preocupa gravemente por esta amenaza y se empiezan a desarrollar métodos de ataque, defensa y detección contra las botnets.

Desde entonces ha habido muchas anulaciones, desmantelaciones o deshabilitaciones exitosas y principalmente dirigidas a proveedores de servicios criminales (normalmente webhostings) que albergaban gran parte de la infraestructura de mando y control.

Tras múltiples quejas Atrivo/Intercage [4] (WebHosting) en 2008 corto su tráfico debido a que muchos cibercriminales empleaban su hosting para el spaming. Este hecho casi destruyó la botnet Mega-D, pero en poco tiempo esta botnet encontró nuevos "alojamientos" (hosting) desde donde operar.

Otro ejemplo fue McColo otro ISP que alojaba hosting de algunos servicios cibercriminales (alojaba sus servidores C&C) para las botnets Srizbi, el Mega-D "revivido", RuStock, Asprox,

Bobax, Gheg y Cutwail. Cuando McColo se retiró de Internet a finales de 2008, se produjo una caída global en los niveles de spam de casi el 80% como se puede ver en el gráfico adjunto 2.2.

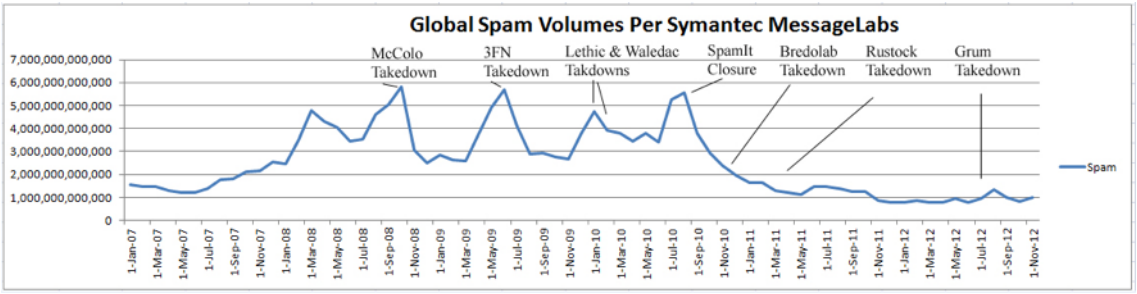


Figure 2.2: Variación volumen de Spam [34].

En dicho grafico también se pueden observar otros casos como los de 3FN (2009), Lethic&Waledac (2010), SpamIt (2010), Rustock (2011)...etc. No obstante, la amplia cantidad de servicios hosting y/o ISPs hace difícil acabar con este tipo de amenazas, habida cuenta que hoy en día diversifican sus servidores C&C por diferentes proveedores de servicio.

No obstante, la acción coordinada que las organizaciones públicas y privadas están tomando contra las botnets hace que la innovación criminal tenga que buscar nuevas vías de "subsistencia" y que las redes poco diversificadas sean rápidamente anuladas o desmanteladas. A medida que surgen las nuevas tecnologías, los delincuentes buscan maneras de adaptarse a ellas con el fin de aumentar su escalabilidad y flexibilidad o para proporcionar un camuflaje más eficaz.

Una de estas adaptaciones se produjo cuando se pasó de la definición de las direcciones IP (o nombre de dominio) de los servidores de C&C en cada bot (haciendo fácil percibir ese tráfico e interrumpirlo) a que cada robot fuera capáz de generar de forma criptográfica nombres de host alternativos para sus servidores de mando y control. Los BotMaster, obviamente, saben qué nombres de host serán generados en un día determinado, y simplemente necesitan añadir ese canal de comando alternativo en la operación.

Esto último fue el caso de Conficker que era capaz de generar 50.000 nombres alternativos cada día! (2008). La industria de la seguridad tuvo que tratar de bloquear el acceso a todos ellos mientras que los criminales sólo tenían que hacerlo bien una vez. Vale la pena recordar que cerca de seis millones de máquinas seguían siendo infectadas por Conficker casi dos años después que apareciera por primera vez.

Además del spam, la denegación de servicio, el robo de información, el chantaje y la extorsión, las botnets también han evolucionado hasta convertirse en redes de distribución de software altamente eficientes para uso criminal. Los criminales pagan por el acceso a máquinas comprometidas para enviar más malware a estos equipos ya infectados. Los bots de spaming puede, además, entregar información de malware por ejemplo. Otras vías de infección usadas son los softwares antivirus falsos.

El hecho, es que muchos cibercriminales hacen su dinero simplemente por el alquiler de acceso a sus botnets en lugar de participar en spam, DDoS o campañas de robo de información.

2.3 Situación actual de la amenaza

2.3.1 Situación Actual

Como una imagen vale más que mil palabras, se presentan la imagen 2.3, obtenida de Trend Micro, que nos muestra la actividad y la situación de las botnets a nivel mundial durante los últimos 14 días (tomada el 14/10/17) relacionadas con las botnets en el mundo divididas en "C&C server" y "Target Computers" cuyos roles se explicaran más adelante en este documento. Se pueden observar que existen más de 9700 C&C servers "localizados" (estimamos que se puede localizar solo un porcentaje menor de ellos) y más de 3.600.000 de Target Computers.

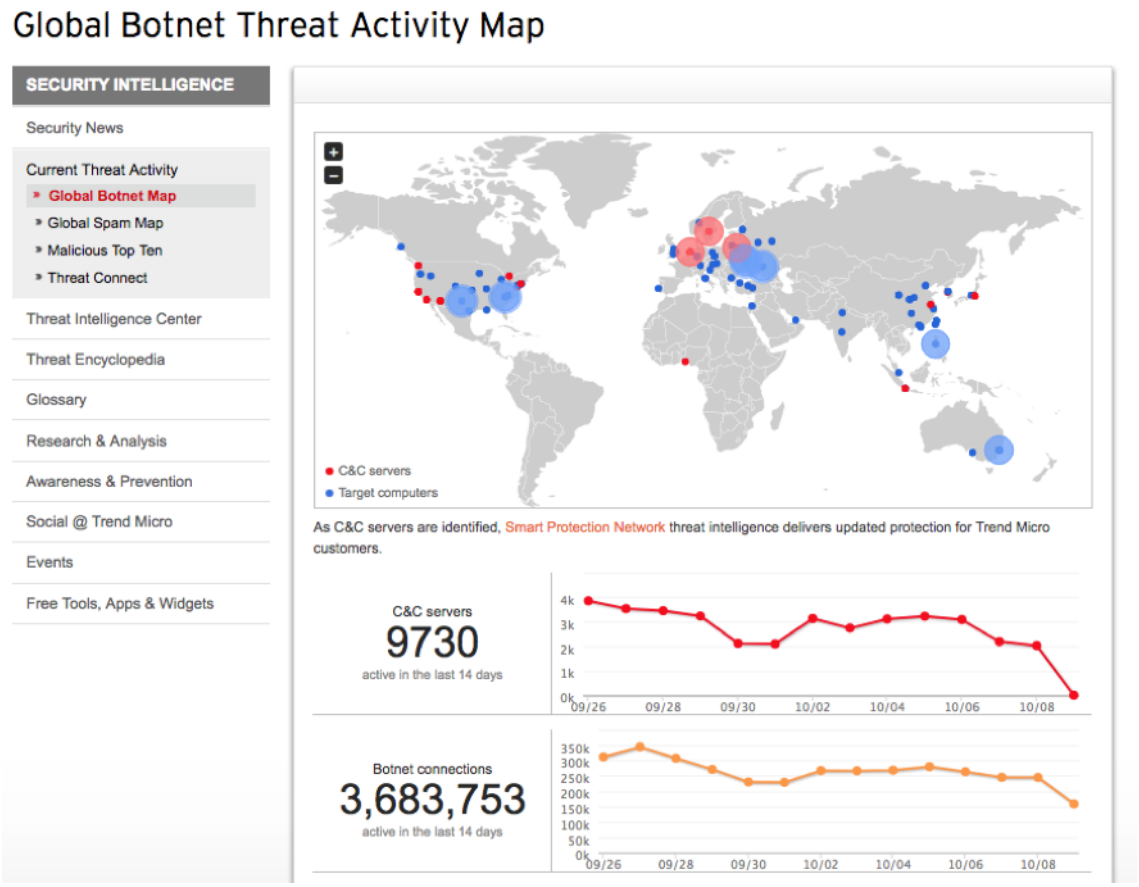


Figure 2.3: Mapa actividad actual botnets [16].

Hay que tener en cuenta, además, que se estima que solo se pueden detectar una minoría de ellos y que algunos no se muestran al no encontrarse en actividad durante el escaneo de los

últimos días. Estas cifras son preocupantes ya que aunque detectados siguen con su actividad y no resulta sencillo poder inhabilitarlos.

Por otro lado, el llamado crimeware (Tipo de software que ha sido específicamente diseñado para la ejecución de delitos financieros) preocupa mucho al ámbito económico y está muy relacionado con las botnets. En la siguiente imagen 2.4 observamos que la mayoría de los hechos crimenware más peligrosos están relacionados con las botnets.

Tanto es así que una empresa como Heimdal Security en un artículo suyo (año 2016) establece que de los 10 malwares más peligrosos comocCrimeware los 9 primeros son botnets relacionadas con Zeus y la décima es un ransomware. La lista es la mostrada a continuación:

1. Zbot/Zeus
2. Zeus Gameover (P2P) (Zeus family)
3. SpyEye (Zeus family)
4. Ice IX (Zeus family)
5. Citadel (Zeus family)
6. Carberp (Zeus family)
7. Bugat (Zeus family)
8. Shylock (Zeus family)
9. Torpig (Zeus family)
10. CryptoLocker

Top Ten Crimeware

Top ten banking and related crimeware as of Q2 2011

Rank	Crimeware Detection Name
1	MAL_BANKER
2	BKDR_QAKBOT.SMG
3	BKDR_PAPRAS.SME
4	TROJ_SPYEYE.SMEP
5	MAL_BANKER2
6	MAL_BANKER11
7	WORM_QAKBOT.QRZ
8	BKDR_QAKBOT.SMC
9	TSPY_BANKER.ES
10	WORM_QAKBOT.BS

Figure 2.4: Top Ten Crimeware [24].

En la imagen 2.5 de Julio 2017, podemos ver los bots que existen en Europa y como España y Madrid están entre los países y ciudades con mayor número de bots de Europa.

España ocupa la quinta posición en el EMEA (Europa, Oriente Medio y África) y la décima a nivel mundial con el mayor número de bots en el año 2017. De hecho, Madrid aglutina más del 72% de los bots de todo el país colocándose, así, como la primera ciudad con mayor número de bots en la región. En España hay, al menos, un bot por cada 30 usuarios de Internet, clasificando el país 40° en Europa por la densidad de bots.

Otro de los motivos por el que la amenaza de las botnets es tan crítica es la gran cantidad de dispositivos conectados a internet como son los teléfonos móviles (smartphones) y el internet de las cosas (IoT -red de objetos cotidianos interconectados-) que, unido al mayor ancho de banda del que disponen cualquiera de los dispositivos hoy en día, hace que estos sean futuros bots para una botnet. Además,la mayor interconexión mundial de los usuarios (mayor en los países desarrollados) facilita la labor de contagio y de la creación de redes botnets.

Por último, reseñar que en está siguiente imagen 2.6 se observa que el 21,2% de la comunicación malware (septiembre 2017) es relativa a comunicación entre bots.

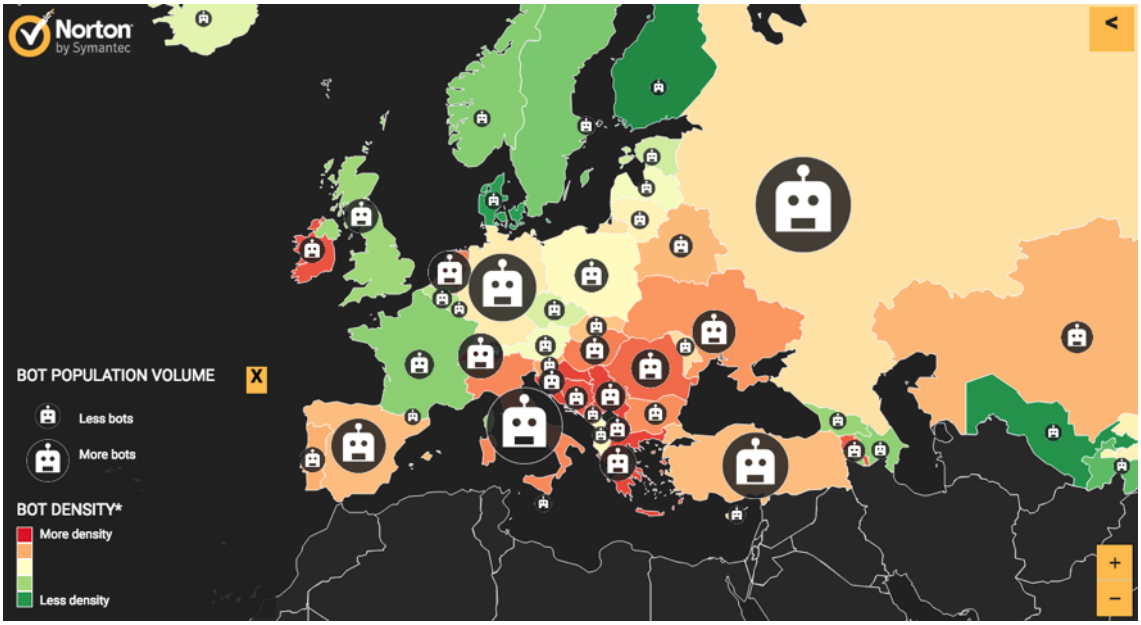


Figure 2.5: Distribución de bots en Europa Jul2017 [25].

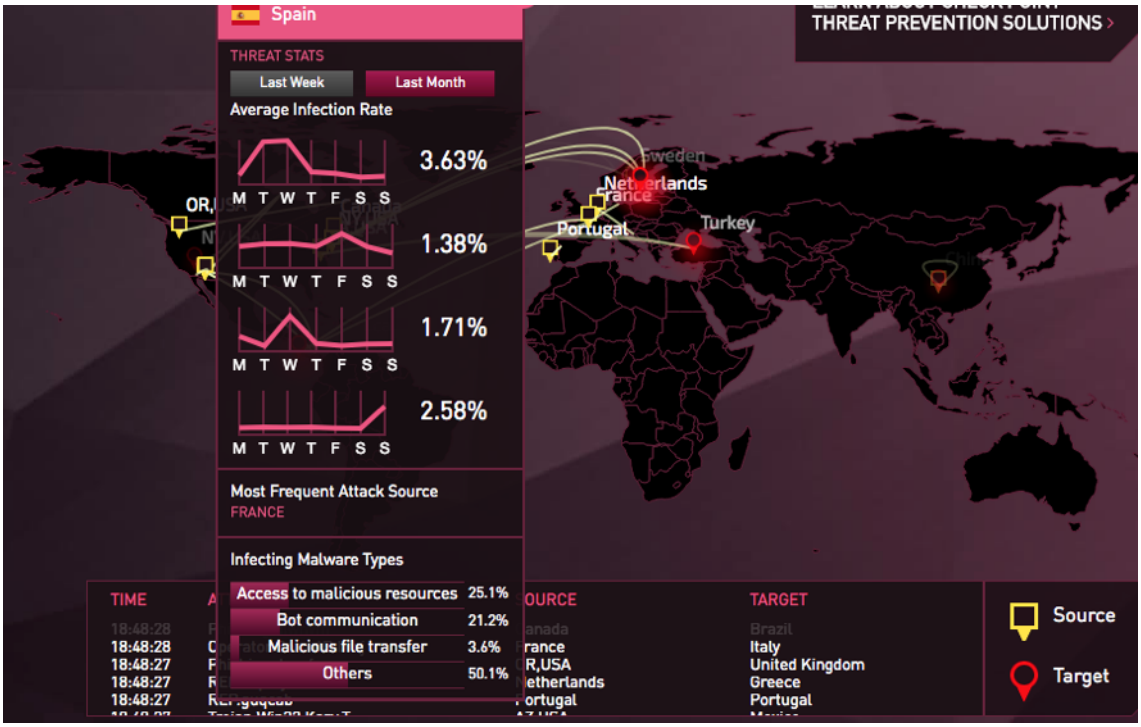


Figure 2.6: Tráfico de comunicación entre bots Sept2017 [23].

2.3.2 Nuevas vías

Desde la segunda mitad de 2007, los delincuentes han estado usando la web 2.0 (redes sociales). Los primeros canales "no convencionales" de C&C identificados fueron los blogs y feeds RSS, donde los comandos eran enviados a un blog público por el cibercriminal y los bots recuperaban esos comandos a través de un feed RSS. Del mismo modo, el envío de información de las máquinas infectadas se publicaba en un blog público completamente separado y legítimo para su posterior recuperación por el servidor de comando y control, de nuevo por RSS.

A medida que los servicios de la web 2.0 se han extendido y han ganado un gran nivel de aceptación dentro de las empresas, sus usos criminales ha evolucionado rápidamente. Uno de estos ejemplos es la detección de servidores comprometidos en la nube EC2 de Amazon, que ha sido utilizados para alojar archivos de configuración para el bot Zeus. También twitter se ha utilizado como URL de destino en campañas de spam, para intentar superar el filtrado URL de los mensajes de correo electrónico. Otros como Facebook, Pastaban, Google Groups y Google AppEngine han sido utilizados como infraestructuras sustitutivas de servidores C&C.

Estos foros públicos han sido configurados para emitir comandos "ofuscados" a botnets distribuidas a nivel mundial, estos comandos contienen URLs adicionales a las que el bot accede para descargar nuevas instrucciones o componentes necesarios. La gran ventaja del uso cibercriminal de estos sitios y servicios radica en el hecho de que ofrecen un medio público, abierto, escalable, altamente accesible y relativamente anónimo para mantener una infraestructura de mando y control que, al mismo tiempo, reduce aún más las posibilidades de detección por las tecnologías tradicionales.

Así pues se ha evolucionado desde los canales IRC, que hoy en día se consideran "débiles" a nivel de botnet, a establecer comunicaciones http o https a un proveedor de contenido reconocido y legal como Facebook, Google o Twitter. El objetivo final es mezclar su "ruido" (huella de tráfico) en las comunicaciones con el normal o habitual de las redes lícitas hoy en día para que no solo sea difícil su disrupción y localización , sino incluso la detección de su presencia.

En la incesante búsqueda de nuevas vías de comunicación y ocultación de un tráfico "normal" se emplean técnicas de P2P o hibridas P2P-http...etc. También se avanza en las comunicaciones entre los bots y el servidor C&C para que se cifren más eficazmente mediante la adopción de la PKI u otro tipo de cifrado asimétrico. El empleo de hosting y tecnologías emergentes como servicios en la nube ayuda a toda esta ofuscación.

2.3.3 Nuevos rehenes (IoT)

La internet de las cosas (IoT) está compuesta no solo de computadores dedicados, sino también de monitores de implantes cardíacos, equipos electrodomésticos e industriales, automóviles, sensores mecánicos y otros dispositivos con direcciones IP y la capacidad de transmitir datos a través de una red. En el contexto de IoT, estos son conocidos como "cosas".

A finales de diciembre de 2013, un investigador de Proofpoint [30] (una compañía de seguridad corporativa con sede en California) notó que cientos de miles de correos electrónicos maliciosos registrados a través de un gateway de seguridad provenían de una botnet que incluía no solo computadoras, sino también otros dispositivos, incluyendo televisores inteligentes, frigoríficos u otros electrodomésticos.

David Knight, de Proofpoint, acuñó la palabra "thingbot" para referirse a dispositivos que no sean computadoras que han sido captados por una botnet.

A partir de este hecho y del incesante aumento de los dispositivos IoT la comunidad científica se empezó a preocupar de este nuevo tipo de "rehenes" y las consecuencias que ellos podrían producir. Sin embargo, fue en 2016 con la botnet Mirai cuando se encendieron todas las alarmas. En octubre de ese año, miles de dispositivos (IoTs) infectados realizaron un ataque de denegación de servicio contra los servidores de Dyn, una importante empresa de DNS. Páginas como Twitter, Spotify, PayPal y Amazon quedaron inaccesibles durante horas.

Según datos de la firma de seguridad BitSight Technologies, cerca de 14.500 dominios que usaban los servicios de Dyn abandonaron la compañía inmediatamente después del ataque. La brutal desbandada supuso la pérdida del 8% de los dominios que dependían de Dyn para la gestión de sus DNS, clientes que se vieron afectados por el ataque de la botnet Mirai.

Cabe destacar que Mirai fue publicado bajo una licencia de software libre por lo que cualquier persona puede obtenerla y usarla para dirigir ataques coordinados y masivos de denegación de servicio. Mirai es capaz de afectar a impresoras, cámaras de vigilancia o routers caseros.

Mirai es efectivo porque cientos de miles, probablemente millones de dispositivos del internet de las cosas funcionan con el usuario y contraseña asignado de forma predeterminada, en muchos casos tan simples como usuario/admin, contraseña admin o contraseñas del tipo 123, 12345 o 1111. El malware se aprovecha de esto, accede y los infecta. Con poder computacional suficiente, es posible hacer un escaneo de cientos de miles de estos dispositivos e infectarlos en pocos minutos para dirigir un ataque coordinado, como el ocurrido el 21 de octubre de 2016.

Un hecho curioso es la aparición de Hajime, una botnet que infecta al IoT pero en este caso sus intenciones son "buenas", bloquea los puertos para impedir el acceso a botnet como Mirai. El creador deja un mensaje en el dispositivo que traducido dice "Sólo un sombrero blanco asegurando algunos sistemas. Estar atento!"

Este gusano, afortunadamente, no está construido con capacidades maliciosas. Pero el miedo que existe es que su desarrollador algún día elija modificar el gusano, lanzar ataques DDoS o participar en otras formas de delito cibernético. Hajime también contiene una característica que hace que sea difícil de detener, el gusano se comunica a través de una red Peer-to-Peer. Esto significa que una gran cantidad de dispositivos infectados con Hajime se pueden utilizar para transmitir archivos o instrucciones al resto del grupo.

Con más de 20.000 millones de "cosas" conectadas a nivel mundial actualmente y una estimación de 200.000 millones en 2020, la seguridad se ha vuelto más importante. Los ataques

a dispositivos IoT aumentaron un 43%, el uso de este tipo de dispositivos en los ataques DDoS están dejando obsoletas las defensas tradicionales.

2.3.4 ¿Sin esperanza?

Ante este panorama los gobiernos y las organizaciones internacionales como la UE, la OCDE y la ONU están prestando una mayor atención a la armonización del derecho penal a nivel mundial en el ámbito de la ciberdelincuencia, lo que permite un procesamiento más eficaz. Los organismos encargados de hacer cumplir la ley buscan formalizar acuerdos multilaterales para hacer frente a un crimen verdaderamente transnacional.

También los proveedores de servicios de internet y los registradores de dominios tienen un papel clave que desempeñar. Los proveedores de servicios de internet deben informar y ayudar a los clientes que creen que están infectados. Los registradores de dominios deben exigir formas más eficaces de identificación rastreable en el momento de la inscripción y los que transgreden las normativas se les debe de suspender el servicio tan pronto como se planteen sospechas creíbles.

La industria de la seguridad ya está extrayendo lecciones valiosas de los niveles de cooperación alcanzados durante la lucha contra las grandes oleadas de botnets (sobre todo de spam). Existen iniciativas impulsadas desde los CERTs (CSIRTs) para educar e informar más eficazmente a los ciudadanos sobre los peligros que representa el cibercrimen y para fomentar prácticas informáticas más seguras.

Así mismo, la amplia preocupación en la ciberseguridad y la ciberdefensa hace que haya surgido una inquietud en este sentido y no solo en el ámbito empresarial o estatal sino en el académico, desarrollándose nuevos estudios, cursos y especialidades tendentes a desarrollar técnicos expertos en seguridad ante unas amenazas tan palpables.

2.4 Casos reales más relevantes

Existen multitud de casos reales y conocidos en el empleo de botnets para el cibercrimen. Se podría realizar una clasificación de los más grandes, los más destructivos o los más conocidos, sin embargo la extensión del trabajo nos impide tratar de todas estas clasificaciones. A continuación se han seleccionado los casos reales que han podido tener más “impacto” social teniendo en cuenta todos los parámetros (tamaño, nivel de destrucción y popularidad) [37].

Es importante remarcar que existen multitud de variantes o “releases” de una misma botnet, ya que los desarrolladores los van mejorando para adaptarse a las medidas de seguridad para variar el tipo de ataques o para obtener nuevos medios de propagación.

STORM

- Año de descubrimiento: 2007
- Año de desaparición: finales 2008

- Reaparición: 2010
- Origen: Desconocido, se sospecha que cibercriminales rusos ya que se encontraron palabras rusas en el código fuente.
- Número estimado de máquinas infectadas: Mas de 1.000.000, existen estimaciones de hasta 50.000.000.
- Tipo de Botnet: Múltiples ataques, incluyendo acceso a datos secretos, retrasmisiones SMTP, recolección de direcciones email, spam y DDoS.
- Estado actual: Desarticulado.
- Características: Storm fue la botnet más grande y con mayor propagación hasta la fecha. Tenía el valor añadido de estar disponible para la venta o su alquiler. Empleada habitualmente por su capacidad de DDoS. La ingeniería social y el spam ayudaron a su propagación, pero sus atacantes también lo lanzaron a través de las descargas en los sitios web populares comprometidos, haciendo de las descargas un importante factor de infección.

CONFICKER

- Año de descubrimiento: 2008
- Origen: Puede ser alemán, por su nombre, o ucraniano, por su servidor primario.
- Número estimado de máquinas infectadas: Entre 9.000.000 y 15.000.000.
- Tipo de Botnet: DoS y spammer
- Estado actual: Prácticamente desarticulado, pero siguen existiendo máquinas infectadas.
- Características: Conficker es el único que descarga actualizaciones por sí solo utilizando binarios firmados y encriptados para ayudar a evitar cualquier acción de desinfección. Una vez que la versión está activa, se descarga e instala el malware “Waledac” para enviar spam. Una de sus variaciones fue SpyProtect (2009) con la que trataba de convencer a las potenciales víctimas para comprar un falso programa antivirus cuando en realidad lo que hace es iniciar una serie ataques DoS a la red local a través inundaciones ARP, bloqueos de cuentas, deshabilitando actualizaciones automáticas y actualizaciones de antivirus.

MARIPOSA

- Año de descubrimiento: 2008
- Año de desaparición: 2010
- Origen: España

- Número estimado de máquinas infectadas: Más de 12.000.000.
- Tipo de Botnet: Cyber-estafa y DDoS.
- Estado actual: Desarticulada gracias a los esfuerzos de las fuerzas de seguridad españolas (Guardia Civil), Defense Intelligence, Georgia Tech y Panda Security.
- Características: Mariposa es un keylogger, software que monitoriza y graba en un registro la actividad de teclado del usuario para capturar credenciales en sitios bancarios, de credenciales con la que realizar envío masivo de spam y tomar el control del equipo para su utilización en ataques DDoS. La red Mariposa estaba disponible para ser alquilada. El 3 de febrero de 2010 [31], la Guardia Civil procedió a la detención de Netkairo líder de la banda DDP Team (Días de Pesadilla Team) que usaba MARIPOSA para detener más tarde a su creador, el hacker apodado "Iserdo", de 23 años de edad y autor confirmado del kit de botnet Mariposa.

ZEROACCESS

- Año de descubrimiento: 2011
- Origen: Desconocido
- Número estimado de máquinas infectadas: 9.000.000.
- Tipo de Botnet: Extracción de bitcoins y ciberestafa.
- Estado actual: Desarticulado, pero con posibilidades de reactivarse. Microsoft y las autoridades norteamericanas intentaron acabar con ZeroAccess apoderándose de las consolas de mando y control, pero se perdió una parte y estos elementos peer-to-peer podrían lanzar nuevos ataques en el futuro. ZeroAccess puede comprometer un sistema infectando tanto el MBR (Master Boot Record) del disco duro como drivers críticos y es capaz de deshabilitar tanto el firewall de Windows como software de antivirus.

GAMEOVER ZEUS

- Año de descubrimiento: 2012
- Año de desaparición: 2014
- Origen: Rusia
- Número estimado de máquinas infectadas: 500.000 a 1.000.000.
- Tipo de Botnet: Ciberestafa y extorsión a través de la propagación del malware Cryplocker. Variante de Zeus en P2P.

- Estado actual: Desarticulado, gracias a las agencias policiales y organizaciones líderes del sector tecnológico incluyendo Microsoft, Symantec y McAffee.
- Características: Gameover Zeus usó comunicaciones para establecer una red de zombis con arquitectura peer-to-peer para hacerlo mucho más resistente y difícil de desarticular. Afectaba equipos desde Windows XP a Windows 8 con el principal objetivo del robo de los datos de usuario y contraseñas de acceso a las entidades bancarias. Este botnet nos enseñó la lección que una defensa por capas en las redes corporativas es crítica.

MIRAI

- Año de descubrimiento: 2016
- Origen: Desconocido
- Número estimado de máquinas infectadas: Desconocido, posiblemente millones.
- Tipo de Botnet: Mirai es una botnet cuyo objetivo son dispositivos del llamado Internet de la Cosas (en inglés, Internet of Things, abreviado IoT). Los principales objetivos de este malware han sido los routers, grabadoras digitales de vídeo y cámaras IP de vigilancia. Usado principalmente para ataques DDoS.
- Estado actual: En activo.
- Características: Este malware ataca a dispositivos con sistemas embebidos Linux. El principal método de infección de Mirai es mediante el uso de credenciales por defecto que el malware, incluye ya que muchas de las cuales son usadas en dispositivos IoT donde la seguridad en muchos casos en deficiente. Ha evolucionado y es capaz de infectar máquinas Windows. El 21 de octubre de 2016, la compañía de servicios de nombres de dominio Dyn fue atacada con un masivo y complejo ataque distribuido de denegación de servicio. En su pico, Dyn fue inundado por 1,2 Tbps de tráfico, el mayor volumen de tráfico DDoS jamás grabado. El análisis del ataque confirmó que el tráfico DDoS se originó de los dispositivos de Internet de Cosas infectados por la botnet Mirai.

2.5 Actores y servicios que ofrecen

Ya se comentó que detrás de la creación y el uso de las botnets nos encontramos desde grupos delictivos, bandas organizadas, simples arrendatarios de estas herramientas o incluso gobiernos. Existen varios países y/o organizaciones donde parece existir más relación con esta amenaza.

Las tres zonas geográficas donde parece existir mayor actividad en el empleo y creación de botnets son Brasil-Mejico, Europa del este (Principalmente Rusia) y Asia (Principalmente China y Corea del Sur). Los actores implicados con las botnets son los siguientes:

Creador del código:

Nos referimos aquí al que genera (o reutiliza) un código con el que puede generar un servidor C&C y un ejecutable bot. Inicialmente no son expertos en su creación, sino que el perfil de "tipo" del que se inicia con esta creación suelen ser personas jóvenes, aficionado a los videojuegos, a internet y a las redes sociales, y que aprenden hacking como un reto personal. Su motivación no es financiera, si bien en algunos casos, como hemos visto antes, acaba lucrándose económicamente.

Existe posteriormente la creación de grupos para estos fines a través de foros y con el uso de repositorios de malware que se ofrecen en la red. Después de obtener la experiencia necesaria pueden verse atraídos por la obtención de un gran rédito económico con ello. Ejemplo: Con tan sólo 16 años, un hacker obtuvo información de más de 160.000 clientes†de TalkTalk en 2015, números de cuenta y códigos secretos.

Tampoco hay que descartar el perfil de creadores expertos con encargos "ad-hoc" o incluso grupos ataques subvencionados por gobiernos u organismos e incluso propios trabajadores de gobiernos (como existe departamentos de cibre-ataque en China y Rusia).

BotMaster:

No tiene por qué ser necesariamente la persona o personas que han creado el código de la botnet. En este perfil entra desde los que la han arrendado, los que han solicitado su diseño e implementación o grupos criminales que las adquieren. Lo que si está claro que el BotMaster tiene que tener conocimientos técnicos debido al hecho de que es necesario saber como emplear el recurso de la botnet, como poder mandarle los comandos o como poder ocultar su rastro para quedar siempre protegido.

Cibercriminal:

Puedo o no coincidir con el BotMaster, ya que un cibercriminal podría sencillamente requerir unos datos o un ataque determinado a un BotMaster sin saber ni siquiera los elementos que este emplea con el fin de obtener beneficio con el ataque. Este perfil va desde particulares, empresas, organizaciones, países..etc.

Víctimas:

En este perfil no solo englobamos a las máquinas zombis donde se encuentran los bots, sino también a aquellas donde existan los servidores C&C e incluso los proveedores de servicios como ISPs, WebHostings, redes sociales...etc. Hay que tener en cuenta que todos son víctimas, tanto los usuarios que alojan un bot en su máquina, y que podría (o no) ser empleada para obtener información de el mismo, como los objetivos de ataques más coordinados como spaming o DDoS normalmente enfocados a servidores. Aquí mismo podemos incluir todos los afectados por los efectos del ataque (disrupción de un servicio, robo de contraseñas...etc).

Los servicios principalmente ofrecidos a través de una botnet (existen cualquier otro imaginable) son los mostrados en la siguiente tabla 2.1.

Crimen	Utilización de bots y botnets
Denegación de servicio (DDoS)	El uso de equipos zombis con la intención de bloquear sitios Web de la red, hacerlos inservibles para sus usuarios y, a menudo, imposibilitar el comercio electrónico.
Extorsión	A través de la infección de los equipos zombis con un malware de cifrado o de la amenaza de un ataque a una determinada empresa antes de producirse y solicitarle dinero por ello.
Robo de identidad	En ocasiones el propio equipo infectado desempeña el papel de víctima y de ataque, ya que no sólo ha sido infectado para poder ser empleado en otros atáques sino que también se les roban la información personal de la víctima y la envían al autor del delito.
Spam	Los bots recopilan las direcciones de correo electrónico de sus víctimas infectadas para que se conviertan en nuevos objetivos de spam. Además, el envío de spam a través de las botnets resulta especialmente rentable, ya que los emisores de spam pueden enviar mensajes desde distintos equipos (todos los equipos infectados de la botnet) en vez de hacerlo desde un solo equipo, lo que dificulta sobremanera su detección.
Fraude ("phishing")	Por lo general, los phishers o ladrones de identidad cuentan entre sus herramientas con un ejército de bots. De igual modo que los emisores de spam, los phishers utilizan los bots para identificar posibles víctimas y enviar correos electrónicos fraudulentos que parecen provenir de organizaciones legítimas, como el banco del usuario. Los phishers también emplean los bots para albergar sitios Web falsos, que se utilizan para robar información personal de los usuarios y sirven de puntos de recogida (servidores “dead drop” o “egg drop”) para los datos robados.
Click Fraud	Las botnets se puede utilizar para participar en Click Fraud, donde el bot software se utiliza para visitar las páginas web y automáticamente “haga clic” en banners publicitarios. Los BotMaster usan este mecanismo para robar sumas de dinero de empresas de publicidad en línea que pagan por cada visíta a la página. Con una botnet de miles de drones, el beneficio es enorme. Puesto que la información vienen de máquinas distintas parece legítimo.
Keylogging	El keylogging es quizás la función de botnet más amenazante para la privacidad de un individuo. Muchos bots escuchan la actividad del teclado informan de las pulsaciones de teclas. Algunos bots tienen activadores incorporados para buscar visítas a sitios web particulares en los que se ingresan contraseñas o información de cuenta bancaria. Esto le da al BotMaster la capacidad de acceder a información personal y cuentas de usuarios.
Otros	Aquí podrían ir desde manipulación de encuestas, noticias, descargas de programas u otros malware, ataques por fuerza bruta,etc.

Table 2.1: Servícios que puede ofrecer una botnet.

CHAPTER 3

TAXONOMÍA SOBRE BOTNETS, ATAQUES Y DEFENSAS

Hasta la fecha, no se ha diseñado un ordenador que sea consciente de lo que está haciendo; pero, la mayor parte del tiempo, nosotros tampoco lo somos.

En este capítulo se explicarán los pasos en la creación e infección de una botnet, la identificación de topologías por sus formas de comunicación, los mecanismos de ocultación de las actividades de una botnet así como los medios de protección, detección y ataque para destruir o inutilizar una botnet.

3.1 Creación y Formas de infección

3.1.1 Creación

Normalmente el código que se emplea para crear los artefactos de control y contaminación de una botnet utilizan lenguajes Orientados a Objetos (POO) ya que resultan mucho más cómodos de utilizar.

El empleo de técnicas de reutilización de códigos de otras botnets (en desuso o actuales) o de empleo de técnicas de coordinación entre artefactos troyanos ya existentes son también habituales en su desarrollo.

Por otro lado, el empleo de modularidad en su diseño y por tanto, el empleado en la descarga e instalación de un bot nos permite crear unos bots iniciales muy reducidos que luego irán descargando las herramientas (o módulos de código) asociadas que necesiten. Ello hace que, además, puedan pasar más inadvertidos ante aplicaciones o herramientas de seguridad.

Es necesario tener en cuenta que un bot, troyano o un RAT (Remote Access Tool) a veces puedan parecer sinónimos, pero no lo son. En este ámbito podríamos describirlos como:

- Remote Access Tool: Herramienta software que crea un medio de enlace a través de un puerto para ponerse al servicio o recibir órdenes remotamente. No todos los RATs son malware, existen herramientas de acceso remoto para, por ejemplo, administrar servicios o máquinas.

- Troyano: Elemento software que tiene una finalidad distinta a la que el usuario cree y por ello (a veces) la instala conscientemente (otras veces no). Este elemento lo que si realiza son acciones maliciosas con distintos fines. Además, este troyano puede o no ser un RAT.

- Bot: Habitualmente es un troyano con funcionalidades de RAT para poder ejecutar ordenes, entregar resultados y descargar o instalar los módulos necesarios para desarrollar sus actividades maliciosas.

3.1.2 Formas de infección

Ya se ha comentado sobre la denominación de "zombis" a las máquinas infectadas por un bot. La similitud con la ficción sobre los zombis ha sido perfecta para el entendimiento del proceso nos ayuda a entender el concepto de control y gestión de dichas máquinas. Sin embargo, para explicar cómo se crean estos "zombis" no podemos seguir con esta similitud, ya que es diferente de la ficción en el caso de la infección.

Así como un zombi en la ficción infecta o contamina a otro ente a través de una mordedura, cuando hablamos de botnes somos nosotros los que nos "mordemos" a nosotros mismos y nos contaminamos (por decirlo de alguna manera).

Normalmente el medio o vector de "contaminación" o "infección" es ejecutado por el propio usuario de la máquina víctima de ser zombi (consciente ó inconscientemente).

El software empleado inicialmente en las infecciones suele ser un programa tipo "troyano" y/o "rootkit". Ambos "ocultan" su apariencia y sus acciones y toman el control de la máquina a nivel administrador (root).

La base de cualquier infección es la estrategia de "echar el anzuelo y esperar". Algunos sistemas están suficientemente protegidos y son sus usuarios lo bastante incautos para ser infectados por un malware tipo troyano o bot, ya que no mantiene las debidas precauciones o medidas de seguridad. Además, sabiendo que en internet están conectados miles de millones de dispositivos la infección de solo un pequeño porcentaje de ellos podría posibilitar la formación de una botnet de varios millones de máquinas.

Las principales formas de infección son las siguientes (todas achacables al usuario):

- Descarga e instalación de software infectado que realiza el propio usuario.

- Ejecución de enlaces a correos (emails) que realiza el usuario.

- Visita a páginas web infectadas o maliciosas (drive-by downloads).

- Falta de seguridad y parches de actualización en el sistema operativo víctima.
- Servicios con contraseñas débiles o por defecto (IoTs).
- Empleo de dispositivos (USB's, DVD, HDD's, etc.) infectados.
- Conexiones a recursos infectados.
- Empleo de redes WIFI no seguras o desconocidas (normalmente gratuitas y abiertas)
- Etc.

Los dispositivos que pueden ser infectados por estos bots y convertirse, por tanto, en máquinas zombis son todos aquellos los relacionados con las comunicaciones y/o la informática, es decir cualquiera que sea capaz de ejecutar código. Ejemplos de ellos serían ordenadores personales, tablets, smatphones, dispositivos de Internet de las cosas (IoT), servidores, máquinas de control industrial...etc.

En las plataformas Windows, es habitual que los usuarios se descarguen programas desde internet sin saber exactamente qué es lo que hace el programa. Este software podría contener un bot, una vez que el programa se ejecuta, puede escanear la red de área local, disco duro, puede intentar propagarse usando vulnerabilidades conocidas de Windows, etc.

En entornos como UNIX, GNU/Linux o BSD, muy empleados en IoTs, la forma de ataque a servidores para construir y expandir una botnet suele por telnet o SSH (puertos 23 y 22). Esto se realiza con el método de "prueba y error", probando usuarios comunes y contraseñas por defecto o al azar o con herramientas de fuerza bruta. También se pueden aprovechar ataques a bugs conocidos que no se hayan corregido.

La forma de infección que llevó a Mirai a "secuestrar" millones de dispositivos IoT fue con el usuario y contraseña por defecto que casi nadie se molesta en cambiar en este tipo de elementos.

3.1.3 Pasos en la creación y ataque de una botnet

La taxonomía de una infección y ataque sigue los siguientes pasos:

1. *El atacante prepara su ejecutable con el bot y lo distribuye a través de uno o varios medios de difusión*
2. *El usuario recibe ese archivo infectado y lo ejecuta. Este es recibido habitualmente por medio de una infección de malware en páginas web, correo spam, archivos maliciosos...etc.*
3. *El equipo queda comprometido (empleando alguna vulnerabilidad que este tenga) y el exploit ha funcionado (hay casos en que esto no se da debido a que el equipo se encuentra suficientemente protegido).*
4. *El equipo exfiltra información (primeramente sobre el).*

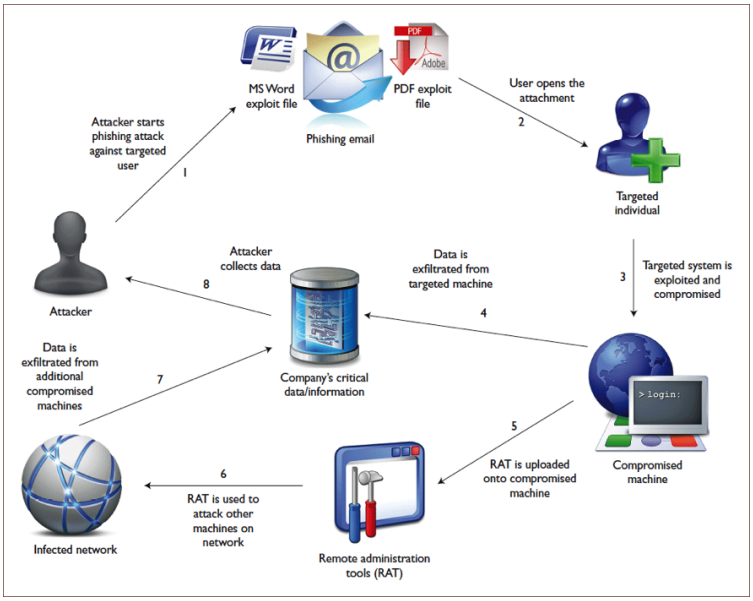


Figure 3.1: Taxonomia de la infección de un equipo [47].

5. *El equipo comprometido se descarga una herramienta tipo RAT (Remote Access Tool), si esta no se encontraba ya en el paquete que ha ejecutado el usuario, para ponerse al servicio de la red botnet.*

6. *El equipo se añade (se incluye) dentro de la red botnet.*

7. *Todos los datos que se requieren son enviados vía sus servidores C&C (o en esquemas P2P entre pares). Desde estos servidores se les envía órdenes para que así los equipos pueden ejecutar de manera coordinada cualquiera de las acciones / ataques o descargarse otro tipo de malware.*

8. *El atacante accede a los recursos buscados o realiza el ataque deseado.*

Todas estas fases de infección se puede ver en el esquema de la imagen 3.1.

3.2 Identificación de topologías, comunicación y formas de ocultación

Las redes zombis han ido aumentando su complejidad, llegando a utilizar técnicas bastante sofisticadas para asegurar su prevalencia (incluso a competir entre ellas por cada equipo infectado, eliminando las infecciones por redes rivales), han pasado de utilizar el protocolo IRC para establecer la comunicación con el centro de control C&C a utilizar HTTP o HTTPS e incluso mensajeria de twiter, que son mas difíciles de detectar al confundirse con tráfico legítimo.

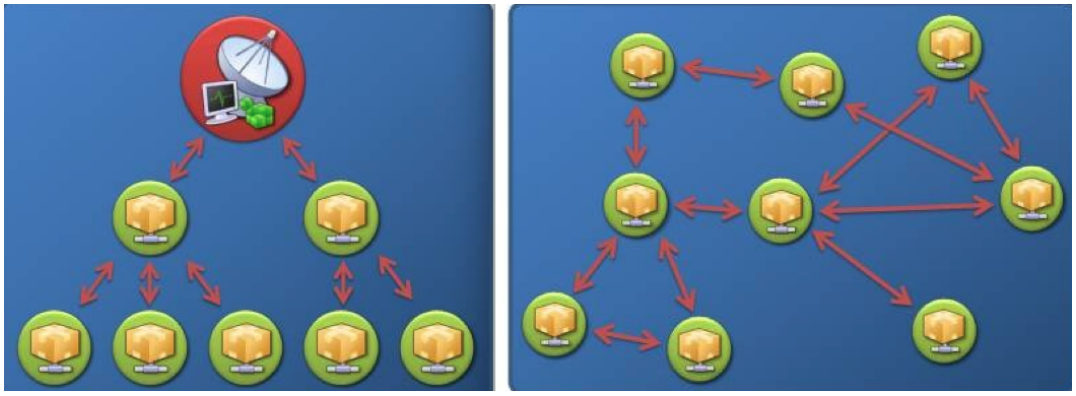


Figure 3.2: Topologías jerárquica y aleatoria (P2P) [5].

Las topologías que pueden existir en los botnets (como en casi cualquier sistema en red) se establecen en función de sus servidores C&C o de si incluso estos existen (P2P) y son las siguientes:

- **Topología en estrella:** Los robots están organizados alrededor de un servidor central.
- **Topología de varios servidores:** Es similar a la topología en estrella, pero hay diversos centros de control que evitan que la red tenga un punto único de fallo al desactivarse el C&C. Además permite planificar una distribución geográfica que mejore las latencias en la comunicación.
- **Topología jerárquica:** Varios servidores C&C se organizan en grupos por niveles.
- **Topología aleatoria:** La ventaja que proporcionan estas redes a sus administradores es que son muy difíciles de desconectar, ya que no hay dependencia con los servidores C&C. La desventaja es que es muy fácil localizar a sus miembros a partir de uno, además se pueden producir grandes latencias entre hosts al ser impredecibles los caminos en su comunicación. La topología de red es P2P.
- **Topología híbrida:** Una mezcla de alguna de las anteriores.

De estas topologías se desprenden al menos 3 modelos de comunicaciones:

- Modelo de arquitectura C&C
- Modelo de arquitectura P2P
- Modelo híbrido.

3.2.1 Modelo de arquitectura C&C

En este modelo podemos englobar aquellos que usan medios de comunicaciones como IRC y HTTP (aunque podríamos considerar también este modelo a los que usan medios de redes sociales -Web

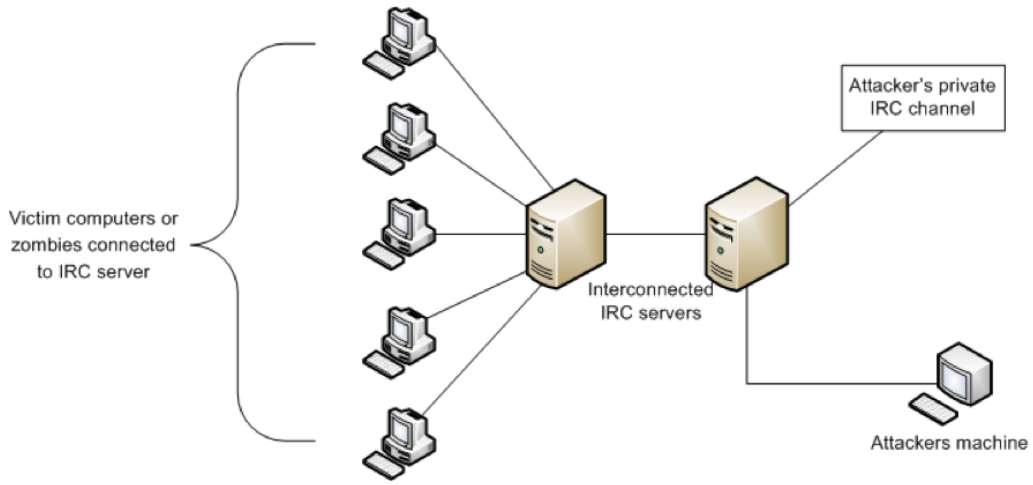


Figure 3.3: Modelo de arquitectura IRC [46].

2.0-. Estos modelos se caracterizan por que existe roles de BootMaster, Servidores C&C y Bots (o máquinas zombis).

El empleo de canales IRC es el precursor de las botnets y aunque hoy en día su uso ha decrecido definiremos que tipo de comunicaciones son estas. Este modelo de comunicaciones en botnets se podría considerar como una primera generación de botnets.

IRC (Internet Relay Chat) es un protocolo de comunicación en tiempo real basado en texto, que permite conversaciones entre dos o más personas. Se diferencia de la mensajería instantánea en que los usuarios no deben acceder a establecer la comunicación de antemano, de tal forma que todos los usuarios que se encuentran en un canal pueden comunicarse entre sí, aunque no hayan tenido ningún contacto anterior.

Las conversaciones se desarrollan en los llamados canales de IRC, designados por nombres que habitualmente comienzan con el carácter # o & (este último solo es utilizado en canales locales del servidor). Es un sistema de charlas ampliamente utilizado por personas de todo el mundo. Los usuarios del IRC utilizan una aplicación cliente para conectarse con un servidor, en el que funciona una aplicación IRCd (IRC daemon o servidor de IRC) que gestiona los canales y las conversaciones.

En el caso de una botnet, el “cliente” es el bot que existe en la máquina infectada (zombi) y el “servidor” realiza las funciones de servidor C&C. El BotMaster puede ser un simple cliente de ese servidor C&C que utiliza el canal para retransmitir sus órdenes y recibir lo que haya solicitado. Usa el protocolo IRC sobre TCP.

HTTP/HTTPs (Hypertext Transfer Protocol -la “s” es de seguro-) es un protocolo de comunicación que permite las transferencias de información en la World Wide Web. Emplea la arquitectura cliente/servidor sobre puertos conocidos (80 – > http y 443 – > https) para comunicarse. HTTP es un protocolo sin estado, es decir, no guarda ninguna información sobre conexiones anteriores. El modelo http podría ser considerado como una segunda generación de botnets.

En este caso el servidor C&C hace de servidor WEB con una BBDD asociada y los clientes son los bots de las máquinas zombis. De esta manera el cliente sabe cuál es la IP del servidor y accede a ella tanto para entregar como para recoger información.

Los anteriores protocolos de capa de aplicación (IRC y http) necesitan que su servidor este en una dirección IP o dominio (en su caso) definido. Para hacer saber a sus clientes donde se encuentra su o sus servidores C&C (lo normal es que haya varios alternativos) se puede actuar de tres maneras:

- Fijados al inicio en el código del propio bot (ver imagen 3.4).
- Obteniéndolo de un medio de red social (twiter, linkedin, blogs, archivos de texto...etc.)
- Empleando métodos dinámicos de creación de dominios o IP's aleatorias

Obviamente a través del primer método necesita tener un dominio activo fijo o IP fija. Esto hace que con colaboración se puedan localizar fácilmente donde están alojados los servidores C&C y eliminarlos "de la ecuación". Baste decir que con romper las comunicaciones entre bots y servidores C&C o entre servidores C&C y el BotMaster la botnet estará desmantelada.

En el caso de que existieran varios dominios asociados solo habría que ir eliminando uno a uno ya que al quedarse sin comunicación con el primero el bot intenta conectarse al siguiente alternativo y así nos delataría su ubicación.

Con el enfoque dinámico de empleo de dominio suele ser mucho más difícil acabar con la botnet. No obstante, el BotMaster debe de saber que dominios estarán activos a cada momento. En este caso la creación y gestión de la botnet es más compleja, aunque es mucho más resilente (Capacidad de un sistema tecnológico de soportar y recuperarse ante desastres y perturbaciones).

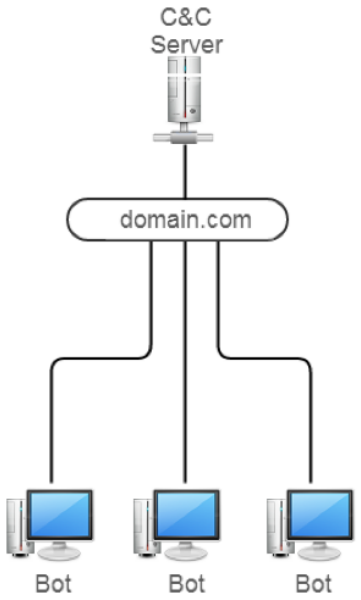


Figure 3.4: Modelo de arquitectura C&C [29].

3.2.2 Modelo de arquitectura P2P

Contra el modelo "C&C" (ver figura 3.4) en la que los bots deben de establecer un contacto con su servidor C&C alojado (normalmente) en un dominio conocido, existe el modelo P2P en el que esta topología no es jerárquica sino aleatoria. Esta podría ser considerada la tercera generación de botnets.

El modelo Peer to Peer (P2P) [29] intentan resolver el problema de que una botnet sea fácilmente inutilizada al establecerse un vínculo definido con un dominio o IP. La idea subyacente es que todos los robots se conecten y se comuniquen entre sí para eliminar la necesidad de un servidor centralizado, sin embargo, esto conlleva una serie de dificultades técnicas. Las dificultades principales son tres:

- Establecer un sistema de comunicaciones "verificable" con el BotMaster.
- Evitar el problema de los DPP (Dispositivos de Protección Perimetral) para la conexión entrantes entre pares (peers).
- Como introducir un nodo en una red P2P.

Antes de explicar este modelo quiero recalcar que un bot puede actuar sobre la máquina infectada, no sobre un firewall externo que exista en la red local de esa máquina zombi.

Comunicación BotMaster con botnet P2P

Si los bots se comunican entre sí, entonces el BotMaster necesita asegurarse de que solo él sea el que puede comandar a esos bots. Para poder implementar esto se suele emplear la firma digital (Ver figura 3.5). Si no se hace así, cualquiera que logre acceder a un bot podría comandar a toda la botnet entera.

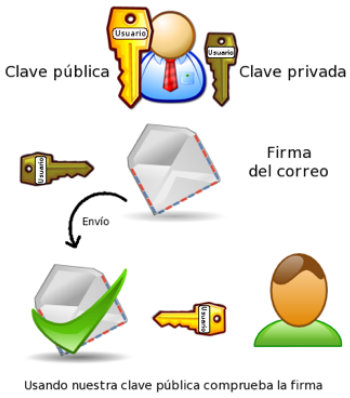


Figure 3.5: Cifrado clave publica/privada [22].

Una firma digital se realiza mediante cifrado asimétrico, este cifrado que requiere dos claves (pública y privada). Si se utiliza una clave para cifrar un mensaje, sólo se puede descifrar con la otra clave. Si el BotMaster mantiene la clave secreta (clave privada) e incorpora la otra clave en el bot (clave pública), puede usar su clave para cifrar comandos y de esta manera solo los bots podrían descifrarlos usando la clave pública y así estar seguro de que este proviene del BotMaster.

El problema de las comunicaciones entrantes a través de un DPP

La idea de la mayoría de las personas cree que en una botnet P2P los bots se conectan entre sí a través de la dirección IP, reenvían los comandos entre sí, eliminando la necesidad de un servidor central o dominio, esta representación es incorrecta en un entorno real "internet".

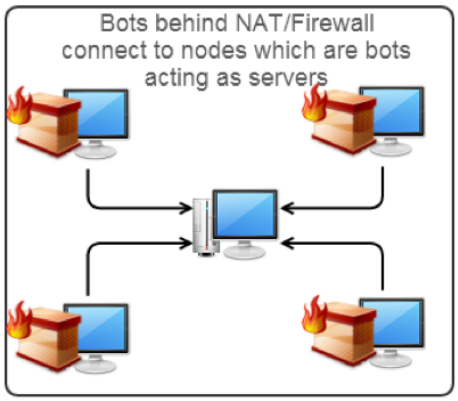


Figure 3.6: Esquema P2P/Server tras NAT [29].

Los equipos que están detrás de un NAT, un Firewalls, o que utilizan un servidor proxy

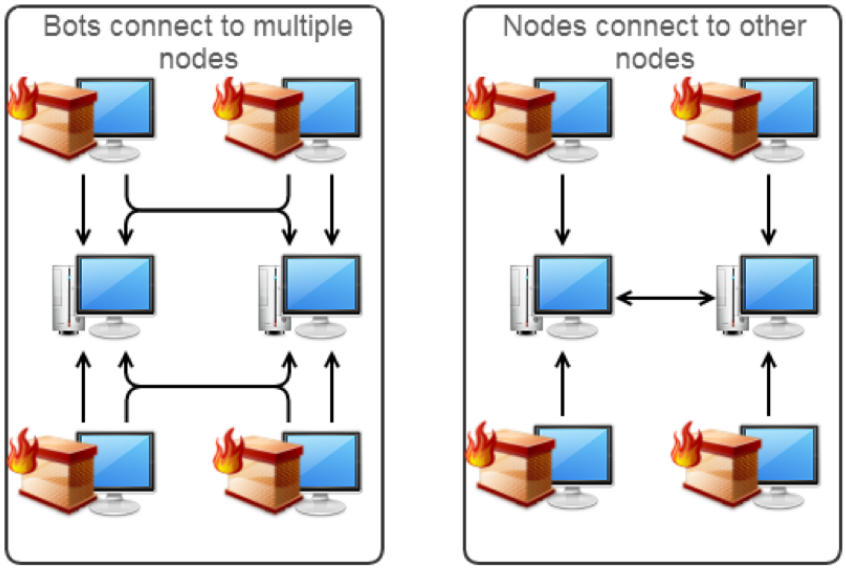


Figure 3.7: Esquemas P2P tras NAT [29].

para acceder a internet no puede aceptar una conexión entrante "nueva" (solo si es una respuesta a una solicitud anterior), pero SI pueden establecer conexiónes de salida. Esto es un problema para estas redes, ya que impediría que la mayoría de los robots que están conectados a otros bots se comunicaran entre sí. En el esquema tradicional (servidores C&C), esto obviamente no es un problema ya que los bots se conectan al servidor. Así pues, las redes P2P, en realidad, necesitan de "servidores" en este esquema.

Estos "servidores" serán los equípos que son capaces de aceptar conexiones entrantes (no están detrás de un Proxy/NAT/Firewall) actúan como servidores (normalmente son llamados "nodos" o pares), los bots que no son capaces de aceptar conexiones (normalmente llamados "trabajadores") se conectarán a uno o más nodos para recibir órdenes (Ver figura 3.6).

Aunque los "nodos" (servidores) son técnicamente servidores y clientes a la vez, se utilizan de manera que no se produzca un esquema jerárquico de conexiones, es decir que a los "trabajadores" se les distribuye entre muchos nodos, lo que les permite cambiar de un nodo a otro si este cae.

Las Botnets P2P **sólo funcionan si hay suficientes "nodos" que hagan de servidores** y cuantos mas existan mejor (ya que la labor de bloquear todos es inmensa). Por otro lado, hay que tener en cuenta que los "nodos" no dejan de ser ordenadores legítimos (obviamente infectados igual que los "trabajadores") y que no tienen la misma capacidad que un servidor dedicado.

En este esquema, cada nodo mantiene una lista de direcciones IP de otros nodos que se comparte con los trabajadores, los trabajadores almacenan las listas, lo que les permite cambiar de nodo si al que están conectado se inhabilita.

Si no establecemos un sistema de comunicación entre los nodos a nivel de comandos (a nivel de listas de direcciones IP ya existe) nuestra red P2P sólo sería muchos pequeños grupos de robots conectados a muchos nodos diferentes, lo que sería imposible de mandar. Para que los comandos

circulen por toda la red con cierta agilidad se pueden emplear las siguientes estrategias (Ver figura 3.7):

- Que los bots "trabajadores" se conecten a varios nodos y pasen los comandos recibidos a los otros nodos a los que también están conectados.
- Que los nodos se conectan a otros nodos y se pasen los comandos entre ellos.
- Que se emplee una combinación de las dos estrategias anteriores.

Como introducir un nodo en una red P2P

Para que un bot se una a la red P2P necesita de al menos la dirección IP de un nodo, aquí es donde aparece el concepto de **bootstrapping** (término empleado para describir el arranque o inicio de cualquier ordenador) en la red. El bot está ya configurado (en su propio código) con una lista de servidores de arranque, a los que se conecta por primera vez el bot del equipo infectado.

Así pues, debe de existir un nuevo elemento (que podría ser uno de los nodos ya utilizados y que el BotMaster lo considere estable) que sería un servidor o nodo de arranque. Su misión es mantener una gran lista de direcciones IP de otros nodos, proporcionando a los nuevos bots que se conecten una lista más reducida de nodos IP's, ahora, con estos nuevos "trabajadores" o "nodos" (según el caso) son introducidos en la red botnet P2P.

Generalmente los servidores de arranque proporcionan algún tipo de firma, lo que impide que sean secuestrados por los especialistas de seguridad y utilizados (por ejemplo) para dar a los nuevos robots IP's de nodo no válidos con lo que se conseguiría que las nuevas máquinas zombis no fuesen agregadas a la red.

Obviamente, los servidores de arranque son un punto central de gestión y por ende de "debilidad" del sistema (igual que los servidores C&C en el esquema tradicional). Sin embargo, si todos los servidores bootstrap fueran incautados al mismo tiempo, no afectaría a los robots que ya se encuentran en la botnet, solo evitaría que nuevas infecciones se unieran. El BotMaster puede, simplemente, dejar de infectar nuevos sistemas hasta que puedan configurar nuevos servidores de arranque, esto sería sólo una retención temporal por lo que no es definitivo atacar el sistema de arranque para acabar con una botnet P2P.

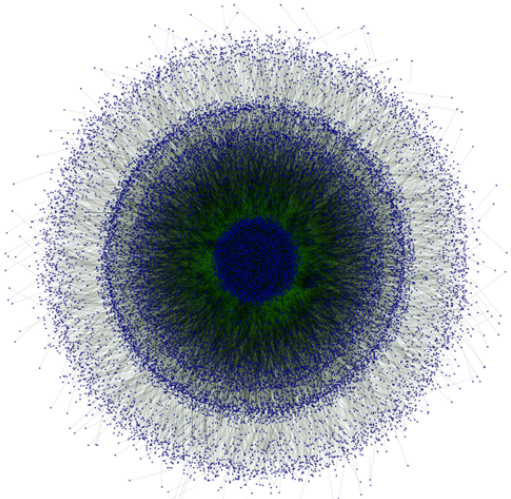


Figure 3.8: Topología P2P en internet [28].

Aunque el esquema de las botnets P2P parezca sencillo de implementar, esto no es así. Existe una serie de tareas que cada bot debe de hacer. Inicialmente un bot tiene asignado un OID (Object Identifier) único en su red y debe de seguir los siguientes tres pasos nada más conectarse (depende del tipo de comunicación, peer-nodo1-nodo2 o nodo-nodo o mixto):

- Conexión – > El bot usa los mensajes de conexión para informar su OID a otros bots (peers) y para recibir una lista de peers cercanos (nodos y trabajadores).
- Búsqueda – > El bot utiliza los mensajes de búsqueda para encontrar recursos para otros nodos basados en OID. Aquí emplea las direcciones descargadas de los servidores nodo.
- Publicitarse – > El bot (peer) utiliza mensajes para informar la propiedad de los recursos de red (OID) para que otros pares puedan encontrar el recurso posteriormente.

Con este último ya el nodo se encuentra en la botnet P2P, es conocido y está activo. Por último se quiere mostrar una imagen sobre la topología de una red P2P (ver figura 3.8) establecida por Zeus GameOver, en la que se aprecia su topología aleatoria.

3.2.3 Modelo híbrido

Teniendo en cuenta los problemas encontrados por las botnets del modelo C&C (Clasico) y la de P2P, sería deseable encontrar un modelo que disfrutara de las ventajas de ambos esquemas y que no tuviese sus inconvenientes. Así, lo ideal sería disponer de un modelo de botnet donde:

- La botnet sea robusta y capaz de mantener el control de sus bots incluso después de que una parte sustancial de su población de bots (sean nodos o servidores) hayan sido inhabilitados.
- Evitar el mostrar la topología de red botnet cuando algunos de sus bots sean capturados.
- Sea fácil realizar la supervisión y la obtención de la información completa de una botnet por su BotMaster.
- Evitar o dificultar la detección de tráfico característico de una botnet (ofuscamiento de patrones) para dificultar su detección.
- Realizar un diseño que considere otras cuestiones como direcciones de dominio o IP dinámicas o privadas, el cifrado de la información y el uso de firmas digitales para su mando.
- Una posible aproximación a este escenario sería el empleo de una botnet P2P/C&C híbrida.

Esta botnet tiene las siguientes características:

- La botnet no requiere ningún procedimiento de bootstrap.
- La botnet se comunica a través de la lista de pares contenida en cada bot. Sin embargo, cada bot tiene una lista de pares de tamaño fijo y limitado y no revela su lista de pares a otros bots. De esta manera, cuando un bot es capturado por los defensores, sólo el número limitado de bots en su lista de pares se exponen.

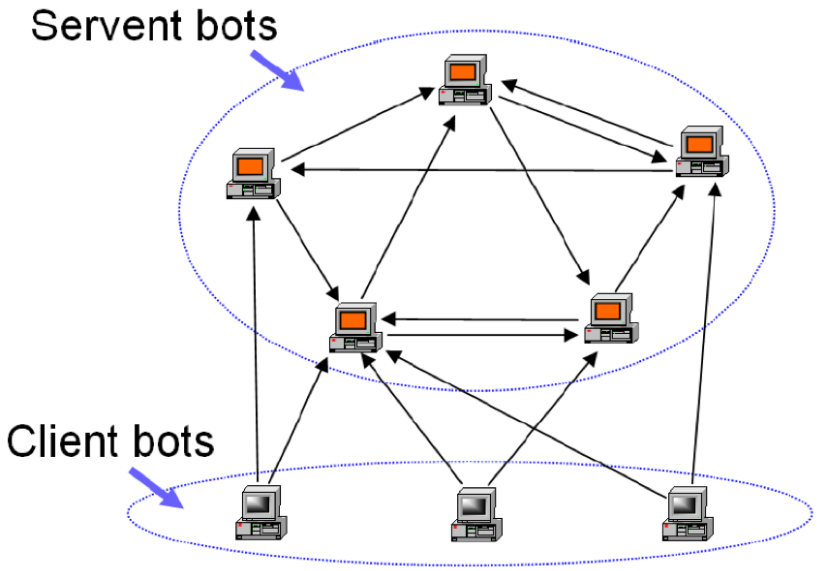


Figure 3.9: Topología de botnets híbrida [2].

- Un BotMaster podría monitorear fácilmente la botnet entera emitiendo un solo comando. Este comando ordena a todos los bots que informen a una máquina comprometida específica (que se denomina host de sensor) controlada por el BotMaster. La dirección IP del host del sensor cambiará cada vez que se emita un comando de informe para evitar que los defensores capturen o bloqueen el host de sensor.

- Después de recopilar información sobre la botnet a través del comando de informe anterior, un BotMaster, si lo considera necesario, podría emitir un comando de actualización para activamente dejar que todos los bots contacten con un host de sensor para actualizar sus listas de pares. Esto reorganiza la botnet de tal manera que tiene una conectividad balanceada y robusta que podría volver a conectar una botnet rota.

- Sólo los bots con direcciones IP globales estáticas y que son accesibles desde internet son candidatos para estar en listas de pares (“nodos” que se comportan tanto con características de cliente como de servidor).

- Cada bot “nodo” escucha en un puerto de servicio autodeterminado para las conexiones entrantes de otros bots y utiliza una clave de cifrado simétrica autogenerada para el tráfico entrante. Este cifrado individualizado y diseño de puerto de servicio individualizado hace que sea muy difícil para la botnet ser detectada a través de análisis de flujo de red del tráfico de la comunicación botnet.

Topología de la arquitectura híbrida P2P Botnet [2] Los robots en la botnet P2P propuesta se clasifican en dos grupos. El primer grupo contiene bots que tienen direcciones IP estáticas y no privadas y son accesibles desde internet, son los llamados “servents” o “nodos” (peers)

y se comportan como clientes y servidores. El segundo grupo contiene los bots restantes (les llamaremos por simplicidad "clients") incluyendo:

- Bots con direcciones IP asignadas dinámicamente.
- Bots con direcciones IP privadas.
- Bots detrás de cortafuegos de tal manera que no se pueden conectar desde internet.

El segundo grupo de bots se llaman robots clientes ya que no aceptarán conexiones entrantes. Sólo los bots servent son candidatos en las listas de "peers", es decir para trabajar en un modelo P2P. Todos los bots contactan activamente a los bots servent (incluso entre ellos mismos). Debido a que los robots servent normalmente no deberían cambiar sus direcciones IP, este diseño aumenta la estabilidad de la red botnet.

Un bot puede determinar fácilmente el tipo de dirección IP utilizada por su máquina host. Por ejemplo: en una máquina Windows, un bot podría ejecutar el comando "ipconfig/all".

Un BotMaster podría confiar en la colaboración entre los bots para determinar tales bots. Por ejemplo: un robot ejecuta su programa de servidor y solicita a los robots servent en su lista de pares (peers) iniciar las conexiones a su puerto de servicio. Si el bot podría recibir tales conexiones de prueba, se etiqueta a sí mismo como un bot servent. De lo contrario, se etiqueta como un robot cliente.

Forma de Mando y Control

La figura 3.9 ilustra la arquitectura de mando y control de la botnet propuesta. La botnet mostrada en esta figura tiene 5 bots servent y 3 bots client. El tamaño de la lista de pares es 2 (es decir, la lista de pares de cada bot contiene las direcciones IP de 2 bots servent). Una flecha del bot A al bot B representa el bot A iniciando una conexión con el bot B.

El BotMaster inyecta sus comandos a través de cualquier bot en la botnet. Tanto los bots client como servent se conectan activamente y periódicamente con los servent bots en sus listas de pares para recuperar comandos emitidos por su BotMaster. Cuando un bot recibe un nuevo comando que nunca ha visto antes (el comando tiene un ID único), envía inmediatamente el comando a todos los bots servent en su lista de pares.

En comparación con una botnet de tipo C&C, es fácil comprobar que la botnet P2P híbrida es en realidad una extensión de una botnet C&C. La botnet P2P híbrida es equivalente a una botnet C&C en la que los bots servent toman la función de servidores C&C. El número de servidores C&C (servent bots) se amplía mucho y se interconectan entre sí. De hecho, el gran número de robots servent es la principal razón por la que la botnet P2P híbrido propuesto es muy difícil de cerrar.

Para la autenticación de comandos un BotMaster debe generar un par de claves públicas/privada y codifica la clave pública en el programa bot antes de liberar y construir la botnet (en su código fuente). No hay necesidad de distribución de claves porque la clave pública está

codificada en el programa bot. Posteriormente, los mensajes de comando enviados desde el Bot-Master deben de ser firmados digitalmente por la clave privada para asegurar su autenticación e integridad.

Un ejemplo de una botnet P2P híbrida es Necurs [26] que apareció por primera vez en 2013, sigue activo y afecta a sistemas Windows 7, 8 y 10. Tiene características de fraude electrónico, DDoS, etc. Esta botnet es mucho más compleja de crear y cuyas características de cifrado y comunicaciones son:

- Todos los peers responden a las solicitudes con un paquete firmado por una clave RSA de 2048 bits del BotMaster (impidiendo que cualquier persona aparte de ellos introduzca nuevas cargas útiles de respuesta).
- Los payloads de P2P pueden contener cualquiera o todos los 3 tipos de bloque:
  - Bloque DNS: Lista de servidores DNS que permitirán al bot usar los dominios ".bit" de Namecoin para encontrar a los servidores C&C.
  - Bloque C&C: Lista de IP's de servidor C&C.
  - Bloque de actualización: permite que las actualizaciones se envíen a través de la red P2P (probablemente sólo se utiliza en el caso de que todos los demás métodos de C&C estén fuera de línea).
- El BotMasters puede optar por generar listas de peers y publicarlos a través del servidor C&C. Al distribuir a los peers de esta manera, es más fácil prevenir los ataques de envenenamiento.
- Cualquier cosa descargada de la red C&C o P2P se almacena en la carpeta temp con un UUID para el nombre de archivo y .tmp para la extensión. El UUID es generado por el hash SHA1 del identificador (OID) de bot con un entero estático de 64 bits (utilizado para identificar el contenido del archivo).
- Los archivos se cifran con RC4 usando una clave derivada con la misma función que anteriormente, excepto que las variables generadas aleatoriamente están juntas con los datos y luego son almacenadas al final del archivo.

3.3 Medios de ocultación de los bots

Antes de hablar de la posible ocultación de los botnets debemos de tener en cuenta que para evitar medios de detección de malware al distribuir nuestro bot en las máquinas víctimas se pueden emplear cualquiera de las técnicas que evitan los antivirus, IDS, IPS o cortafuegos. Algunas son:

- Autocifrado del paquete evitando que se puedan corresponder las firmas.

- Dividir en módulos la carga y desordenarla para que sea ordenada en destino.
- Empleo del polimorfismo.
- Empleo de un paquete inicial que parezca inocuo para luego descargarse paulatinamente paquetes malware.
- Etc.

Lo que si es necesario (habitualmente) es que este malware (futuro “bot”) tenga privilegios de administrador (o root) y para ello o se lo concede al instalar expresamente el usuario (sucede mucho más de lo que podemos creer) o se aprovecha de una configuración deficiente de la seguridad de las aplicaciones (como el navegador) o busca algún exploit, algún bug existente o se emplea alguna técnica que eleve sus privilegios a root (como smash the stack).

Una vez que la máquina ha quedado infectada con privilegios de administrador y se une a la red botnet ya solo queda emplear técnicas de ocultación de las actividades que realiza la propia red para que no pueda ser detectada o en su caso desmantelada.

3.4 Búsqueda de Resiliencia (servidores C&C)

La capacidad de un agente de bot para localizar la infraestructura tipo C&C es un requisito necesario ya que sino un agente bot no podrá recibir nuevas instrucciones.

Mientras que algunos agentes bot pueden funcionar en modo exclusivo "zombi", lo cierto es que la mayoría de los bots siguen recogiendo información local del anfitrión (máquina infectada) a horas regulares. Los operadores de botnet utilizan una serie de tecnologías para aumentar la probabilidad de que los bots puedan localizar su infraestructura C&C aunque esta sea “atacada” por responsables de seguridad o incluso por otro cibercriminal rival que quiera apoderarse de ella.

Una tecnología clave que permite la resolución de localización de los C&C “correctos” y asi obtener la resiliencia es denominado "fluxing" [5]. El Fluxing tiene dos modelos principalmente:

- IP Flux (Fast Flux)
- Domain Flux

Ambas tecnologías son ampliamente utilizadas por los BotMaster.

3.4.1 IP Flux (Fast Flux)

Se asocian varias IP’s con un registro DNS y además cambian rápidamente las IP’s asociadas. Hay dos tipos de fast-flux:

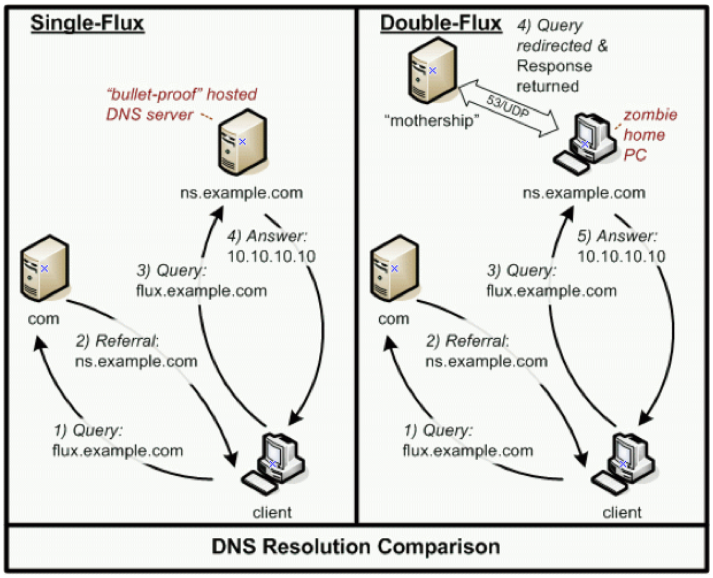


Figure 3.10: Fast Flux vs Doble Fast Flux [21].

- Single-flux: cientos o miles de IP's asociadas a un nombre de dominio son registradas y de-registradas rápidamente. RoundRobin de DNS + TTL bajo, en la configuración de los registros.

- Double-flux: además de cambiar las IP's asociadas a los registros cambian los servidores DNS asociados al nombre de dominio.

La gran cantidad de IPs asociadas a un solo dominio se consigue a través del uso de "nodos" capturados (máquinas zombis) que realizan la función de redirigir el tráfico al servidor C&C final.

En la anterior imagen 3.10 se puede ver la comparación entre un Fast Flux (Single Flux) y un Doble Fast Flux.

3.4.2 Domain Flux

El domain flux es el inverso del flujo anterior y se refiere al cambio constante y asignación de múltiples FQDN (fully qualified domain name) a una única dirección IP o infraestructura C&C. Las técnicas aplicables al Domain Flux abarcan el Domain Wildcard y el Domain Generation Algorithms -DGA- (algoritmos de generación de dominio):

- **Domain Wildcard:** Usa de la funcionalidad DNS nativa wildcard (*) al dominio superior tal que todos los FQDN que apuntan a la misma dirección IP. Por ejemplo: *.uoc.edu podría encapsular tanto pec.tfm.uoc.edu como tfm.uoc.edu. Esta técnica es comúnmente empleada por botnets que se usan para el envío de contenido de spam y phishing. Mediante esta técnica se podría emplear información que aleatoria (por ejemplo, "mnbpstr" para

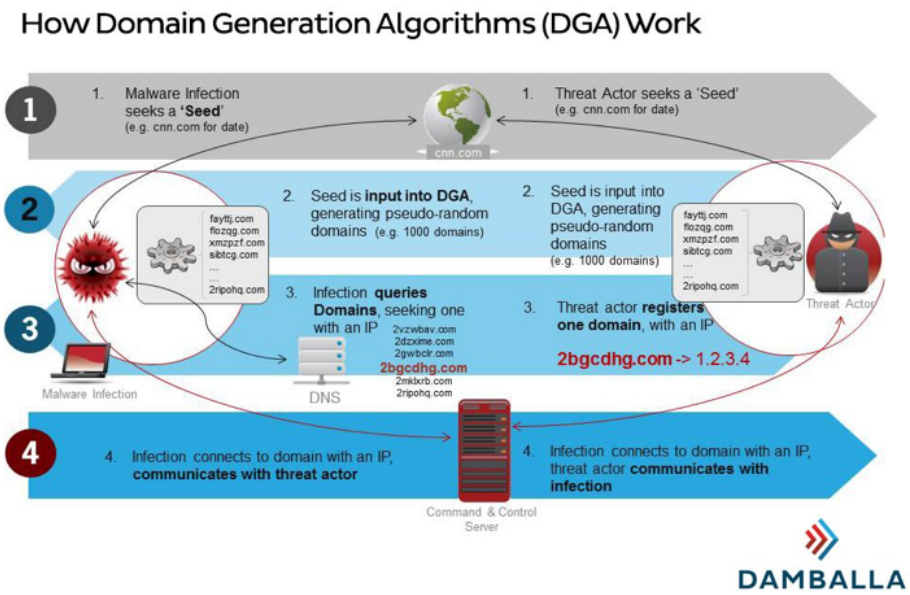


Figure 3.11: Funcionamiento del DGA [3].

mnbpstr.tfm.uoc.edu), de esta manera el BotMaster puede identificar de manera única a una víctima.

- **Domain Generation Algorithms:** Es la técnica más reciente de las aplicadas. Se basa en que los agentes bots generan una lista dinámica de múltiples FQDN cada día, que luego son consultados por el propio bot al tratar de localizar la infraestructura C&C. Dado que el dominio creado se genera dinámicamente y, normalmente, tiene una vida de sólo un día, la rápida rotación hace que sea muy difícil investigar o bloquear cada nombre de dominio posible. Esta técnica se explicará más en detalle para poder entenderla.

La implementación de este tipo de algoritmo de generación de dominios la podemos dividir en las siguientes cuatro fases (ver imagen 3.11):

1. El atacante o BotMaster genera una semilla, que es la que utilizarán sus bots para generar los dominios aleatorios.

2. El bot dispone del algoritmo DGA para que, utilizando la semilla, genere aleatoriamente los dominios (a veces más de 1000 diarios).

3. El equipo infectado intentará conectarse a cada uno de los dominios, donde tan solo uno será valido y en el que habrá sido registrado previamente por el atacante. Esto lo puede realizar ya que dispone de la misma semilla y el mismo algoritmo que los bots.

4. El atacante tendrá control sobre la botnet para dar nuevas órdenes, obtener información o desplegar actualizaciones del malware.

3.4.3 Blind Proxy Redirection

Tanto IP Flux como Domain Flux proporcionan niveles avanzados de redundancia y resiliencia para la infraestructura C&C de una botnet. Sin embargo, los BotMaster suelen emplear una segunda capa de abstracción para aumentar aún más la seguridad.

Esta técnica se basa en utilizar un servidor proxy (o varios de ellos) para realizar una redirección para llegar a los servidores finales C&C. La redirección ayuda a interrumpir los intentos de rastrear o apagar redes de servicio IP Flux. En este caso el BotMaster emplea agentes bot que actuaran como redireccionadores que canalizan las solicitudes y los datos hacia y desde otros servidores bajo el control del operador de la botnet.

En la imagen 3.12 se puede ver el esquema de Fast Flux con redireccionamiento a través de una máquina zombi [21].

Como la mayoría de las botnets dependen del DNS como el servicio de ubicación de la infraestructura de C&C a continuación se establece una clasificación de robustez según el método empleado.

- Muy debil– > Dominio único
- Debil – > IP Flux (Single Flux).
- Resistente – > Doble Flux
- Muy resistente – > Domain Flux.

Otras técnicas de ofuscación son son:

- Ocultar el tráfico de la botnet con el tráfico habitual y para ello emplean medios de redes sociales. Algunas pueden ser Linkedin, Twiter, Blogs, etc que se utilizan como servidores C&C donde se pueden dejar los comandos que son leídos por los bots y estos envían la información donde se les ha dicho, como por ejemplo un blog.
- Técnicas criptográficas en redes sociales. Existen otros (como Stegobot) que, además de usar estos medios sociales para su comunicación, emplean técnicas estenográficas en imágenes que se cuelgan en estas redes sociales, pasando del todo inadvertidas para cualquiera que acceda a ellas.
- Establecer comunicaciones y tráficos aleatorios entre bots o de estos con los servidores C&C.
- Ocultación a través del cifrado de tráfico o del uso de túneles cifrados (como IPSec) a través de proxies.
- El domain shadowing [35]. Esta técnica consiste en tomar el control de un dominio registrado consiguiendo las credenciales de administración de modo que sea posible crear

registros DNS para nuevos subdominios. Creando multitud de subdominios, el atacante configura una lista lo más amplia posible. Este comportamiento ha demostrado ser altamente efectivo para evitar técnicas de bloqueo típicas como el blacklisting y/o sinkholing de sitios o direcciones IP. A diferencia de fast-flux donde se cambia rápidamente la IP asignada a un único dominio, domain shadowing rota subdominos asociados a un dominio. Esos subdominios pueden apuntar bien a una única IP, o aun conjunto de ellas según las necesidades y circunstancias.

La base de todas las técnicas es utilizar tráfico legítimo, habitual y de muy costoso análisis debido a que volumen de datos que existe entre los bots es muy pequeño.

Conclusiónes

Independiente de la topología, múltiples capas de flujo de DNS y redirección hacen a algunas botnets altamente resistentes a su parada o descubrimiento.

Las botnets que utilizan redes P2P o P2P híbridas son también altamente difíciles de deshabilitar.

Afortunadamente, existen pocas botnets tan avanzadas, aunque siempre un análisis concienzudo de una máquina zombi puede desvelarse como tal, aunque el detectar los servidores C&C o los "nodos" (en el caso de redes P2P) es mucho más complicado.

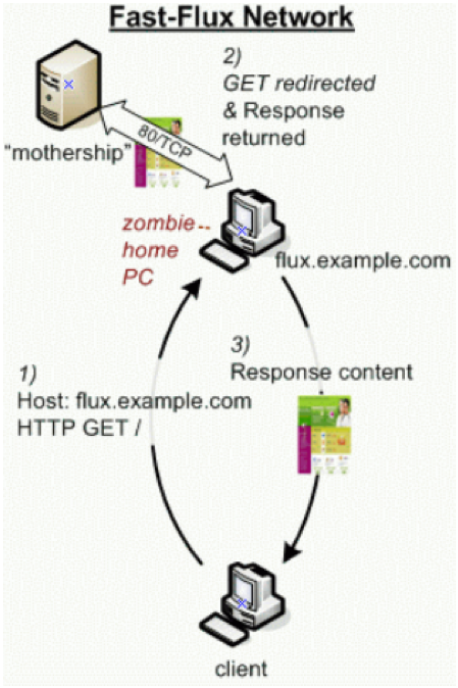


Figure 3.12: Exfiltración de información DGA [21].

Exfiltración de información (imagen 3.12)

Dentro del proceso de comunicación, es importante tener en cuenta que se utilizan diversas técnicas para llevar a cabo el proceso de exfiltración de información sensible:

- Diferentes métodos: HTTP, FTP, SSH, SMTP, etc.
- Dividen la información en un gran número de ficheros de tamaño pequeño, en lugar de enviar únicamente uno de gran tamaño.
- De manera habitual, utilizan algoritmos de cifrado simétrico por sustitución para cifrar la información, los más habituales son RC4 o ROT13. No es habitual la utilización del cifrado asimétrico ya que tiene un coste computacional alto.

En el proceso de comunicación, merecen especial atención los “covert channels”, es decir, las técnicas que permiten enviar información empleando para ello las cabeceras de los protocolos de comunicación. Mediante estas técnicas es posible tanto enviar instrucciones a los nodos de la botnet como exfiltrar información sensible.

Al llegar los paquetes a su correspondiente destino, se procesan los valores de las cabeceras de modo que se obtiene información que el bot master puede comprender. Aparte de este metodo existe otro tipo de covert channels como son el “Timing Covert Channel” y el “Storage Covert Channel”.

- **Timing Covert Channel:** Se basan en alterar el tiempo y la frecuencia de envío del recurso.

- **Storage Covert Channel:** Se basa en escribir los datos en un área de almacenamiento y se leen a través de otro proceso (se podría por ejemplo para ello usar un blog y publicarlos. En este caso se pueden incluso emplear protocolos que no son de envío de información (como ICMP) para ir enviando poco a poco esa información dentro de esos paquetes.

3.5 Medios de detección y protección

Existe una infinidad de clasificaciones que podríamos exponer sobre medios de detección y protección ante las botnets. Algunas podrían fijarse en si se usan técnicas pasivas o activas, si son a nivel de host o de red, si son de detección de prevención restrictivas o correctivas, etc.

Este desafío de “clasificar” lo inclasificable se debe a la multitud de puntos de vista que puedan existir con respecto a una botnet. Así podremos verlo desde el punto de vista del atacante, de la víctima o incluso desde el punto de vista del tercero que las está detectando.

En este apartado queremos, sencillamente, ceñirnos a la clasificación inicial, es decir formas o medios de detección y formas o medios de protección desde un punto de vista de host y de red (network). Si se necesita especificar alguna de las clasificaciones anteriores se hará de manera individualizada.

3.5.1 Detección de botnets

En los anteriores apartado ha quedado claro que existen una serie de parámetros “distinguibles” que se asocian a una botnet. El tipo, la frecuencia (o cadencia), los destinos y el tamaño del tráfico es un patrón característico de cualquier servicio y por ende de una botnet.

A pesar de esto, existe multitud de modelos topológicos y medidas de protección de las botnets (como ya se ha explicado) convirtiendo la labor de su detección en una empresa difícil.

Técnicas de detección a nivel de host: La forma de detección en la que somos víctimas o verdugos (en cao de ser un equipo zombi) de una botnet es la siguiente [6]:

- Detección de una infección por el antivirus (existen muchas infecciones que no se detectarán).
- Paquetes de detección de rootkits (Conjunto de herramientas usadas frecuentemente por los intrusos informáticos o crackers que consiguen acceder ilícitamente a un sistema informático).
- Percepción en la modificación del archivo de hosts de Windows.
- La máquina tarda más de apagarse de lo normal.
- Recepción de correo extraño o tus contactos reciben correos tuyos que no enviaste.
- Aparición de ventanas emergentes aleatorias que, aunque sean, probablemente, una infección de adware, a veces puede ser una forma de actividad relacionada con botnets.
- Lentitud de la máquina. Aunque se pueda deber a otros aspectos, es una señal de alarma para malware tipo botnet o relacionado.
- Servidores de resolución DNS predeterminados en la máquina no confiables. Compararlos con los servidores DNS de la compañía ISP, o el del enrutador de su LAN interna.
- Actividad de la máquina (disco duro, uso de memoria o conexión al router del hogar) sospechosa en momentos en que no está utilizándose.
- Las aplicaciones antivirus no se actualizan adecuadamente.
- Programas con nombres sospechosos en el administrador de tareas (entornos Windows).
- Picos de funcionamiento inestable en la conexión a internet.
- Etc.

La realidad es que las posibilidades de detección a nivel de host una vez el equipo está infectado son realmente bajas. Ello nos lleva a considerar otras técnicas más efectivas (a nivel de red).

Técnicas de detección a nivel de red

El monitoreo de una red para la detección de malware (y en nuestro caso botnets) se puede realizar a nivel de nuestra red local y su conexión al exterior (y viceversa) o se puede ver desde el punto de vista de la monitorización de internet (labor más ardua) que suele ser llevada a cabo por organismos constituidos para proporcionar seguridad, como los CERT's.

Los "síntomas" que nos pueden indicar que existen sospechas de tráfico botnet en la red son los siguientes [6]:

- Trafico IRC (cada día menos común) que suele ir en texto claro (con lo que se puede analizar) y podríamos realizar búsquedas de palabras clave relacionadas con comandos de una botnet. Búsqueda de puertos IRCs predeterminados (el rango de puerto completo especificado por el RFC es 6660-6669,7000 y el 113). Hay que tener en cuenta que muchos administradores de botnet utilizarán puertos IRC no estándares.
- Emplear listas de servidores conocidos C&C para ver si existen intentos de conexión a ellos. En dsshield mantienen una lista negra de IP's generada cooperativamente por sus miembros.
- Existen firmas creadas expresamente para la detección de servidores C&C y que se pueden usar en herramientas como Snort (IDS).
- Detección de tráfico mediante análisis de paquetes. Es una técnica que emplea varias de las descritas anteriormente.
- Detección de tráfico mediante análisis de flujo.
- Detección través de tráfico DNS. Si una gran cantidad de máquinas están haciendo las mismas solicitudes de DNS, o accediendo al mismo servidor probablemente sea debido a la existencia de una botnet.
- Detección de cualquier tipo de malware en la red propia. Este podría haber sido descargado por el propio bot o tener relación con este.
- Instalación de un honeypot basado en malware en la red interna para detectar propagaciones de malware de las máquinas infectadas.
- Establecer una estrategia de vigilancia de puertos abiertos típicamente vulnerables. Por ejemplo, excesivo tráfico en los puertos 135,139,445 (intercambio de archivos Windows).
- Existencia de tráfico de escaneo de puertos. Señal inequívoca que ese están buscando servicios y vulnerabilidades en la red.
- Trafico NO correspondiente a esa máquina. Por ejemplo, si se observa tráfico SMTP en una máquina que no es un servidor SMTP (Botnet SpamThru).
- Si en nuestra red usamos un proxy http no deberíamos observar solicitudes de datos HTTP externos al proxy.
- Empleo de estadísticas en tiempo real ajustadas a patrones de botnets conocidas. Hay que tener en cuenta que muchas botnets están relacionadas entre si y sus desarrollos provienen de un código "base" común entre ellas.
- Etc.

Vamos a explicar, un poco más en profundidad, alguna de estas técnicas:

Detección de botnets mediante análisis de paquetes [14]:

Estas técnicas (incluye algunas de las referenciadas antes) consiste en inspeccionar los paquetes del tráfico de red en busca de patrones o características previamente definidas para para detectar posibles amenazas. Se basan en la simple observación, sin interferir o modificar ninguno de los elementos implicados. Estas técnicas tienen la ventaja de ser más difíciles de detectar por los BotMasters.

Ejemplos de uso de esta técnica es comprobar las IP de destino de los paquetes con las listas negras de IP clasificadas como C&C (centros de control) de botnets, monitorizar conexiones a puertos diferentes de los habituales o la búsqueda de cadenas de caracteres especiales que puedan identificar una amenaza.

En función del tipo de análisis de paquetes podríamos distinguir entre inspección superficial de paquetes (Stateful Packet Inspection o SPI) o inspección a fondo de los paquetes (Deep Packet Inspection o DPI)

El análisis SPI se centra exclusivamente en los datos de cabecera (IP's, puertos, protocolos, etc) pero no en el contenido del paquete. Este tipo de análisis SPI es el que realizan normalmente los firewalls.

Por otro lado, en un análisis DPI, además de tenerse en cuenta los datos de cabecera de los paquetes, el sistema analiza el contenido o parte útil del paquete. Este tipo de análisis es el más interesante desde el punto de vista de detección de botnets, ya que utiliza una combinación de técnicas de análisis basados en firmas, estadísticos y de anomalías para detectar posibles amenazas en la red.

Los tipos de sistemas o aplicaciones existentes enfocados en realizar análisis DPI pueden ser clasificados en IDS (Intrusion Detection System) e IPS (Intrusion Prevention System).

Los IDS son sistemas pasivos que solo generan alertas, mientras que los IPS toman medidas activas como cortar las conexiones sospechosas.

Ejemplos de este tipo de software IDS/IPS son SNORT, Suricata o Bro.

Aunque esta técnica de detección es útil tiene de varios problemas:

1. Falta de escalabilidad, en redes con gran tráfico la cantidad de información puede crear un cuello de botella si las reglas del sistema no están bien definidas.

2. Alta tasa de falsos positivos.

3. Dificultades con el análisis de paquetes si las comunicaciones están cifradas, ya que no se podrían descifrar el contenido de los paquetes a analizar.

4. Legalmente pueden existir problemas al analizar se la información enviada en los paquetes.

A favor el análisis de paquetes tendría las siguientes ventajas:

1. Tiempo de reacción bajo, "simplemente" actualizando las reglas se puede proteger al sistema frente a nuevas amenazas.

2. Flexibilidad, el sistema es capaz de detectar amenazas no definidas si se descubre comportamientos anómalos.

3. Permite analizar ataques a posterior, si se registra el tráfico de la red.

Detección de botnets mediante análisis de flujo [13]:

Esta técnica se basa en analizar el tráfico desde un punto de vista más abstracto, sin entrar a analizar los datos propiamente dichos.

Básicamente, la idea es caracterizar las comunicaciones a partir de datos como direcciones IP de origen y destino, puertos de comunicación, volumen de tráfico transmitido o duración de las sesiones. Con todos estos datos, el sistema de análisis de flujo de red puede detectar comportamientos estadísticamente anómalos o patrones de comportamiento mediante los cuales se detecten botnets u otro tipo de amenazas.

Para este tipo de detección se usan herramientas como Netflow (protocolo de red que se ha convertido en estándar de facto para el análisis de flujos de red en routers y switches).

El análisis de flujo de red puede detectar tanto amenazas desconocidas, al detectar comportamientos anómalos como por ejemplo una botnet no identificada, como amenazas previamente identificadas, como son IPs conocidas de los centros de C&C de una botnet.

Otra ventaja de esta técnica de análisis es la posibilidad de almacenar los reducidos metadatos de las comunicaciones, gracias a lo cual se pueden realizar análisis retrospectivos de amenazas o de ataques con el objeto estudiar la respuesta del sistema y mejorar las contramedidas existentes. Por el contrario, esta técnica genera una sobrecarga de tráfico considerable, ya que los dispositivos de enrutamiento envían los metadatos de las comunicaciones al servidor encargado de almacenar y procesar esta información.

Por supuesto, esta técnica de detección es fácilmente evadible si es la única medida que se aplica para detectar botnets. Existen técnicas, como la del algoritmo de generación de dominios (DGA) o covert channel (por ejemplo, con el de empleo de otro tipo de tráfico, como el ICMP), que son utilizadas para evitar su detección, intentando que las conexiones realizadas por tipo de amenazas pasen desapercibidas.

Por lo tanto, aunque útil, esta técnica no puede ser la única a ser usada en un sistema y deberia ser complementada con otras, como el análisis de paquetes, análisis de logs, análisis de DNS, etc.

Detección de botnets basada en DNS:

Ya se explicó todas las técnicas que emplean las botnets para evitar ser detectadas (DGA, fast flux, etc.), debido a ellas se producen unas características en el empleo a las llamadas a los dominios (a través de la traducción que proporciona el DNS) y son estas características las que se

emplean para la búsqueda de patrones botnet. A continuación, se muestran algunos ejemplos de técnicas de detección de Botnets basadas en DNS [12]:

1. Solicitudes DNS fallidas (NXDOMAIN): existen estudios que muestran que una forma de detectar amenazas potenciales de malware perteneciente a una botnet es el análisis estadístico de las peticiones fallidas de resolución de DNS al no estar registrados los dominios utilizados por las botnets como C&C. Botnets como Conficker o Torpig utilizan dominios con una alta entropía para evitar su posible detección, es decir que la utilización de cada uno de los dominios es poco probable, con lo que necesita un gran número de dominios para funcionar y muchos de estos pueden fallar en su resolución. Se buscan pues, masivas respuestas de petición a dominios que no existen.

2. Monitorización de dominios maliciosos: Consiste en monitorizar todas las peticiones realizadas al servidor DNS y comprobar que el dominio a resolver no está en ninguna lista negra (como las publicadas en DNSL o Realtime Blackhole List-RBL-). Estas listas son generadas por diferentes organizaciones como Spamhaus o SpamRats.

3. Dominios con bajos TTL's: Otro de los métodos usados por los creadores de botnets para dificultar su detección es la modificación de la IP asociada a un dominio, técnica conocida como fast-flux. De este modo al cambiar la IP de destino se dificulta la detección de anomalías. Para realizar este cambio, estos dominios tienen un TTL o tiempo de vida muy bajo, de este modo se fuerza a que los sistemas DNS a refrescar frecuentemente la caché de resolución de la IP asociada al dominio, o en caso de TTL nulo, ni siquiera almacenarla. Por lo tanto, aquellas peticiones DNS cuyo TTL sea muy bajo son sospechosas. Esta técnica puede crear falsos positivos, ya que existen sistemas legítimos conectados a Internet que utilizan este tipo de técnicas de ir cambiando la IP asociada a un dominio para balancear carga en sus sistemas (por ejemplo, Google).

4. Detección de tráfico DNS anormal: Existen estudios que analizan la detección de botnets en base al comportamiento anómalo de las peticiones DNS. Según estos estudios, algunas de las aproximaciones que se utilizan son:

    a) Búsqueda de nombres de dominio cuyas tasas de consulta sean anormalmente altas o que están temporalmente concentradas.

    b) Análisis de peticiones similares, tanto temporalmente analizando el tráfico DNS generado por una misma IP o analizando las peticiones generadas por IP's diferentes. El objetivo es la detección de patrones de comunicaciones.

    c) Peticiones a servidores DNS de países extranjeros puede ser un indicio para sospechar de la existencia de una botnet. Quitando los grandes servidores de DNS, como Google, que alguien de un país dado realice peticiones a servidores DNS de otros países puede ser un indicio de un comportamiento sospechoso.

d) Análisis de las direcciones que se solicitan para su resolución (DNS), analizando posibles patrones como por ejemplo porcentaje de caracteres numéricos (cuando se generan aleatoriamente como ya hemos comentado) o la inclusión de palabras reconocibles.

e) Análisis de uso de TCP como protocolo de transporte para consultas DNS, debido a que su uso suele ser residual (se suele utilizar UDP) puede ser un indicador de un comportamiento anómalo.

Empleo de honeypots:

Un honeypot es un recurso informático cuyo único objetivo es ser atacado. Así, un honeypot atacado e investigado puede proporcionar información valiosa sobre el ataque y su atacante. El empleo de honeypots para la investigación de botnets es habitual y es, también, la forma de descubrirlos, aprender sobre ellos y luego (si es posible) inhabilitarlos. Existen dos tipos de honeypots [18]:

- De baja interacción: A los honeypots de baja interacción se les permite una interacción "limitada" con su atacante. Todos los servicios de un honeypot de baja interacción son emulados. Esto significa que los honeypots de baja interacción no son vulnerables y no se infectarán con el exploit. Estos servicios emulados se disfrazan de software vulnerable o sistemas completos, fingiendo todo el diálogo de red a medida que avanza el ataque. El objetivo final es simplemente recolectar una muestra de malware descargado. Ejemplos de software Honeypots de baja interacción son Google Hack Honeypot, HoneyBOT, honeytrap, KFSensor, Multipot, Dionea, etc.

- Alta interacción: Los honeypots de alta interacción utilizan el servicio o software vulnerable real, controlando de cerca el sistema, ya que es realmente explotado por los atacantes. Esto tiene una ventaja sobre los honeypots de interacción bajos, ya que es posible obtener una imagen mucho más detallada de cómo exactamente progresa un ataque o cómo se comporta una muestra de malware particular en un sistema real.

  Además, los servicios emulados no se tiene la posibilidad de descubrir exploits previamente desconocidos. Por su propia naturaleza, sin embargo, es probable que los honeypots de alta interacción se infecten por sí mismos, lo que requiere la mayor atención por parte de los operadores para evitar que las consecuencias desastrosas se sigan propagando a sistemas remotos o incluso locales. Es por estas razones que las salvaguardas más estrictas deben construirse alrededor del honeypot en lo que respecta a las políticas de seguridad de la red. Ejemplos de estos honeypost de alta interacción son HoneyWall y Sebek.

Una vez que estas botnets han sido detectadas lo habitual es protegerse de ellas. Aquí entramos en lo que llamamos la protección e inhabilitación de botnets.

3.5.2 Técnicas de protección e inhabilitación de botnets

Después de reunir suficientes evidencias sobre la existencia de una botnet se deben de tomar medidas para inhabilitar nuestras máquinas afectadas por la botnet.

En caso de detección hay que ponerse en contacto con las agencias pertinentes (CERTs, ISP's, etc.) para lograr que la red botnet se cierre y que los actores cibercriminales sean detenidos. Además, es deseable que la información relativa a dichas botnets (DNS, topología, tipos de ataques, etc.) sea publicada en el mayor número de sitios que existen al efecto y que ya se han comentado (ShadowServer, dsshield, DNSL, RBL, Incibe, etc.)

Las técnicas de protección en nuestra red y de las máquinas infectadas seria la siguiente:

- Inicialmente TODAS las máquinas se consideran infectadas a la espera de la verificación que niegue esa suposición.
- Si se quiere investigar sobre el bot hay que realizar un análisis forense (Cuidado: podría servir como prueba en un hipotético juicio o demanda) utilizando Sand Box y aislando de la red de producción la máquinas zombis.
- Hay que valorar la posibilidad de formatear todas las máquinas y que esto acabe con el malware instalado (existen APT's capaces de permanecer en un dispositivo, aunque este sea formateado).
- No poner nada en producción hasta que se comprueba que el sistema y toda la red está libre de ese malware.
- En general:
  - Todos los hosts deben de tener actualizado su sistema operativo (incluyendo los parches correspondientes)
  - Todas las aplicaciones deberían de ser legales, oficiales y obviamente también actualizadas y parcheadas.
  - Lo mismo se puede decir de un antivirus y un firewall de host eficaz.
  - En la red es aconsejable disponer de sistemas IDS (como Snort) e IPS (como suricata) con reglas específicas para botnets como *emerging-botcc.rules* y *botnet-cnc-ru*.
  - Es deseable que la red tenga bien configurados sus dispositivos cortafuegos y sus capacidades IDS/IPS si las tienen.
  - Los usuarios no deben ejecutar contenido no confiable y menos con privilegios de administrador (si disponen de el).
  - Las comunicaciones de cualquier tipo deberían pasar a través de servicios proxies.
  - Solo deberían permitirse consultas a DNS internos o confiables.

  - Es una buena práctica disponer de un Honeypot (o varios) en la red empresarial y monitorizar la red en tiempo real.
  - Empleo de herramienats gratuitas (o de pago) de detección de botnets (por ejemplo, Bot Revolt, RuBotted, Bot Hunter, etc.).
  - Empleo de herramientas on-line de servicios antibotnet (como las de OSI-Incibe, Costant Guard, Microsoft Safety Scaner, etc.).
  - Etc.

Ademas de las anteriores y para los administradores de sistemas se recomienda:

- Usar un firewall para bloquear todas las conexiones entrantes de Internet a servicios que no deberían estar disponibles al público (WhiteList).
- Aplicar una política de contraseñas.
- Asegúrarse de que los programas y usuarios del equipo usen el nivel más bajo de privilegios necesarios para completar una tarea.
- Desactivar la reproducción automática para evitar el inicio automático de archivos ejecutables en la red y de las unidades extraíbles (USB's).
- Desactivar el uso compartido de archivos si no es necesario. Si se requiere compartir archivos, usar ACL's y protección con contraseña para limitar el acceso. Deshabilitar el acceso anónimo a las carpetas compartidas. Conceder acceso únicamente a cuentas de usuario con contraseñas seguras a carpetas que deben compartirse.
- Desconectar y anular los servicios innecesarios.
- Si un malware infecta un equipo de la red interna, deshabilitar o bloquear el acceso a los servicios desde este equipo hasta que se limpie.
- Mantener un DNS interno o confiable y no permitir anadir nuevas URL's a HOSTS.
- Configurar el servidor de correo electrónico para bloquear o eliminar el correo electrónico que contenga archivos adjuntos que se usan comúnmente para propagar amenazas, como archivos .vbs, .bat, .exe, .pif y .scr.
- Capacitar a los empleados para que no abran archivos adjuntos a menos que los estén esperando y que no ejecute software que se descargue de internet a menos que haya sido escaneado en busca de virus. Simplemente visitar un sitio web comprometido puede causar infección si ciertas vulnerabilidades del navegador no estan parcheadas.
- Desactivar el bluetooth cuando no sea necesario.

- Mantener nuestra red local con el software de los equipos de red (router, switches, firewall, etc.) actualizados.
- Tener activados IDS, IPS y otro tipo de medidas con reglas de detección precisas y actualizadas.

Por último, y aunque las técnicas de inhabilitación de las botnets pueden ser muy diversa y estas suelen involucrar a varias organizaciones, diferentes organismos y gobiernos, se exponen formas de desmantelar o al menos inutilizar una botnet.

Si tenemos estas sospechas, debemos emplear las siguientes medidas orientadas a la limpieza de los equipos (en algunos casos es aconsejable desconectarnos de la red):

- Realizar una búsqueda o escaneo a través de aplicaciones en búsqueda de artefactos maliciosos.
- Programar la acción de un escaneo del antivirus antes del arranque del S.O.
- Empleo de herramientas online que buscan conexiones “sospechosas” o a URL “maliciosas como podria ser la del servicio Antibotnet del INCIBE [33] que incluso dispone de un plug-in para navegadores.
- Empleo de herramientas de desinfección locales como HitmanPro.
- Empleo de herramientas para la desinfección especifica de Zeus como ZbotKiller (Kaspersky) o SpyHunter Anti-Malware Tool.
- Si es posible, restaura el equipo a una configuración segura anterior a través de un Backup o una aplicación que guarde este tipo de configuraciones.
- Realizacion de un análisis forense (muy técnico)
- Desinfeccion manual de los equipos (muy técnico). Aislar las computadoras comprometidas para evitar que las amenazas se propaguen aún más. Realizar un análisis forense y restaurar las computadoras utilizando medios confiables.
- Crear reglas específicas para IDS/IPS una vez analizado el malware, su código (si esto es posible) y su comunicación.
- Difundir lo más posible sus direcciones DNS de C&C y la botnet en si para que sea conocida (cuanto más mejor). Inhabilitar los alojamientos Web a través de sus webhosting o su IPS.
- Cerrar el tráfico a determinadas direcciones IP o dominios (listas negras en routers o servidores proxy).

- Infectar al propia botnet para inutilizarla (haciéndola desconectarse de sus peers o tomando el control de sus servidores C&C). A esta técnica del ataque a botnets P2P se la denomina "peer poisoning".

Hay que tener en cuenta que existen avisos de infección NO reales que nos invitan a ejecutar algo que nos limpiará el equipo. Esto habitualmente nos descarga y nos instala un malware. Las botnets como Zeus quieren pasar desapercibidas, no llamar la atención para poder seguir operando a través de esa máquina sin que el usuario lo perciba. En esto estriba la gran dificultad de su detección.

Existen iniciativas de detección de malware tipo botnet para (trabajando de forma conjunta) poder bloquear servidores C&C o nodos P2P con el fin de deshabilitar la red. Estas iniciativas se basa en:

- Control del tráfico con grandes volúmenes a servidores inusuales.
- Generacion de listas de URLs o dominios maliciosos.
- Gestion de avisos de infección entre organizaciones y usuarios.
- Creacion de reglas especificas ante malware conocidos (como Snort).

Peer poisoning

Casi todas las redes P2P tienen una vulnerabilidad en el mecanismo de intercambio entre peers. Como ya se explicó anteriormente, los nodos deben mantener y compartir una lista de otros nodos con los trabajadores ("workers") para, así, poder distribuir a los trabajadores entre la gran cantidad de nodos. Cuando se identifica un nuevo robot como nodo (capaz de aceptar conexiones entrantes), el nodo al que está conectado lo agrega a la lista de nodos y lo comparte con los otros nodos.

El peer poissoning consiste en introducir un equipo en la red que sea capaz de convertirse en "nodo" de esa botnet y a partir de ahí distribuir listas de IP's NO validas tanto a sus "workers" como a los otros nodos.

Si se introducen los suficientes nodos "contaminados" (poisoning) podrían llegar a separar a los trabajadores de los nodos legítimos. Al solo emitir las direcciones IP de otros nodos contaminados, aumenta la posibilidad de que los trabajadores solo conozcan los nodos contaminados y disminuyan significativamente las posibilidades de que se reincorporen a la red. Este ataque puede no ser definitivo, pero si podría paralizar partes grandes de la red.

3.6 Conclusiones del capítulo

Después de lo analizado hasta ahora han quedado claros los siguientes conceptos relativos a las botnets:

- Existe mucha reutilización de código o refinamiento de las botnets ya creadas para la creación de nuevas botnets.
- Las técnicas de comunicación son la parte de la creación más compleja.
- La subsistencia de una botnet suele deberse a las formas de comunicación que emplean, ofuscación, ocultación y su topología.
- Las botnets, en general, provienen de un código "troyano" que establece un canal de comunicaciones con un servidor C&C, peers-to-peers (P2P), métodos híbridos o empleo de redes sociales o similares.
- Las botnets pueden ser adaptadas a diferentes usos modificando sus payloads.
- En casi todos los casos se busca un beneficio personal, estatal o empresarial.
- Nadie "regala nada", si se liberan códigos de botnets suele ser para acabar con la competencia entre cibercriminales.
- El desarrollar una botnet activa tiene un coste elevado (en conocimiento, tiempo y a veces en material) y el pago por el uso de sus "servicios" se puede obtener a través de internet si tan siquiera conocer las partes implicadas.
- Las víctimas no suelen percibir el contagio de su máquina.
- Organismos, gobiernos y empresas de seguridad mejoran sus herramientas, procedimientos y protocolos estableciendo fórmulas de colaboración debido a la desaparición de fronteras en la comisión de delitos.
- La identificación y detención final de las personas implicadas suele ser difícil debido al carácter transnacional de los delitos y la ocultación de estos entre las ingentes máquinas de su red.
- Las medidas de protección para evitar ser infectado por un malware tipo botnet a nivel de host no son difíciles de implantar.
- Las medidas de protección a nivel de red son más complejas y requieren de personal específico y experto en seguridad.
- La comunidad internacional y las empresas privadas cada día trabaja más conjuntamente para la detección, alerta, análisis y eliminación de este tipo de amenazas. Este tipo de redes representan unas pérdidas (económicas, en credibilidad, en recursos, etc.) y unos trastornos enormes una vez que están activas.

Nota: Siempre se puede realizar una investigación para poder intentar arrestar a los cibercriminales o al BotMaster antes de desmantelar la botnet (depende de la estrategia que las autoridades quieran seguir).

CHAPTER 4

EXPERIMENTACIÓN PRÁCTICA

La verdad es omnipresente, pero lo puede reconocer solamente el que lo busca. Nicolae Iorga, 1871-1940, Historiador Rumano

4.1 Diseño del escenario práctico

Con el objetivo de obtener un conocimiento más profundo de las botnets y de su implementación en particular, se ha propuesto desarrollar un laboratorio en el que observar, verificar, comprobar y ejecutar un escenario que simule al que pudiera darse en la vida real con las limitaciones inherentes al tiempo, elementos y herramientas empleadas en él.

En este escenario se ha primado la funcionalidad por encima de la flexibilidad. Entendiendo la primera como un desarrollo con resultados empíricos comprobables, repetibles, contrastados y verificados. El detrimento de la flexibilidad se produce debido a la falta de tiempo y medios en poder establecer diferentes laboratorios con aproximaciones cada vez más reales a una red simulada con los elementos necesarios en su similitud a internet. Así pues, el método práctico se basa en:

1. Establecer el laboratorio inicial.
2. Implementar una red botnet suficiente para poder desarrollar las pruebas.
3. Emplear métodos diversos de ataques a través de dicha botnet.
4. Emplear algún método alternativo de ataque con la botnet desplegada.
5. Realizar un análisis de los equipos infectados y realizar su limpieza sacándoles de la red botnet.

6. Obtener unas conclusiones extraídas del laboratorio y de futuros trabajos relacionados.

4.2 Elección de la Botnet

A lo largo de los anteriores capítulos de este trabajo se ha podido conocer la amenaza relacionada con las botnets y la complejidad no solo en su detección e inhabilitación, sino en el desarrollo de una red tan compleja como estas.

En el caso particular de las botnets, es difícil encontrar herramientas preparadas para el desarrollo de un laboratorio práctico, que si son fáciles de obtener en otros campos relativos a la seguridad informática. Esto es debido a la naturaleza propia de este tipo de amenazas.

Mientras que un malware puede ser desarrollado para aprovechar una determinada vulnerabilidad, una botnet se emplea para obtener un "rédito" de cualquier tipo de ataque o exploit conocido. Digamos que una botnet es "un medio" y no un malware en sí.

Si bien, hoy en día las botnets están muy especializadas (en ataques particulares) no es menos cierto que pueden readaptarse para poder cumplir con cualquier otro tipo de técnica maliciosa.

Existen proyectos como UFONet [38] que nos pone a disposición una herramienta de software gratuita diseñada para probar ataques DDoS contra un objetivo y utilizando una redirección a través de sitios web. Sin embargo, esto no deja de ser similar a herramientas como Davoset que utiliza los mismos métodos pero que no se pueden definir como una botnet "perse" aunque su funcionamiento a nivel de "distribución" del ataque si es similar a una botnet.

La diferencia de ambas herramientas con una botnet "al uso" es que la botnet tiene a su disposición máquinas zombis infectadas con bots que ejecutan sus órdenes y que "entre otras cosas" pueden realizar un ataque DDoS. Las anteriores herramientas, sin embargo, se aprovechan de un tipo de servicio o vulnerabilidad para emplear a máquinas intermedias a modo de bots, pero sin ser controladas nunca en su totalidad por el BotMaster. Porque una botnet es realmente UN EJERCITO DE ZOMBIS a nuestra disposición para poder hacer aquello que nosotros queramos con el.

Por todo ello, el código o los ejecutables de las botnets no son fáciles de obtener y emplear en trabajos académicos, ya que pueden ser "reacondicionadas" o actualizadas y empleadas con otros propósitos volviendo a ser útiles (para el BotMaster) y no es habitual que se regalen este tipo de capacidades fácilmente.

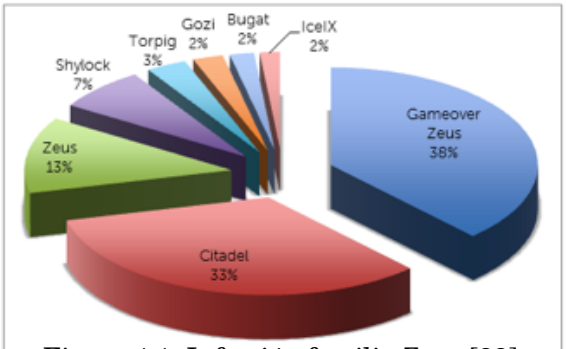


Figure 4.1: Infección familia Zeus [32].

Nos encontramos pues con el problema de que no existen muchas herramientas conocidas que impliquen todos los elementos de una botnet (bots y servidores C&C) para poder ser

empleadas en labores de estudio.

Después de una labor de investigación en la red, me he decantado por el empleo de una botnet llamada Zeus (también llamada ZeuS o Zbot) de la que existen infinidad de versiones adaptadas a diferentes escenarios y/o ataques. He elegido Zeus 2.1.0.1 por los siguientes motivos:

- Zeus es una botnet que apareció en 2007 y sigue activa en una o varias de sus variantes, lo que demuestra su efectividad.
- Existen multitud de variantes con funciones más específicas, pero la versión que nosotros usaremos tiene unas funciones básicas muy avanzadas para una botnet.
- Los casos de infección de botnets por la familia de herramientas Zeus (Citadel, GameOver..etc.) es enorme en España (ver imagen 4.1).
- Sus objetivos suelen ser desde bancos a ordenadores personales con un gran daño a nivel económico a través de sus acciones.
- La versión 2.1.0.1 fué "filtrada" en el 2011 lo que no la convierte en muy antigua teniendo en cuenta que afecta a equipos hasta Windows 7.
- Su arquitectura es la de Servidores C&C, lo que implica un mejor entendimiento de su funcionamiento a nivel educativo.
- Emplea el protocolo de comunicación http (puerto 80) ampliamente aceptado y permitido por dispositivos de seguridad.
- Las versiones de Zeus (como GameOver) tiene topologías que combinan (DGA, P2P y C&C) aunque en este laboratorio solo se utiliza esta última.
- La versión (2.1.0.1) que se ha obtenido es estable y funciona adecuadamente no como otras probadas.
- La interface de gestión es muy visual, fácil de aprender y los resultados de los ataques están muy bien representados en ella.

4.3 Características de Zeus

4.3.1 Historial de Zeus botnet

El Virus Troyano Zeus es muy conocido actualmente al ser utilizado como via de envío del malware de cifrado CryptoLocker que tanto ha salido en la prensa internacional.

A continuación, se expone una breve historia de este malware desde su primera aparición hasta el año 2017 [44].

- 2007 - Zeus, su primera aparición.

  El año 2007 marcó el comienzo del malware Zeus. Incialmente se utilizó para robar información del Departamento de Transporte de los Estados Unidos para, a continuación, convertirse en una de las botnets más utilizada mundialmente.

- 2009 - Zeus se expande.

  En 2009, Zeus comenzó su expansión masiva por la red y ya se empezó a considerara como una gran amenaza. Había comprometido más de 74.000 cuentas de FTP en sitios web de grandes compañías como el Banco de América, la NASA, Monster.com, ABC, Oracle, Play.com, Cisco, Amazon y BusinessWeek entre otros.

- 2010 - Zeus, amenaza mundial.

  Este fue el año en que el FBI anunció que el virus Zeus se utilizó para infectar computadoras en todo el mundo. En esta época Zeus era usado principalmente como un troyano bancario, para robar información mediante el registro de las pulsaciones de teclado realizadas al utilizar un navegador o mediante la inyección de formularios web. Además, se usaron correos electrónicos para hacerla llegar al máximo numero de personas posibles.

- 2013 - Zeus, el mejor “amigo” de CryptoLocker.

  A finales de 2011 se establecieron nuevas capacidades a Zeus, apareciendo GameOver especializado en delitos bancarios. Su evolución permitio que la red zombi fuese menos vulnerable a los intentos de eliminación ya que empezó a utilizarse un sistema cifrado simétrico para las comunicaciones con sus servidores C&C (Command & Control). En 2013, todo se intensificó cuando GameOver Zeus fue responsable de la distribución masiva del virus CryptoLocker (ransomware) por todo el mundo.

- 2015 - Zeus evoluciona hacia Chthonic.

  2015 también fue un año importante para Zeus, ya que evolucionó en Chthonic, una nueva variante descubierta por los investigadores de malware de Kaspersky. Se emplea la misma técnica de cifrado utilizada ya por los troyanos "Zeus V2" y "Zeus AES". Chthonic es conocido por haber atacado a más de 150 bancos en 15 países, en 2015.

- 2016 - Zeus y TOR.

  En agosto de 2015, se utilizó el código de Zeus para crear una variante personalizada para él, llamada "Sphinx Banking Trojan", que se vendia en el mercado negro por unos 500 dolares. La red TOR fue utilizada para ello. Los mas afectados fueron principalmente los bancos brasileños. No obstante, ya se conocian usos de Zeus con redireccionamiento a través de la red TOR donde estarian alojados los servidores C&C y que, además, funcionarían en versión de 64 bits.

- 2017 - Zeus Trojan.

  Zeus sigue siendo una de las infecciones informáticas más grandes, y sus muchas variantes siguen aún extendiéndose.

4.3.2 Formas de distribución de Zeus

El malware Zeus ha empleado muchas formas de distribución a lo largo de los años para todas sus variantes. Una de las primeras formas para que llegue a un sistema informático fue con la ayuda de redireccionamientos web (drive by download o descarga directa). Se puede activar el redireccionamiento ejecutando cualquier elemento en línea como un enlace o un botón.

Uno de los redireccionamientos contiene un script que descarga el malware. Otra forma de descargar es hacer clic en un anuncio que nos plantea una pregunta y nos promete un premio al contestarla.

Los correos electrónicos no deseados con archivos adjuntos eran y siguen siendo una forma destacada de distribución para Zeus. Dentro del archivo adjunto hay un archivo ejecutable que está ofuscado y oculto como otro tipo de archivo de documento, con una extensión como .pdf o .doc

4.3.3 Precio de mercado

La botnet Zeus es una herramienta criminal para desarrollar troyanos (bots) personalizados y difíciles de detectar. Debido a las muchas capacidades diferentes que se pueden incluir en los troyanos (bots) y como estos ejecutables pueden evolucionarse y actualizarse en las máquinas zombis es muy apreciado y con un largo futuro en el mundo cibercriminal. Zeus se podría decir que se basa en modelo de Software-as-a-Service (SaaS) donde los cibercriminales pueden emplear, comprar o alquilar los servicios de dicha “nube”.

Zeus se ofrece también con la posibilidad de que un cibercriminal emita comandos específicos, como descartar una pieza de código malicioso de un tercero o utilizar la geolocalización para afectar únicamente a determinados países, regiones o ciudades. También permite que el cibercriminal establezca intervalos para la comunicación de C&C, lo que puede reducir la carga en la infraestructura de C&C y dificultar la detección de la comunicación maliciosa.

La versión básica de Zeus puede ser adaptada con funciones específicas que se venden a día de hoy por entre 700 y 15.000 dolares [27].

Como se ha comentado, se puede disponer de la botnet Zeus básica y luego ir añadiendo complementos compatibles como módulos separados, que se pueden comprar de forma individual. Los precios aproximados de estos módulos son:

- Módulo DDoS: Precio de 350 dolares. El número de tareas/objetivos es ilimitado, al igual que el número de subprocesos, el intervalo entre el envío de paquetes y el tamaño real del paquete. El módulo admite HTTP (GET), UDP e ICMP tipo de técnicas de inundación,

además de que permite que el cibercriminal lo use para cambiar estas configuraciones sobre la marcha.

- Módulo Socks 4/5: Precio de 120 dolares. El complemento permite al cibercriminal ocultar la botnet, para convertir fácilmente los hosts infectados por malware en proxies de anonimización, un módulo bastante común que se encuentra en muchos bots de malware de la competencia. El autor del bot también permite a sus clientes especificar el puerto del servidor Sock o el bot elegirá uno al azar.

- Módulo modificador de archivos HOSTS: Precio de 50 dolares. El complemento permite la modificación de dichos archivos.

- Módulo de Back Connect Hosts: Precio de 380 dolares. Es otro plugin estándar disponible en bots de malware de la competencia, que permite a los ciberdelincuentes conectarse y abusar de hosts detrás de un NAT.

4.3.4 Caracteísticas de Zeus

El bot Zeus dispone de muchas características de ataque, pero las más conocida son la de robar información bancaria mediante el registro del teclado del navegador y la inyección en formularios web. Zeus es básicamente un proxy/troyano que utiliza técnicas MITM (Man In The Middle) en el propio sistema del usuario.

Zeus ataca explotando vulnerabilidades en la seguridad del navegador para modificar páginas web y manipular transacciones monetarias cambiando o agregando campos de la página web. Además, no necesitan privilegios de administrador para "secuestrar" una máquina basta con ejecutar el programa "bot.exe" que difunda el BotMaster.

Lo realmente peligroso es que no existe una forma de seguridad efectiva (como podría ser SSL) que pueda proteger dicha forma de ataque. La captura de formularios web es una técnica de captura de datos en formularios web antes de que se envíe la información al servidor..

La botnet dispone de un panel de control que se usa para monitorizar, actualizar los bots, enviar las ordenes y recoger la información de toda la botnet. Algunas de las características que posee esta botnet son:

- Captura de credenciales a través de HTTP, HTTPS, FTP, POP3.

- Robo de certificados de infraestructura de clave pública X.509.

- Proxy SOCKS integrado.

- Robo/Eliminación de cookies HTTP y flash.

- Captura de pantallas y scraping HTML o scraping web -web injection- (transformación de formularios web) en los sitios de destino (usuario).

- Modificación de archivos del host local (HOSTS).
- Agrupa los sistemas de usuarios infectados en diferentes botnets para distribuir los comandos y el control según sea el S.O.
- Capacidades de búsqueda que pueden usarse a través de un formulario web.
- Todos sus archivos en la máquina zombi se encuentran cifrados simétricamente (256-byte RC4). No se usan cifrados asimétricos porque, el uso de sofisticados algoritmos no tendria sentido, el cifrado solo necesita ocultar el tráfico y los archivos del bot y este es suficientemente robusto para los objetivos pretendidos.
- Comunicación entre bots y server C&C cifradas (256- byte RC4).
- Servidor C&C que puede realizar tareas adicionales como descarga y ejecución de archivos en los sistemas, actualización bots, etc.
- Tiene una cadena de identificación de bot única.
- Envío de información al servidor C&C automáticamente, como la versión del bot, el sistema operativo, la hora local, las ubicaciones geográficas, etc.
- Los programas bots (bot.exe) NO necesitan ser ejecutados con privilegios de administrador, lo que les convierte en todavía mas peligrosos.

De entre sus características de ataque se comentarán las más relevantes en cuanto a ex-filtración y robo de información [32].

Robo de credenciales en FTP y POP3.

El bot se conecta a las funciones de winsocks (Windows) que se emplean en los buffers y que es "case sensitive" a palabras/comandos como "user" o "pass", asociando esos "strings" encontrados hasta que se pulsa la tecla "enter". Estos socks "vigilados" por el bot están relacionados con servidores FTP y POP3 ya que vigila cualquiera de los comandos de este tipo de aplicaciones (CWD, PWD, TYPE, FEAT, PASV, STAT, o LIST) y así con los usuarios y contraseñas interceptados se construye una estructura que se envía al servidor C&C. Esta estructura seria protocol://user:password@nnn.nnn.nnn.nnn/, donde:

- "protocol" es POP3 si los comandos STAT o LIST son los encontrados (interceptados), en caso contrario es FTP.
- "user y password" son las credenciales introducidas con los comandos USER y PASS.
- "nnn.nnn.nnn.nnn" es la dirección IPv4 del host remoto y obtenida usando el comando "getpeername" en el socket interceptado (dicha función nos devuelve la IP del host remoto)

Captura de teclado y pantalla.

El bot vigila cualquier URL definida en su archivo de configuración y cuando es detectada envía la información que obtiene cuando la víctima pulsa el botón izquierdo del ratón (por si al teclear se confunde y corrige los datos), también manda una imagen del PC de la víctima al pulsar dicho botón. Además, es capaz de convertir los elementos marcados con un teclado virtual transformándolos a formato Unicode.

Robo de certificados.

Zeus tiene la habilidad de robar los certificados almacenados, incluidas las claves privadas al conectarse o interceptar la función PFXImportCertStore, esta función importa el archivo PFX BLOB (Los archivos PFX contienen claves privadas cifradas, certificados y otra información secreta. Un PFX BLOB es una representación binaria de PFX).

Inyección al Navegador web a través del secuestro (Hijacking) de sesión.

Una de las funciones principales de Zeus es la de robar sesiones bancarias via web. Para ello el bot vigila las webs especificadas en el archivo webinjects.txt, e inserta los campos que se le han indicado para que la víctima, ignorante de esta inyección de campos adicionales, complete esa información que será robada por el bot y enviada al server C&C. Este comportamiento está configurado dentro del archivo *config.bin* de la botnet.

Robo de Cookies

Zeus llama a la función *InternetGetCookie* para obtener las cookies de cada URL. Esta información se almacena para luego ser enviada.

Además de todo lo anterior Zeus dispone de comandos para dar órdenes a sus bots (o a todos o a los que se designe). Cuando el comando es recibido por el bot, este utiliza el numero del comando como parámetro para enviar un “pipe-command”. Un “pipe-command” no es más que un sistema de sincronización entre procesos para poder ejecutar los comandos que le solicita el BotMaster.

Una importante característica asociada a los comandos de Zeus es la de poder descargarse y ejecutar archivos en la máquina remota, lo que le concede una flexibilidad de nuevos ataques.

La funcionalidad del rootkit que se utiliza en este trabajo es compatible con Windows 2003/XP/7, pero no es compatible con sistemas de x64 bits. No nos ha sido posible encontrar una versión de 64 bits para poder ser utilizada.

Con respecto a lo anterior es conveniente resaltar que existe una forma de contaminación web que detecta la versión del navegador utilizado (x86 o x64) para descargar una u otra versión e infectar adecuadamente dichos dispositivos. Estas infecciones ya se ejecutan en cualquier tipo de plataforma Windows, desde 2003 a 2017.

4.4 Escenario Planteado

Una vez seleccionada la botnet para la realización del laboratorio se debe de establecer un escenario para que se pueda desplegar dicha red en él. Con motivo de la falta de una red física donde implementar el escenario se ha decidido su despliegue en máquinas virtuales con las siguientes características:

- Maquina anfitriona con S.O Windows 10 (Intel i7 16 Gb RAM)
- Capa de virtualización de un hipervisor de tipo 2 (sobre un sistema operativo de la máquina anfitrión Windows 10) VMWare Pro 12.
- Seis máquinas virtuales Windows 7 (de diferentes distros) con los siguientes roles:
  - 1 Servidor C&C.
  - 3 Maquinas zombits (bots) -mas adelante 4-.
  - 1 Maquina víctima.
- Una sola red interna en la que se encuentran todas las máquinas conectadas y aisladas de la máquina anfitriona y de la red externa (internet).

La disposición del laboratorio se eligió de esta manera por motivos técnicos y de seguridad que se comentan a continuación.

- La elección de un hipervisor de tipo 2 se debe al empleo del propio material del alumno que tenía disponible en su hogar y este ya contaba con un S.O anfitrión (Windows 10). Por otro lado, se eligió VMWare Pro 12 por los siguientes motivos:
  - Existe una gran facilidad de obtener máquinas virtuales VMWare oficiales de la web de Microsoft.
  - La versión Pro proporciona una funcionalidad completa con varias máquinas y una gestión de red entre ellas eficiente.
- Aunque se disponen de 16 Gb de RAM, el consumo de las máquinas virtuales junto con el de la propia máquina anfitriona es una sobrecarga alta para una virtualización de este tipo (hypervisor de tipo 2). Además, el empleo de equipos virtuales con un S.O actual (Windows 7 todavía cuenta con soporte Microsoft) requiere de una memoria RAM dedicada individual de cada dispositivo de entre 1 a 2 Gb y en el caso del Servidor C&C de entre 2 y 3 Gb. Estas características hacen que se retraigan entre 8 y 10 Gb de RAM de la máquina anfitriona. Por todo ello el laboratorio solo dispuso (en total) de seis máquinas virtuales corriendo a la vez.

- Se valoró la instalación de una máquina virtual adicional que actuara como BotMaster para que este accediese remotamente al (o los) servidor C&C. Sin embargo, debido a la limitación de recursos se ha actuado desde el propio servidor C&C de manera local.
- La elección de una red aislada fue debido a la "peligrosidad" de este tipo de malware ya que cuando se utiliza no puede estar seguro que no sea una versión con un troyano oculto que intente realizar una infección a través de la red local o de movimientos laterales entre máquinas.
- También se valoró la implementación de un firewall ASA virtual para dividir la red interna entre subredes a través de un dispositivo de protección perimetral avanzado para poder introducir un elemento más acorde a un escenario real. Esta idea se descartó por los siguientes motivos:
  - El dispositivo (ASAv) necesita otra máquina virtual adicional en la red de configuración "management" del CISCO ASAv con lo que en realidad el consumo de memoria RAM seria de dos máquinas virtuales adicionales, y la máquina anfitriona no disponía de muchos más recursos.
  - Las máquinas virtuales ya empleadas en el escenario mas sus "snapshots" consumían muchos recursos de disco duro (mas de 130 Gb) siendo ya critico, también el espacio en disco duro.
  - El ASAv es un dispositivo sin licencia (no existe una versión freeware) y no se quería hacer uso de elementos "crackeados" para este trabajo.
  - La falta de tiempo imposibilito el desarrollo de este escenario o de una alternativa freeware.

El esquema del laboratorio creados e puede ver en la imagen 4.2.

Nota: Es de reseñar que mas adelante se ha añadido otra máquina virtual (zombi) más a nuestra práctica con el objetivo de realizar ataques DDoS más efectivos.

4.5 Desarrollo del escenario

Teniendo ya claras las herramientas y el escenario a implementar se describirán los pasos realizados y la problemática encontrada en cada uno de ellos.

4.5.1 Obtención de los archivos botnet Zeus

Inicialmente se descargó la versión de Zeus 1.8.9.0 pero después de varios días de trabajo con esa versión se observó que el servidor C&C tenía problemas de conectividad con los bots y dejaba de funcionar el servidor C&C. Finalmente opté por buscar otra versión (Zeus 2.1.0.1) que funcionó correctamente durante todo el trabajo.

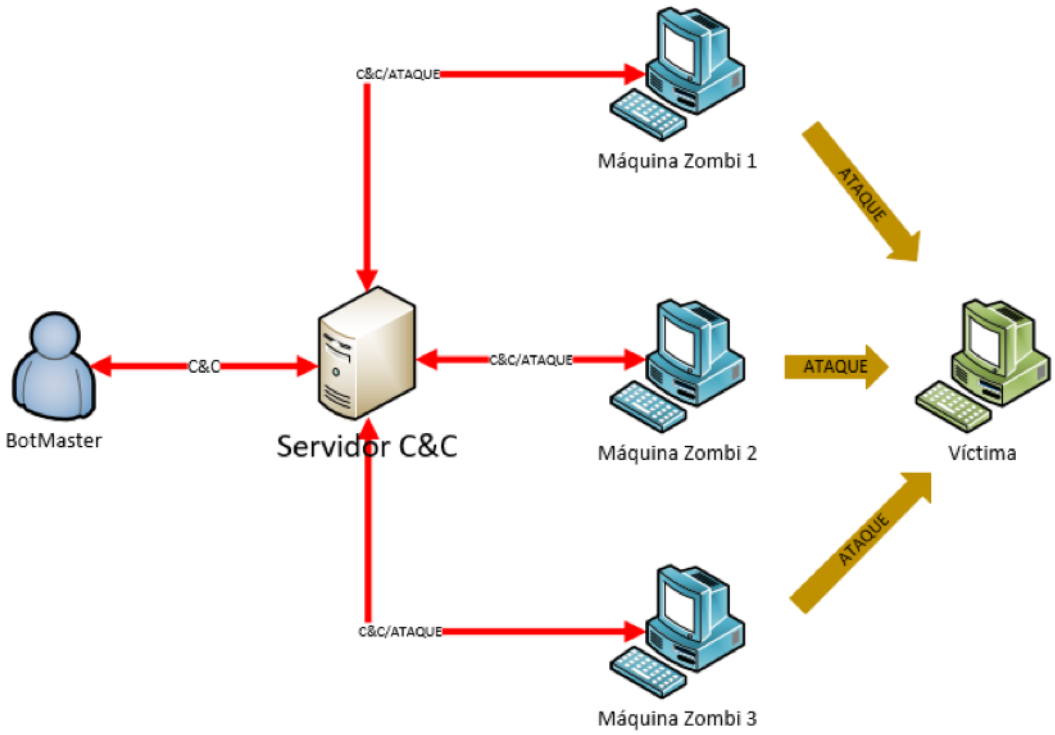


Figure 4.2: Arquitectura del laboratorio práctico.

Encontrar esta versión no fué fácil y tuve que registrarme en algunas webs de hacking para poder descargarla.

Los números en la versión significan lo siguiente (a.b.c.d):

- a – > Un cambio completo en el dispositivo bot.
- b – > Cambios importantes que causan incompatibilidad total o parcial con versiones anteriores.
- c – > Corrección de errores, mejoras y funciones adicionales.
- d – > Problema de limpieza de antivirus para la versión actual a.b.c.

4.5.2 Instalación del Servidor C&C y del bot Zeus

El funcionamiento del servidor C&C requiere del uso de una base de datos (BBDD) asociada a la herramienta C&C. Esta herramienta se ejecuta como un servidor web (HTTP) que se conecta con una BBDD para ir guardando toda la información de los bots así como la información necesaria para su gestión.

La gestión del servidor C&C se hace vía explorador web que se puede ejecutar en la propia máquina (por el BotMaster) o desde otro equipo.

Obviamente se necesita de un servicio web a parte del de la BBDD. Por todo ello se utilizó una versión de XAMPP que dispone de las siguientes funciones (Apache, Tomcat, MySQL, Mercury y FileZilla) y que me facilitó mucho el despliegue de todos los servicios en un solo producto.

4.5.2.1 Creacción de la BBDD

La versión de XAMPP descargada inicialmente era la 7.1.10 que disponía de la versión de PHP 7.1.10 y que dió problemas de instalación. Después de analizar el código fuente de Zeus 2.1.0.1 se observa que utiliza instrucciones PHP como "mysql_connect()".

Este tipo de instrucciones fueron consideradas "deprecated" a partir de PHP 5.5 y eliminada a partir de la versión 7. Finalmente se optó por emplear la versión de XAMPP/PHP 5.6.31.

Después de realizar la instalación, se arrancaron los servicios "Apache" y "MySQL". Se accedió a phpMyAdmin que sirve para gestionar la BBDD y se creó una nueva BBDD llamada "zeusdb" en la que el usuario "root" (se dejó sin password) tenía todos los privilegios sobre ella (aunque esto es muy poco seguro hay que pensar que es un entorno de estudio y aislado).

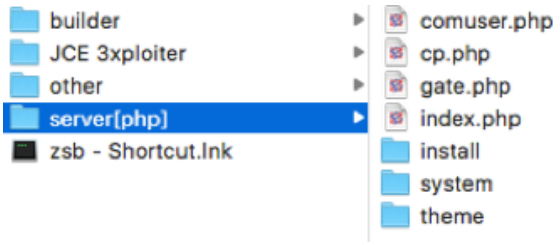


Figure 4.3: Archivos Zeus 2.1.0.1.

Además, también se observó que los comentarios del código aparecen en ruso, lo que indica la procedencia de, al menos, esta versión. Hay que tener en cuenta que algunos programadores hacen esto a propósito para despistar en cuanto al origen del programa.

4.5.2.2 Instalación del server C&C

Dentro de la carpeta de Zeus 2.1.0.1 accedemos a la carpeta llamada server[php] (ver imagen 4.3) y la copiamos en la carpeta de XAMPP *C:\xampp\htdocs* para poder acceder a ella vía web.

Inicialmente accedemos al path *127.0.0.1/server[php]/install/index.php* donde nos aparece la pantalla de la imagen 4.4 con la instalación inicial del servidor C&C [41]. Aquí, debemos completar los siguientes apartados:

- User name – > Usuario para poder acceder al control de los bots.
- Password – > Contraseña para ese usuario.
- Host – > Se refiere al host que albergará la BBDD (pudiera estar en otro equipo para proteger la información).
- User – > Usuario para acceder a esa BBDD ("root" por defecto).
- Password – > Del usuario con privilegios sobre la BBDD.

- Database – > Nombre de la BBDD (“zeusdb” en nuestro caso).
- Reports – > Carpeta donde se guardarán los datos (también) de manera local.
- Online bot timeout – > se refiere al tiempo en que un bot se le considera no conectado si no establece comunicación durante ese tiempo.
- Encrytion key – > Importante parámetro porque es en el que se basan las comunicaciones entre el bot y el servidor C&C, ya que están cifradas con esa clave. Cuando creemos el bot hay que introducirle la misma clave en “config.txt”. El cifrado es del tipo 256- byte RC4.
- Enable write reports to database – > Habilita el almacenamiento de los informes recibidos de los bots en la base de datos.
- Enable write reports to local path – > Habilita el almacenamiento de los informes recibidos de los bots en el archivo local especificado anteriormente.

Pulsando el botón “install” se produce la creación de la estructura completa en la BBDD y la generación necesaria de los parámetros en los archivos que se emplean en la comunicación de los bots.

La gestión de la BBDD es crucial, ya que en ella se establecen las tablas y las estructuras que permiten la comunicación con los bots. Con esto ya ha quedado preparada la BBDD que usará el servidor C&C vía web como observamos en la imagen 4.5.

Para el acceso al panel de control nos dirigimos a al path *127.0.0.1/server[php]/cp.php* donde nos pide el usuario y contraseña para poder gestionar el panel de control.

En la imagen 4.6 observamos el panel de control (CP) y todo el menú e información que nos ofrece.

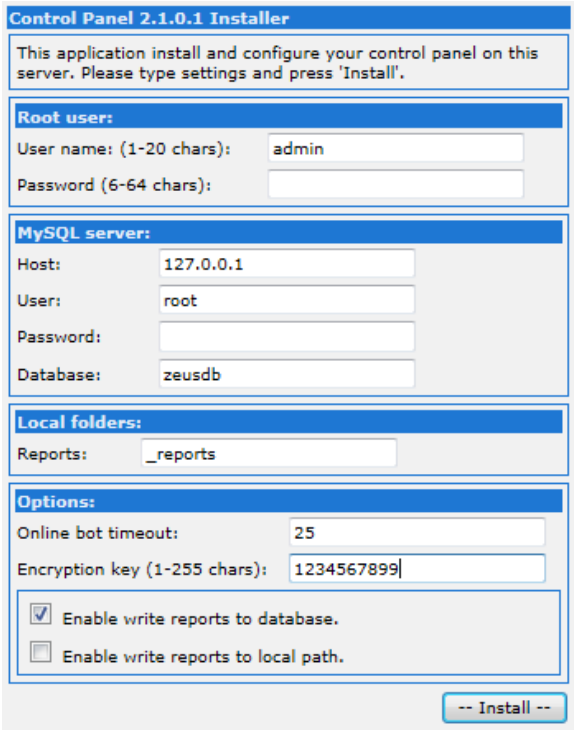


Figure 4.4: Pantalla de instalación.

4.5.2.3 Creacción de los bots

El siguiente paso es la creación de una ejecutable (.exe) para poder crear nuestros bots que “infecten” las máquinas de nuestra botnet [40].

La creación de este archivo ejecutable se realiza con una aplicación creada para ello y que se encuentra en la carpeta *Zeus_2.1.0.1/builder/zsb.exe*.

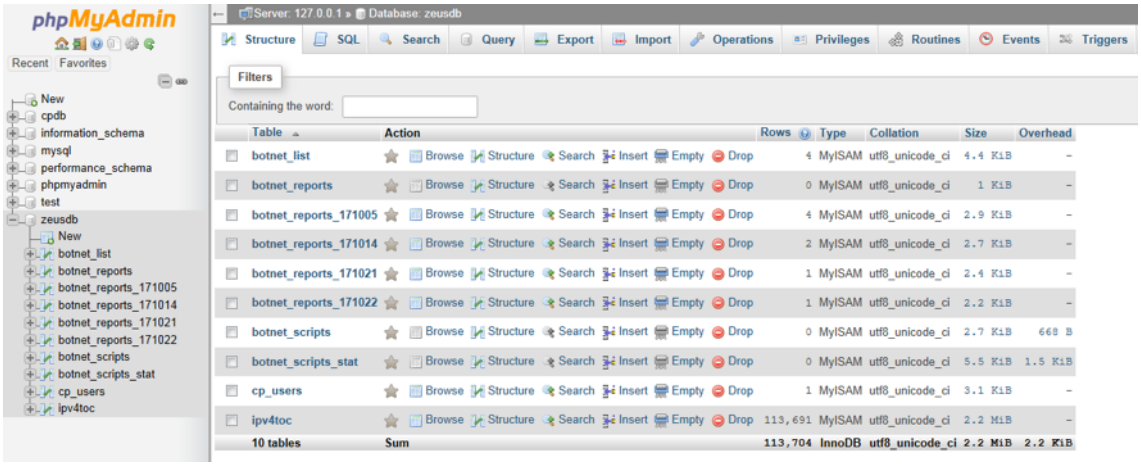


Figure 4.5: Base de Datos de Zeus instalada.

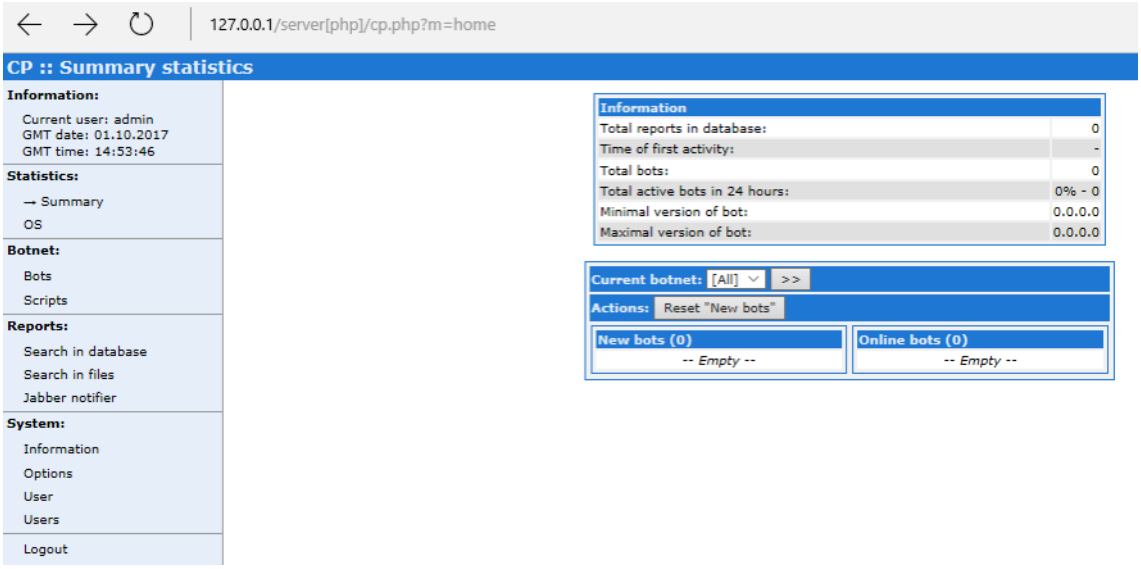


Figure 4.6: Panel de Control (CP) de Zeus.

Sin embargo, antes debemos de configurar los parámetros que tendrán nuestros bots modificando el archivo “config.txt” de esa misma carpeta.

Según se ve en la imagen 4.7 tenemos dos tipos de configuraciones (Static y Dynamic) y necesitamos, al menos, modificar los siguientes parámetros [7]:

- url_config – > http//192.168.99.1/Zeus_2/config.bin. Hacemos referencia al binario que se creará al crearse el ejecutable del bot. La IP hace referencia (en este caso) donde esta alojado nuestro servidor C&C.

- encryption_key – > 1234567899. Debe de coincidir con la misma clave que utilizamos en el servidor C&C ya que la comunicación en toda la botnet va cifrada con ella.

- url_loader – > http//192.168.99.1/Zeus_2/bot.exe. Este path le indica al bot que si necesita "actualizarse" (piénsese que podríamos cambiar las claves de comunicación o utilizar otra versión de bot) es ahí donde debería descargarse la nueva versión.

- url_server – > http//192.168.99.1/Zeus_2/gate.php. Aquí se le indica con quien debe de comunicarse que "puerta de enlace" con el servidor tiene. Este archivo es a través del cual se comunican bot con el servidor C&C y viceversa.

- file_webinjects – > se hace referencia a donde se encuentran los patrones de las webs a buscar e inyectar.

- webfilters – > Este parámetro tiene dos propósitos:

  - Enumerar una lista de URL que se deben anotar o eliminar del registro, independientemente del tipo de solicitud (GET o POST). Si el primer carácter de la máscara es un "!", Entonces la coincidencia de la URL NO se producirá (por ejemplo, la máscara "! *" Prohibirá la entrada de cualquier URL, excepto las enumeradas antes)
  - La máscara URL que tenga el símbolo "@" al comienzo realizará capturas de pantalla a cada clic del botón izquierdo del ratón. Esto es muy útil para las webs que emplean teclado virtual.

Existen muchos más parámetros que se podrían configurar, pero quedaría fuera de la extensión de este trabajo explicarlos todos.

Una vez actualizado nuestro archivo config.txt ya podemos ejecutar el archivo "zsb.exe" (ver imagen 4.7). Nos dirigimos a la carpeta "Builder" y allí cargamos el archivo config.txt y a continuación ejecutamos los botones "Build the bot configuration" y "Build the bot executable". Esto nos generará dos archivos en el mismo path donde estaba config.txt, el archivo "bot.exe" y el archivo "config.bin".

El Bot-ID consta de dos partes: %name%_%number%, donde:

"name" - - > Nombre del equipo (resultado del comando GetComputerName).

"number" – > un número generado en base a algunos datos únicos del sistema operativo.

El archivo "bot.exe" es el que distribuiremos a las víctimas y ambos archivos (el "bot.exe" y el "config.bin") los guardaremos donde marcamos en el config.txt que en nuestro caso era en *XAMPP/htdocs/Zeus_2/bot.exe* y en *XAMPP/htdocs /Zeus_2/config.bin* respectivamente.

4.5.3 Distribución de los bots e infección de las víctimas

Ya se comentó anteriormente que las vías de distribución de malware eran extensas. En este caso se tendrán en cuenta dos particularidades:

- Para que el ordenador víctima quede infectado se debe de ejecutar el programa bot (bot.exe).

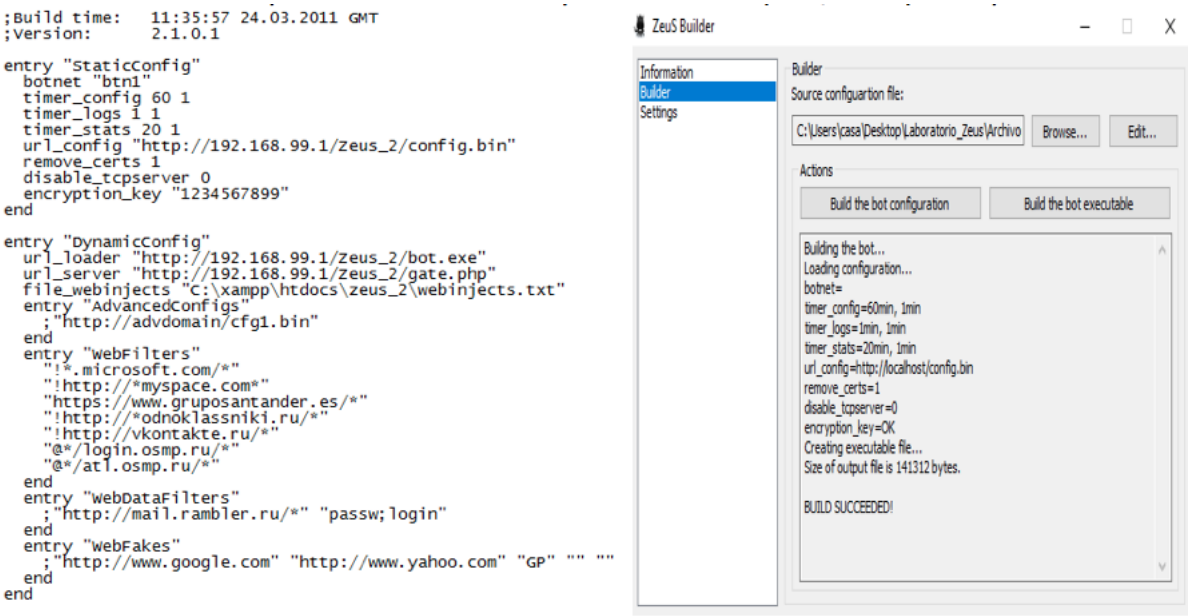


Figure 4.7: Archivo config.txt y Builder.exe.

- Estos bots solo actúan para máquinas basadas en windows a 32 bits, como windows 7.

Después de comentar estas dos particularidades hay que hacer llegar a la víctima el ejecutable bot. Esto se puede realizar bien por correo electrónico y cambiando la extensión del archivo (para que parezca otro tipo de archivo), a través de una página web infectada (drive-by download) que descargue el archivo y lo ejecute en la máquina víctima, a través de otro programa malware que lo descargue por módulos, a través de una aplicación gratuita e incrustado en ella.

Hay que tener en cuenta que el "payload" (carga útil) del bot viene "ofuscado" a través de capas de cifrado que le protegen de herramientas de análisis estático como IDA Pro y otros desensambladores.

Una vez se instala el bot, la carga útil se ejecuta y se pone en funcionamiento la función de resiliencia y ocultación en la máquina víctima.

Normalmente este troyano (o bot) crea ciertos ejecutables y carpetas y se aloja (según su versión o variante) en los siguientes paths:

Variante 1

C:\WINDOWS\system32\ntos.exe – > Troyano binario o Bot.

C:\WINDOWS\system32\wsnpoem\audio.dll – > Contiene los datos robados.

C:\WINDOWS\system32\wsnpoem\video.dll – > Contiene la configuración cifrada.

Variante 2

C:\WINDOWS\system32\oembios.exe – > Troyano binario o Bot.

C:\WINDOWS\system32\sysproc64\sysproc86.sys – > Contiene los datos robados.

C:\WINDOWS\system32\sysproc64\sysproc32.sys – > Contiene la configuración cifrada.

Variante 3

C:\WINDOWS\system32\twext.exe – > Troyano binario o Bot.

C:\WINDOWS\system32\twain_32\local.ds – > Contiene los datos robados.

C:\WINDOWS\system32\twain_32\user.ds – > Contiene la configuración cifrada.

Variante 4

C:\WINDOWS\system32\sdra64.exe – > Troyano binario o Bot.

C:\WINDOWS\system32\lowsec\local.ds – > Contiene los datos robados.

C:\WINDOWS\system32\lowsec\user.ds – > Contiene la configuración cifrada.

Como hay muchas variantes de Zeus, el archivo ejecutable puede variar. Existen otros como 73mendjd.exe (2017) u onlineservicesw.exe (2017).

Posteriormente, Zeus utilizará uno de los ejecutables mencionados anteriormente para inyectar el código en uno de los dos procesos siguientes (dependiendo de qué privilegios logró adquirir) winlogon.exe o explorer.exe. Esa inyección de código se realiza con el objetivo de ocultar Zeus entre otros procesos.

Los archivos de configuración y almacenamiento no están solo ocultos, sino también cifrados con el algoritmo RC4. La clave de 256 bits de RC4 es específica para cada versión de los bots compilada, por lo que es imposible encontrar un descifrador universal que nos muestre los contenidos de los archivos del bot. Esta clave se almacena en el archivo ejecutable encriptado.

Zeus realiza unos cambios en los registros de windows con el objetivo de que el ejecutable de Zeus sea arrancado al inicio del dispositivo.

El firewall de Windows queda deshabilitado por el programa malicioso, pero esto hace que aparezca un icono de advertencia del centro de seguridad de Windows en la bandeja del sistema que será la única indicación visible de que el dispositivo ha sido infectado.

El robot emite un comando "M-SEARCH" para encontrar dispositivos de red UPnP (Universal Plug and Play). Esto es un intento de acceder e infectar otros procesos en el mismo sistema a través de la inyección de hilos remotos.

Los procesos infectados son habitualmente: winlogon.exe, svchost.exe y explorer.exe. Todo el proceso de distribución, compilación e instalación se resume en la imagen 4.8.

4.5.4 Comunicación de los bots con el servidor C&C

Una vez que el bot tiene el control de la máquina zombi se pone en contacto con el servidor C&C a través de un comando http/GET (gate.php) para obtener el último archivo actualizado de la configuración dinámica (config.bin) que recibe del servidor C&C.

A continuación, el bot comenzará a capturar y registrar información de la máquina infectada según lo marcado en el archivo DynamicConfig. El bot envía dos comandos http/POST para enviar el registro recopilado y la información de estadísticas al servidor C&C.

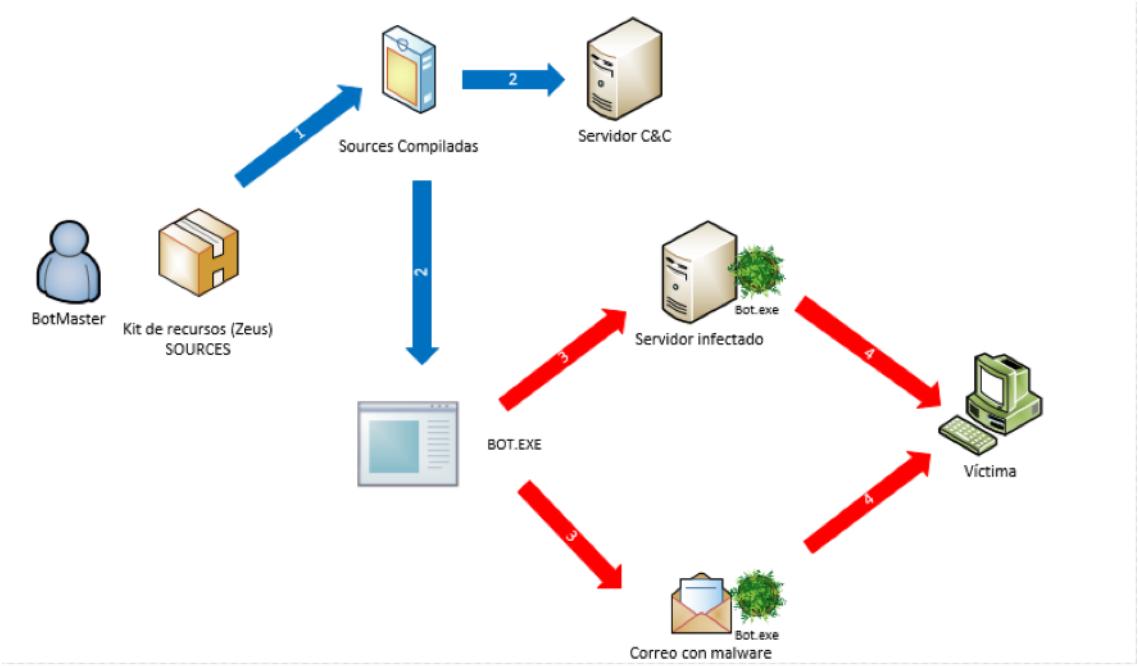


Figure 4.8: Proceso de creación y contaminación de Zeus.

Existen tres temporizadores que se han establecido en los valores en StaticConfig (config.txt), para que:

- Obtenga un nuevo archivo de configuración (DynamicConfig) del servidor C&C (predeterminado 60 minutos).
- Envíe los datos recogidos al servidor C&C (predeterminado 1 minuto).
- Envíe las estadísticas al servidor C&C (por defecto 20 minutos).

Si la respuesta a un http/POST contiene un comando script (cifrado), el bot lo ejecuta y devuelve una indicación de éxito o error junto con cualquier dato obtenido. A parte de estas comunicaciones prefijadas, el bot establece comunicaciones aleatorias de comprobación (técnicas de evasión de traza). Los puertos de comunicación son aleatorios para cada bot.

4.5.5 Selección de las versiones de las máquinas virtuales

Como ya se ha comentado anteriormente las máquinas utilizadas deben ser Windows de 32 bits, con lo que las seleccionadas han sido las de Windows 7 (32 bits), aunque nos hemos encontrado con los siguientes problemas:

- Estas máquinas tienen una licencia trial de un máximo de 90 días con lo que es necesario realizar algún snapshot de las máquinas de vez en cuando con lo que reduce los recursos de disco duro de la máquina anfitrión.

- El problema más importante fue el de las distribuciones (distros) descargadas de la página web de Microsoft. Todas las que tengan la misma versión son copias "exactas" y esto nos plantea un problema con los bots.

Asumiendo que en un entorno real sería casi imposible encontrar 2 computadoras exactamente iguales, en nuestro laboratorio SI tenemos todas ellas iguales y el problema se da cuando son infectadas con el bot y se ponen en comunicación con el servidor C&C.

El servidor C&C registra cada bot con un nombre que depende de la máquina zombi con parametros como su versión, licencia, versión de Explorer, etc. Como todas las imágenes son iguales los bots se registran con el mismo nombre en la BBDD y la información de unas con otras "se sobrescribe" es decir, solo tenemos una máquina zombi (bot) conectada y recibiendo información y por si fuera poco solo podemos distinguirla por su IP (que sería la misma si se comunicasen a través de un NAT). Es decir, en el peor de los casos no sé qué máquina me está enviando la información y solo podría gobernar una de ellas.

Así pues, se optó por descargar tres versiones de Windows 7 (todas las que ofrece la web) que se diferencian por su versión de IExplorer y de esta manera los tres bots tienen diferentes nombres y pueden ser gobernados sin ningún problema. Hay que poner de manifiesto que otra solución podría haber sido el cambiar el nombre de la computadora (máquina virtual) cambiando el nombre del bot (Este se hizo posteriormente).

Las seis máquinas virtuales tienen roles diferentes por lo que disponen de diferentes aplicaciones y/o servicios. La disposición de las aplicaciones en cada una es:

- Servidor C&C.
  - Apache (Servidor XAMPP): Sirve de servidor web para la gestión, mando y control de la Botnet.
  - MySQL (Servidor XAMPP): BBDD a la que se conecta el servidor web para el almacenamiento de datos de todo lo relacionado con la botnet.
  - WireShark: Analizador de protocolos que sirve para ver el tráfico entre los bots y su servidor C&C.
  - Blue Botnet: Botnet malware que sirve para realizar un ataque DDoS compuesto por una consola de control y ejecutables a distribuir entre las máquinas. Este ejecutable sera puesto a disposición a los bots para que se lo descarguen y realizar un ataque DDoS con él.
- Maquinas zombis.
  - Internet Explorer: Ya viene preinstalado en el ordenador y se empleará para simular accesos a servidores web como si de un usuario se tratara. Se implementará una inyección web a un usuario como práctica.

  - Diversos ejecutables:
    * Bot_Blue_botnet
    * HitmanPro
    * Belkasoft -dump Memory RAM-
    * Dependency Walker
    * PEStudio.

- Maquina víctima.
  - Apache (Servidor XAMPP): Sirve de servidor web para simular una web lícita a la que accede un usuario-víctima. Se ha reutilizado una web creada para una PEC anterior de la UOC. Se practicará un ataque DDoS a dicha web.
  - MySQL (Servidor XAMPP): BBDD a la que se conecta el servidor web y que dispone de verificación del usuario y contraseña, así como de algo de información a mostrar.

Con todo lo anterior se ha establecido la siguiente topología de red (ver imagen 4.9) dentro de la misma red local (192.168.99.0/24) en la que todos los equipos pueden acceder entre sí sin NAT o puertas de enlace entre ellas. Se muestra la imagen 4.10 de VMWare Pro 12 con las máquinas en funcionamiento.

Nota: Es de reseñar que mas adelante se ha añadido otra máquina zombi mas a nuestra practica con el objetivo de realizar ataques DDoS más efectivos.

A continuación, se expondrán las pruebas realizadas en nuestro laboratorio/escenario. Las dividiremos en cuatro prácticas para cubrir casi todos los aspectos en los que emplear esta botnet.

1. Práctica de funcionamiento básico y empleo de comandos.
2. Práctica de web injection.
3. Práctica de ataque DDoS (Blue Botnet).
4. Práctica del análisis forense de un equipo infectado (bot).

Solo se han propuesto las pruebas anteriores por que de otra manera la extensión de este trabajo seria inabordable, pero se quiere señalar que existen otro tipo de ataques muy interesantes [42] como son los siguientes:

- Se pueden redirigir las consultas de web legitimas a falsas para poder hacer un ataque de tipo phishing. Esto se puede configurar en el archivo “config.txt” en el parámetro “WebFakes”.

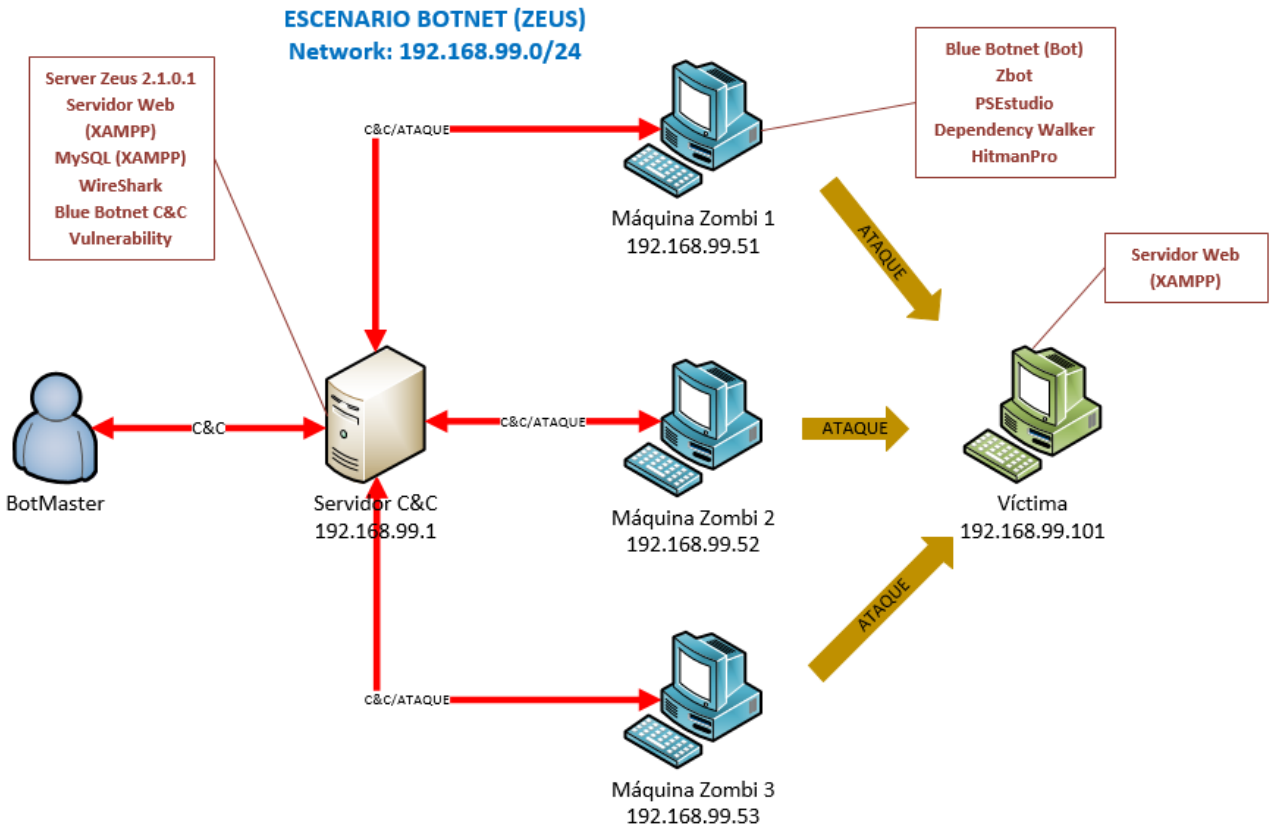


Figure 4.9: Escenario final práctico.

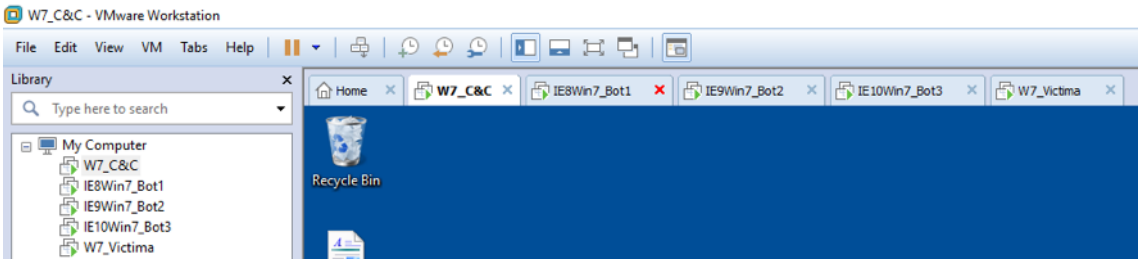


Figure 4.10: Máquinas virtuales del laboratorio en funcionamiento

- Especificar URL's para que sean monitorizadas e incluso envíen automáticamente una imagen de pantalla (screenshot) cuando se detecta la pulsación del botón izquierdo del raton. Esto se puede configurar en el archivo "config.txt" en el parámetro "Webfilters".

- Añadir entradas al DNS de la máquina local (HOSTS) para realizar redirecciones a sitios maliciosos (DNS poisoning). Esto se puede configurar en el archivo "config.txt" en el parámetro "DNSMap".

4.6 Práctica de funcionamiento básico y empleo de comandos

Inicialmente vamos a comprobar el funcionamiento de nuestra botnet con la configuración de los bots por defecto sin modificar el archivo "config.text" para otros tipos de ataques que realizaremos más adelante.

Es necesario recalcar que en la botnet de Zeus las máquinas zombis no son solo máquinas desde las que realizar ataques a otra víctima, sino que son víctimas en sí mismas. De hecho, la mayor parte de sus funcionalidades son, en ese sentido, el extraer la información de las máquinas infectadas. No obstante, también se realizará una práctica en la que una víctima será atacada desde las máquinas zombis.

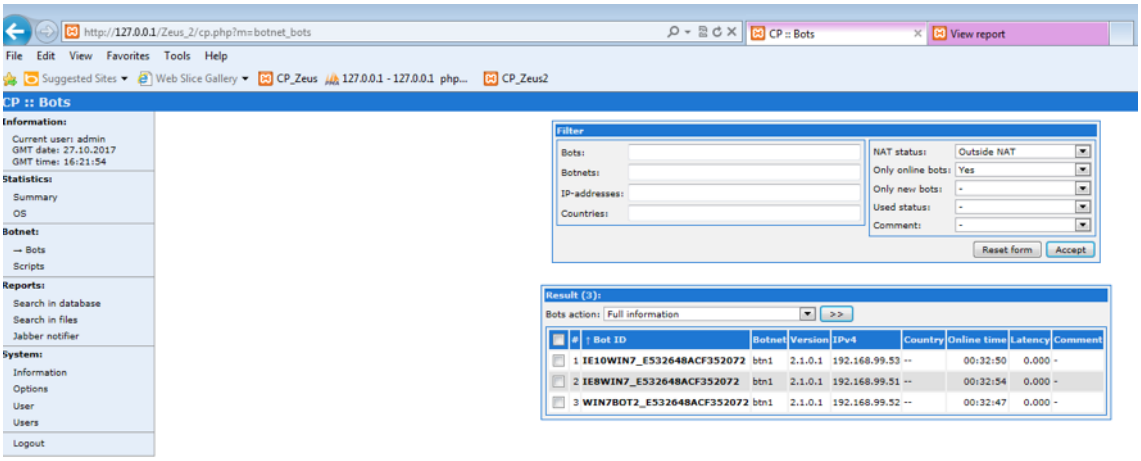


Figure 4.11: Imagen del CP con tres bots conectados al servidor C&C.

El menú del servidor C&C es amplio y no se quiere inundar este trabajo con imágenes que solo se centren en ver los menús, por lo tanto, se presentarán solo las partes más relevantes de él.

En la imagen 4.11 vemos que tenemos conectados a nuestros tres bots (máquinas zombis) y que podremos obtener de ellos información que será registrada en la BBDD para consultarla cuando queramos. En ella se aprecian las denominaciones de los bots (dependientes del nombre de la máquina y de la versión del S.O). Todas se conectan con el servidor C&C a través de un puerto aleatorio designado entre el bot y el servidor C&C. Podemos observar la denominación de la botnet (btn1) que hemos configurado en nuestro archivo "config.txt" antes de generar los bots.

Ahora solicitamos un informe y un screenshot de las tres máquinas para que nos muestren lo que tienen en pantalla. Podemos ver en 4.12 el sistema operativo de la máquina anfitrión-zombi, su IP, el nombre del bot, el S.O del equipo, etc.

La siguiente prueba se centrará en obtener las credenciales usadas desde un equipo infectado a un servicio web (podria ser el de un banco). Para hacer el rol de servidor se ha configurado los servicios de BBDD y Apache en XAMPP en la máquina virtual "víctima" como si de un servidor web se tratara. Las páginas y la BBDD que se han cargado son las relativas a una practica que

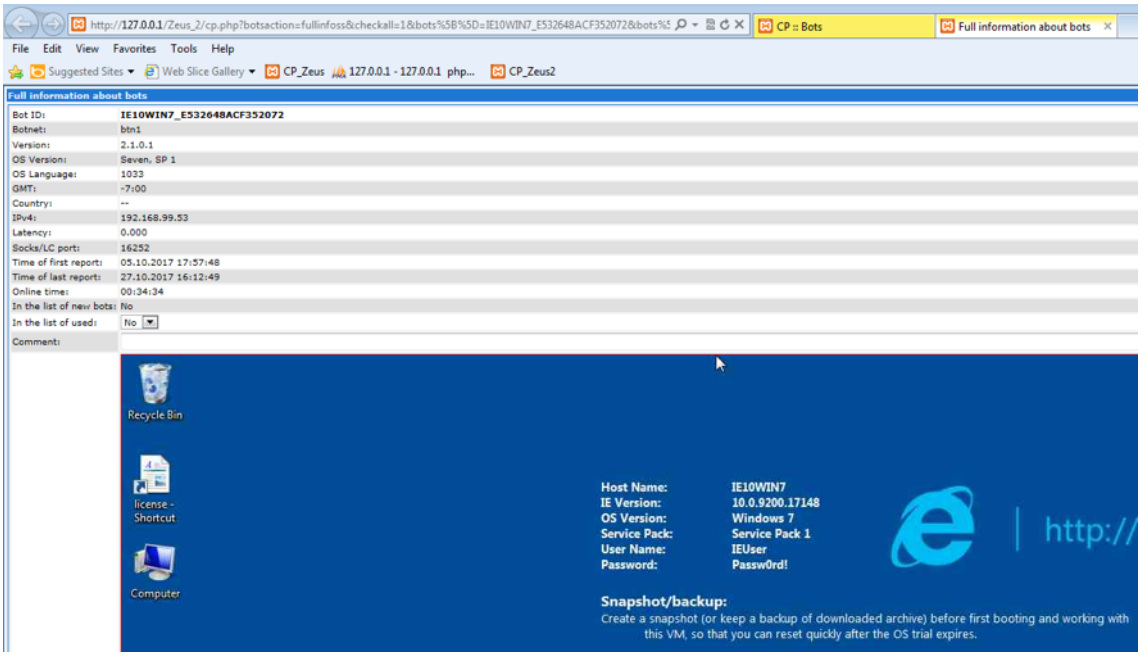


Figure 4.12: Informe + ScreenShot solicitado a los bots.

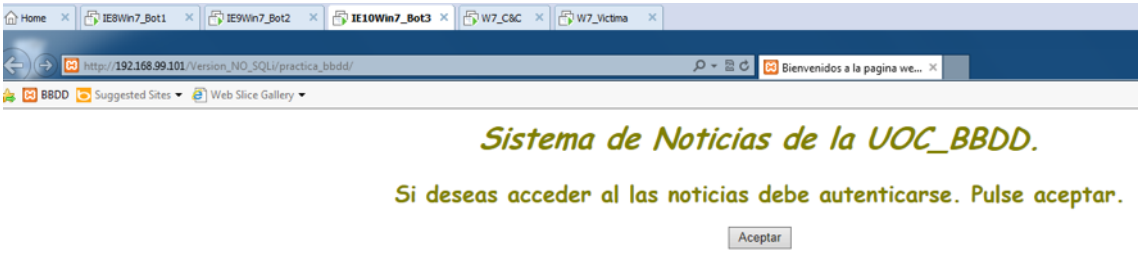


Figure 4.13: Página principal del servidor víctima.

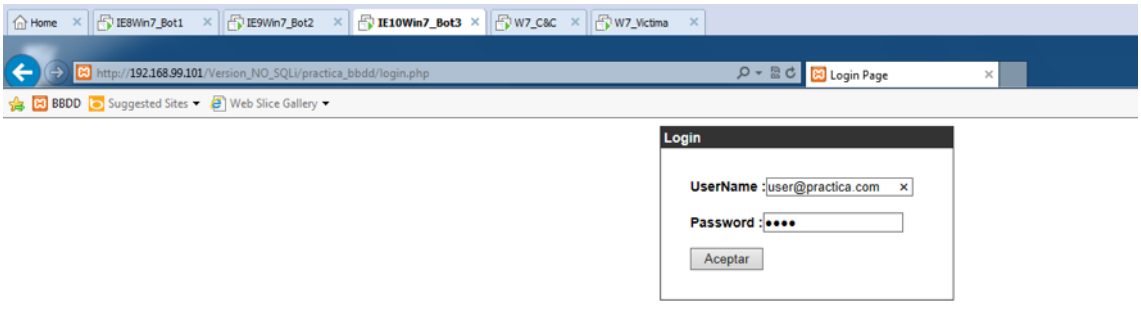


Figure 4.14: Página de login del servidor víctima.

se realizó en la asignatura “Seguridad en BBDD” de la UOC. La imagen 4.13 nos muestra la página de bienvenida que recibe el usuario-víctima que accede desde el equipo infectado bot3

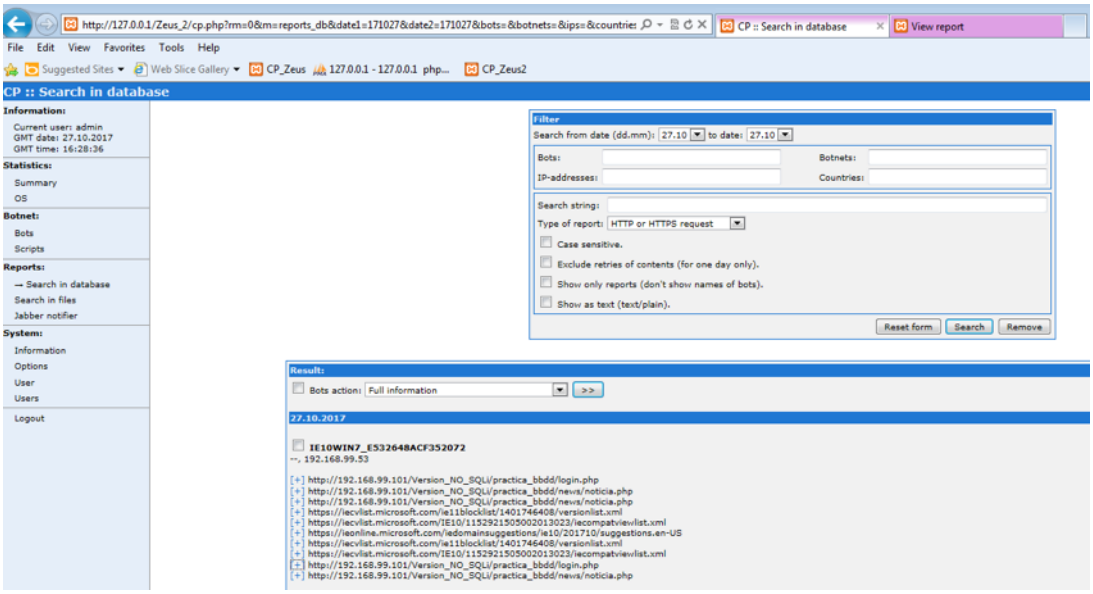


Figure 4.15: Solicitud de comunicaciones http a un bot.

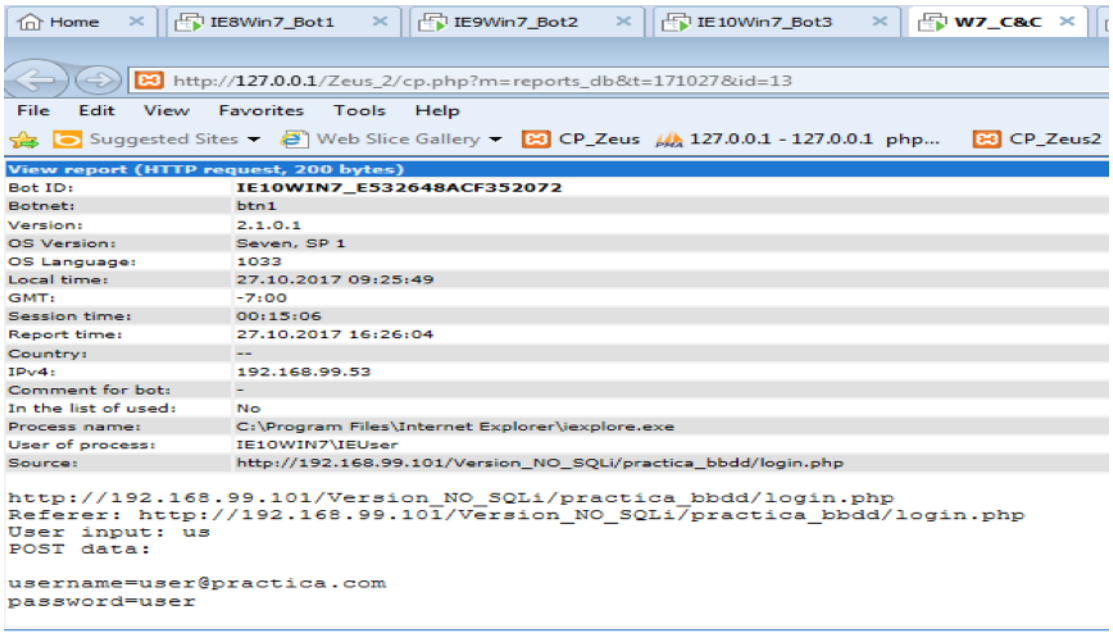


Figure 4.16: Informe de un acceso web de un bot (máquina zombi).

(llamado así para discriminarlos convenientemente).

Cuando se accede al “login” a través del botón de esa página nos solicita un usuario y contraseña que en este caso son “user@practica.com” y como password “user” como se ve en la imagen 4.14 siguiente.

En la imagen 4.15 solicitamos información de las comunicaciones http que ha hecho la víctima (bot3) y vemos que nos muestra todos los accesos hechos desde esta máquina y podemos acceder a la información de cada una de ellas.

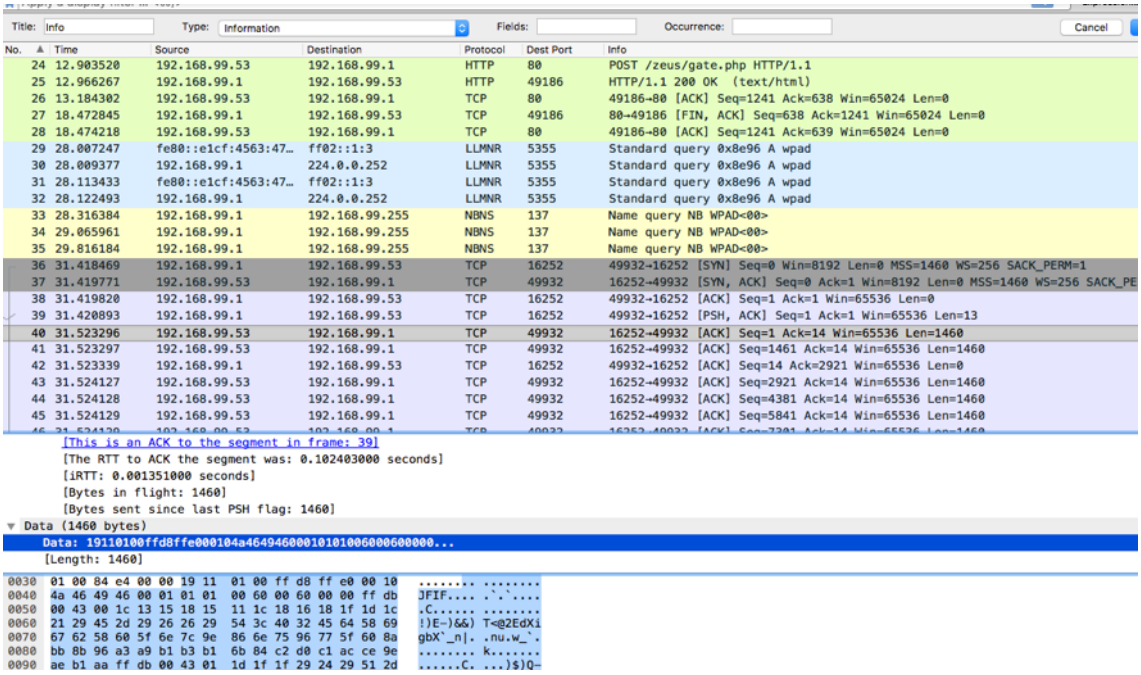


Figure 4.17: Tráfico generado entre servidor C&C y los bots.

Al acceder a la información de la navegación a la página web “login.php” vemos que nos muestra no solo información de los accesos sino, además, el usuario y contraseña en texto plano (ver imagen 4.16).

Se ha recogido el tráfico de estos intercambios de información entre el server C&C y el bot3 y vemos en la imagen 4.17 que existe una comunicación por el protocolo http desde el cliente (bot3) al servidor C&C llamando a “/Zeus/gate.php” (línea 24) a traves del protocolo POST para recibir comandos u ordenes del BotMaster (puerto 80 del servidor C&C).

Tras recoger los comandos del servidor C&C se observa el envio de la información requerida a través de un ptocolo TCP por un puerto aleatorio (acordado entre el servidor y el cliente) como se ve a partir de la linea 40. Toda la información (como ya se comento) va cifrada con RC4. Es importante aclarar que al establecer un puerto de comunicación la víctima puede comunicarse con ella a través de ese puerto para que el servidor C&C le envie los “ACKS” cuando recibe información.

Se quiere mostrar la funcionalidad de script (comandos de Zeus) en ella se pueden enviar órdenes a los bots para que las ejecuten y nos informen de ello cuando lo realicen (esta realización es asíncrona).

Existen multitud de comandos implementados en el servidor C&C. Algunos de ellos son: *os_shutdown*, *user_execute*, *os_reboot*, *bot_uninstall*, *bot_update [url]*, etc. La demostracion se hará con un script que ejecute un “.exe” de la máquina anfitriona de ese bot (bot3). Se le ordena a dicho bot que abra el “wordpad” con el comando: *user_execute “%ProgramFiles%\Windows*

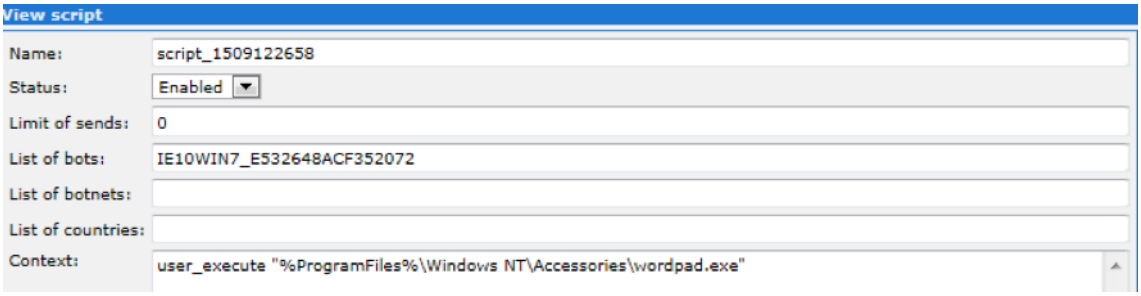
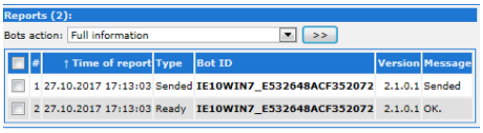


Figure 4.18: Creación de un script (comando).

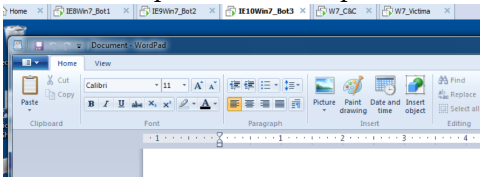
NT\Accessories\wordpad.exe"

Se puede ver en la imagen 4.18 como hemos contruido este script y se lo hemos asigando solo a un bot y lo habilitamos (enable).

Cuando se ejecuta el script (asincronamente) nos muestra un mensaje de enviado y de recibido por parte del bot (OK), como vemos en la imagen 4.19. Ello nos indica que el script ha sido recibido y ejecutado por la máquina infectada por el bot (máquina zombi). Por ultimo, vemos con en dicho equipo se ha abierto el wordpad (ver imagen 4.19.).



(Respuesta a un Script.)



(Accion ejecutada.)

Figure 4.19: Ejecución Script.

Se ha observado que los comandos enviados a los bots para realizar una acción son respondidos por estos en un tiempo variable y dependiente del archivo "config.bin". Estos tiempo, a veces son largos, debido a que la configuración por defecto del archivo "config.txt" (que se utiliza para crear el archivo "config.bin") tiene unos tiempos adaptados a un entorno real, pero no a un entorno de laboratorio de pruebas.

Antes de continuar con la realización de las siguientes pruebas vamos a ajustar estos tiempos explicándolos a continuación.

Hay 3 parametros dentro de la sección "StaticConfig" del archivo "config.txt" que vamos a variar:

- timer_config: Esta opción le dice a la víctima (al bot) con qué frecuencia buscar un nuevo archivo de configuración. Por defecto los valores están puestos en un tiempo máximo de 60 minutos y un minimo de 1. Vamos a poner un máximo de 2 minutos y un minimo de 1.

- timer_logs: Esta opción le dice al bot la frecuencia con la que debe enviar los datos almacenados del equipo víctima. Por defecto los valores están puestos en un tiempo máximo de 1 minutos y un minimo de 1. Lo dejaremos como está.

- timer_stats: Esta opción le dice al robot con qué frecuencia enviar las estadísticas de las víctimas. El tipo de estadísticas que se envían son el estado en línea, puertos abiertos, servicios, estado SOCKS, estado NAT y capturas de pantalla. Por defecto los valores están puestos en un tiempo máximo de 20 minutos y un minimo de 1. Vamos a poner un máximo de 2 minutos y un minimo de 1.

4.7 Práctica de web injection

Para esta practica tenemos que modificar el archivo "webinjects.txt" [39]. En dicho archivo se alojan las webs que quieren modificarse en las víctimas a través de una inyección de datos. El principio de funcionamiento de un ataque web injection es el siguiente:

1. La víctima visita una página web donde existe unos campos de introdución de datos para autenticarse (por ejemplo, usuario y contraseña).

2. El bot comprueba si la página visitada esta entre las que el tiene que "vigilar". Si es asi busca los datos marcados dentro de esa web a inyectar.

3. El bot se interpone entre las comunicaciones del servidor web legitimo y el "front-end" (imagen web que se le presenta a la víctima) e inyecta los campos marcados en el archivo "webinject.txt" (se usó en la compilación del archivo "config.bin" que posee el bot) para que se le presenten a la víctima los campos reales y el campo inyectado malicioso.

4. La víctima introduce todos los datos que le solicita la web inyectada.

5. Todos los datos son enviados en texto plano al servidor C&C y tambien al servidor web legitimo con lo que el usuario accedería bien al recurso solicitado ignorando que ha introducido datos adicionales.

En nuestro ejemplo hemos configurado un servidor web (XAMPP) en una máquina llamada "víctima" que, aunque es un nombre un poco confuso, se ha llamado asi porque aparte de un servidor web servirá para intentar realizar sobre ella mas adelante un ataque DDoS.

Los pasos para configurar nuestra práctica de webinjection son los siguientes:

1. Modificar el archivo "webinjections.txt" para introducirle la web a inyectar con los siguientes parámetros (ver imagen):

   a) set_url – > le asignamos la URL que hay que "controlar".
   En nuestro caso *http://192.168.99.101/version_NO_SQLi/practica_bbdd/login.php*.

   b) data_before – > datos buscados en la web "original" y donde después inyectaremos los nuestros. En nuestro caso buscamos una etiqueta con nombre "username" (name="username").

c) data_inject – > datos a inyectar. En nuestro caso inyectamos un campo similar a los legitimos pero en el que pedimos el código PIN de una supuesta tarjeta de crédito. *<tr><td>PIN_Code:</td><td><input type="text" name="PINCode" id="PINCode" /></tr>*.

2. Volver a compilar los archivos "config.bin" y "bot.exe" y guardarlos donde son accesibles a los bots (configurado en el archivo "config.txt" y ya explicado anteriormente).

3. Crear un script para ser lanzado a todos los bots en el que se les solicite que actualicen sus archivos de configuración. La sintaxis es *bot_update http://192.168.99.1/Zeus_2/config.bin -f*. Donde:

   a) bot_update – > es el comando que deben de ejecutar los bots.

   b) URL – > Es la dirección donde se encuetra el archivo .bin de configuración.

   c) -f – > Parámetro que fuerza a los bots a descargarse y ejecutarlo, es decir que se actualicen inmediatamente.

4. Despues de enviar ese script y ser ejecutado por los bots la inyección web nueva ya esta configurada.

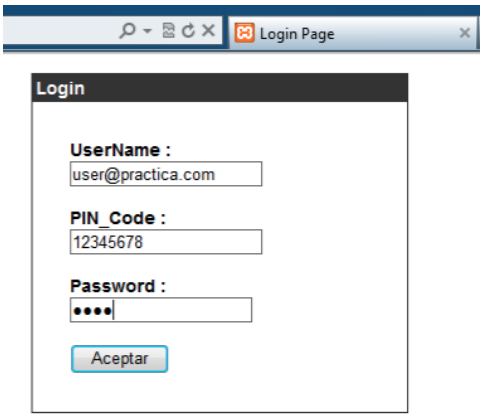


Figure 4.20: Acceso web modificado.

Podemos ver en la imagen 4.21 el script antes de ser ejecutado y el acceso a la web antes de este envio.

A continuación, vemos la imagen 4.20 con el campo nuevo inyectado.

Para obtener los datos debemos buscar (esperando un poco a que nos los mande el bot) en el menú de "search database" y en la petición http marcada abrirla donde veremos todos los datos introducidos por el usuario. Si realizamos un análisis con WireShark de esta tráfico entre la máquina víctima el servidor legitimo y el servidor C&C observamos que la máquina zombi envía TODOS los campos al servidor legitimo. Es decir, le envia los datos solicitados y los que se han inyectado.

El funcionamiento habitual de un servidor/servicio web es recoger los datos que se han solicitado y no recoger datos adicionales que se puedan haber introducido (puesto que desconoce que existan más). En la imagen 4.22 podemos ver el envio de información entre el dispositivo víctima/zombi y el servidor legítimo. Observamos que se le ha envido "username", "PINCode" y "password" asi como sus valores correspondientes (método POST).

Sin embargo, en la imagen 4.23 comprobamos que no podemos ver lo que envía el bot de el ordenador víctima al servidor C&C ya que esta el texto cifrado. Asumimos que es dicha información ya que utiliza un método POST a "Zeus_2/gate.php" y en el encabezado hace referencia al navegador mozilla. Ademas se produce al minuto o minuto y medio de la acción

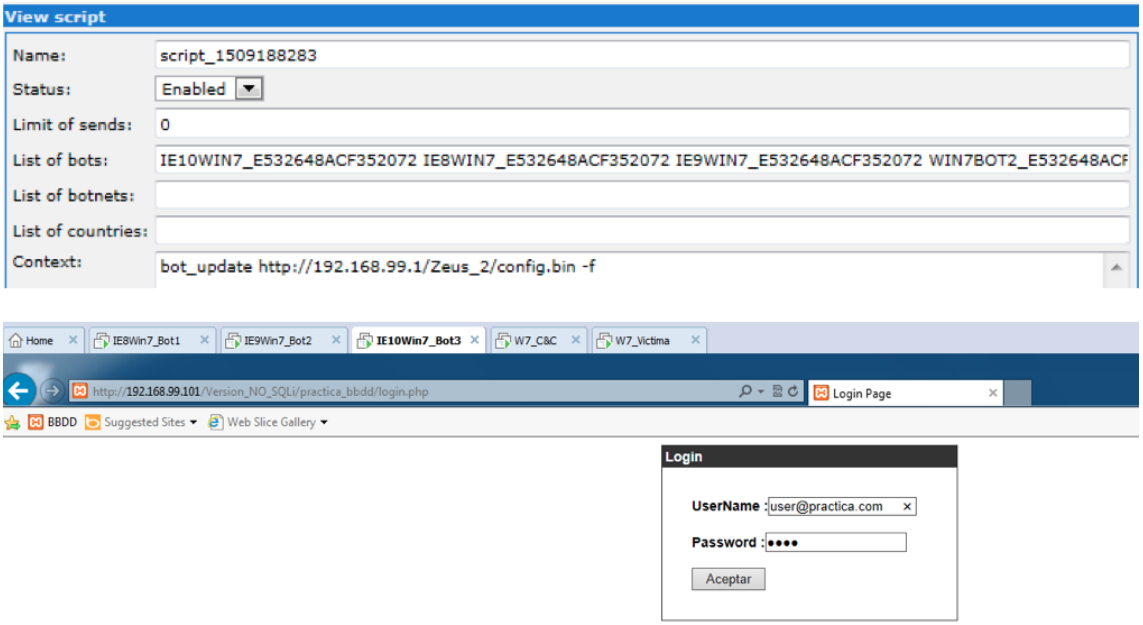
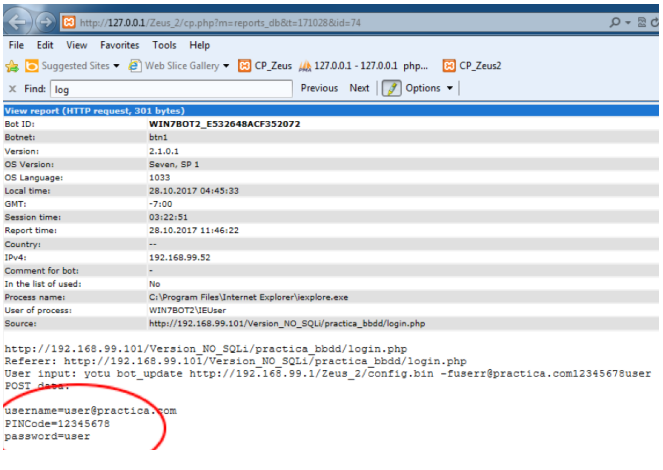


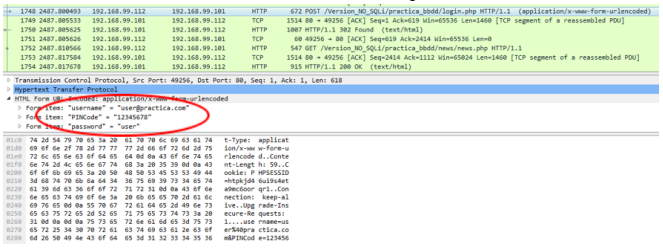
Figure 4.21: Script y acceso web antes de ejecutarse.

anterior y como se han modificado los tiempos (entre 1 y 2 minutos) se tiene dicha certeza cuando nos dirigimos al servidor C&C y comprobamos que ya tenemos esa información. Hay que tener en cuenta que la comunicación para el envio de información siempre es a través de "Zeus_2/gate.php" portocolo http y método POST.

Una cuestión interesante es que sucede cuando las comunicaciones a interceptar se producen a través del protocolo https (SSL). Zeus hace una interceptación de los datos antes del envio y de su cifrado. Asi pues es ilusoria la sensación de seguridad a través del protocolo SSL ya que el bot de Zeus realiza el proceso de keylogger en la propia máquina zombi.



(Servidor C&C web injection.)



(Tráfico web injection al servidor legítimo.)

Figure 4.22: Resultados Web injection.

4.8 Práctica de ataque DDoS (Blue Botnet)

Con esta práctica se quiere realizar un ataque DDoS a una víctima en la que se encuentre en funcionamiento un servicio web.

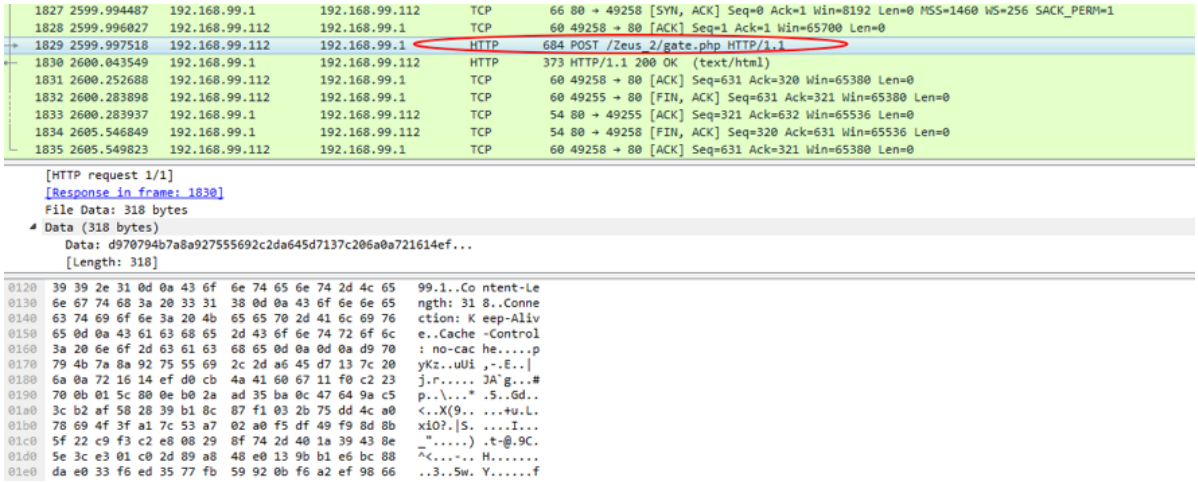


Figure 4.23: Tráfico al servidor C&C.

La versión de Zeus con la que estamos trabajando NO dispone de ninguna capacidad de ataque DDoS preconfigurado. Una versión que si dispone de ello es GameOver. Como ya se comentó, en el mercado negro es posible obtener dichas capacidades pagando por ellas.

En este escenario se quiere aprovechar otras capacidades que dispone Zeus para poder realizar ataques DDoS a través de otra herramienta. La idea es que la botnet Zeus desplegada realice la instalación de otro malware con el fin de poder realizar este tipo de ataques.

Esta idea ya se ha empleado con el caso del ataque que se produjo con CryptoLocker en 2013. La red de distribución del malware de cifrado fué Zeus.

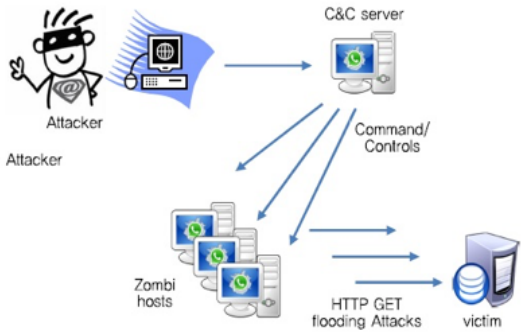


Figure 4.24: Ataque tipo HTTP [1].

La dificultad de realizar esta practica ha sido en la de obtener una herramienta de DDoS. Ello es debido a que existen ya herramientas (como UfoNet) preparadas para hacer ataques DDoS pero sin ningún control de los equipos que lanzan el ataque (bots). No obstante, se buscó una herramienta en la que se puediese tener un control de las máquinas infectadas (red botnet) y que pudiese ser distribuida y "activada" por nuestra botnet Zeus.

Inicialmente se barajó el uso de "Dirt Jumper v5" que se consiguió descargar con muchas dificultades. Sin embargo, su funcionamiento daba problemas en todos los navegadores empleados y no se pudo implementar su uso final. Al final se optó por el uso de "Blue Botnet" una botnet

dedicada exclusivamente a ataques DDos y con la posibilidad de distribuir sus bots via remota (a través de Zeus). La version descargada funcionaba bien, la interface es sencilla en su manejo y la creación de sus bots es simple y rápida.

4.8.1 Blue Botnet

Esta botnet se basa en un panel de control (cpanel) o servidor de control que debe de alojarse en un servidor web (XAMPP en nuestro caso) desde el que enviar comandos a sus bots. No necesita de una BBDD (como es el caso de casi todas las consultadas) sino que guarda la información en archivos de texto. Ello es posible ya que las botnets de ataques DDoS no requieren de el alojamiento de mucha información, simplemente tener registrados a sus bots y poderles enviar los paramentros del ataque. Es necesario reseñar que existe muy poca información (por no decir nula) de esta botnet.

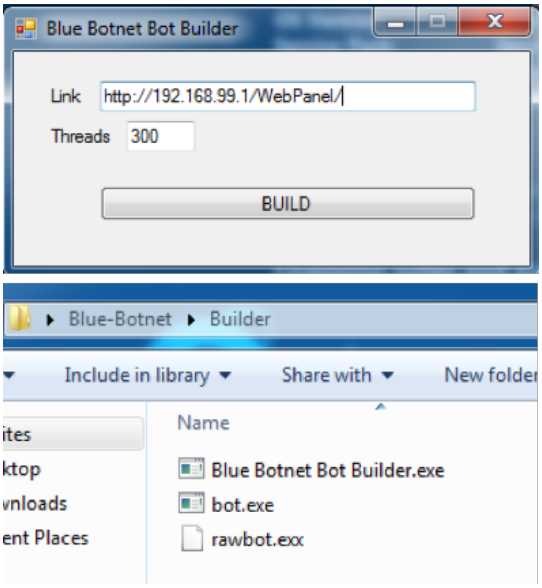


Figure 4.25: Creación bot Blue Botnet.

Blue Botnet puede realizar muchos tipos de ataques en diferentes capas del modelo TCP/IP, entr los que están [36]:

- UDP flood – > Ataque a través de paquetes UDP a puertos aleatorios.
- SYN (SYN Flood) – > Ataque que se produce aprovechando la debilidad del “three-way handshake”, dode el servidor espera el ACK del cliente y deja la conexión abierta un tiempo predeterminado.
- HTTP – > Ataque de capa 7 que se produce con peticiones masivas del parámetro GET y/o POST (ver 4.24).
- TCP – > Ataque que aprovecha las vulnerabilidades propias del protcolo TCP “three-way handshake” y los asociados a las características de este protocolo.
- HTTP Proxy – > El mismo que el anterior, pero a través de servidores proxy. Este tipo de ataque hace que podamos ocultarnos más a través de dispositivos intermedios lo que dificulta el detectar la procedencia del ataque.
- XMLRPC – > Este ataque se aprovecha de la posibilidad de ejecución remota de funciones XML en algunas webs, lo que se utiliza para enviar miles de estas solicitudes al objetivo.

Ademas esta botnet permite modificar (adaptar) los proxies remotos a utilizar y las XML-RPC_list pudiéndolo adaptar a nuestro escenario o necesidades.

Se estima que con solo 50 bots de esta herramienta es posible inhabilitar (tirar) un servidor web. Por otro lado, esta herramienta dispone de un "builder" para construir nuestros bots.

Para la realización de esta practica se ha añadido un equipo zombi más (bot4) para aumentar el tráfico en el intento de que nuestro ataque DDoS contra el servidor de la máquina víctima sea exitoso. Los pasos para configurar nuestra práctica de DDoS son:

1. Preparación de los bots de Blue Botnet:

    a) Debemos alojar la carpeta "WebPanel" en nuestro servidor web para que actue como servidor C&C de Blue Botnet. Hemos instalado nuestra Blue Botnet en el equipo C&C donde ya tenemos el servidor de C&C de Zeus, asi pues será el mismo servidor para ambas botnets.

    b) Creacion de los bots para lo que le indicaremos la URL de nuestro servidor C&C (http://192.168.99.1/WebPanel/) y el numero de Threads (hilos) que podrá lanzar cada bot (por defecto 300). Ver la imagen 4.25

    c) Alojamiento del bot.exe de Blue Botnet en el servidor http en la raíz del servidor http://192.168.99.1/bot.exe (eso corresponde a C:\xampp\htdocs\bot.exe) para que pueda ser accedido via http. Aquí debemos de hacer la siguiente anotación: "*Si cambiamos el nombre del ejecutable NO funcionará, debido a que debe de ejecutarse como bot.exe*".

2. Preparacion en la distribución de la Blue Botnet por Zeus:

    a) Debemos crear un script donde se pida descargar y ejecutar el archivo bot.exe (de Blue Botnet) en cada máquina zombi (anfitriona de cada bot). El script es el siguiente: *user_execute http://192.168.99.1/bot.exe*. Este comando ordena al bot descargar el archivo "bot.exe" en *%TEMP%\random_name\bot.exe* y lo ejecuta.

    b) Guardar el script y mandar su ejecución por todos los bots de Zeus.

3. Preparacion del servidor (cpanel) de Blue Botnet:

    a) Accedemos a la dirección de Blue Botnet donde nos pide un password para registrarlo y a partir de entonces podamos utilizarlo.

    b) Una vez logeados accedemos al panel principal en el que existen 4 pestañas (imagen 4.26). Aquí nos interesan la de "Attack Hub" (desde donde podemos realizar los ataques) y la de "Online Bots" (donde podemos ver que IP de bots tenemos activas). La pestaña de "Settings" tiene utilidad para ataques HTTP Proxy y XMLRPC que no vamos a emplear en esta práctica.

c) Esperamos a tener nuestro 4 bots “en línea” antes de actuar.

4. Realizacion del ataque DDoS (ver imagen 4.26) contra el equipo víctima (192.168.99.101). Realizaremos el ataque contra la página web empleada en la practica anterior.

   (*http://192.168.99.101/Version_NO_SQLi/practica_bbdd/index.php*).

Vamos a realizar 2 tipos de ataques DDoS (SYN y TCP ATTACK) y a la vez acceder al recurso a través del propio servidor C&C. Ademas, “intentaremos” capturar los datos a través de WireShark (se ha remarcado “intentaremos” porque la cantidad de datos que recoge WireShark es enorme y habitualmente se queda bloqueado).

Para generar estos ataques debemos ingresar la IP de nuestra vitima y el puerto al que atacar, que en este caso es el puerto 80 donde corre nuestra aplicacion web (192.168.99.101).

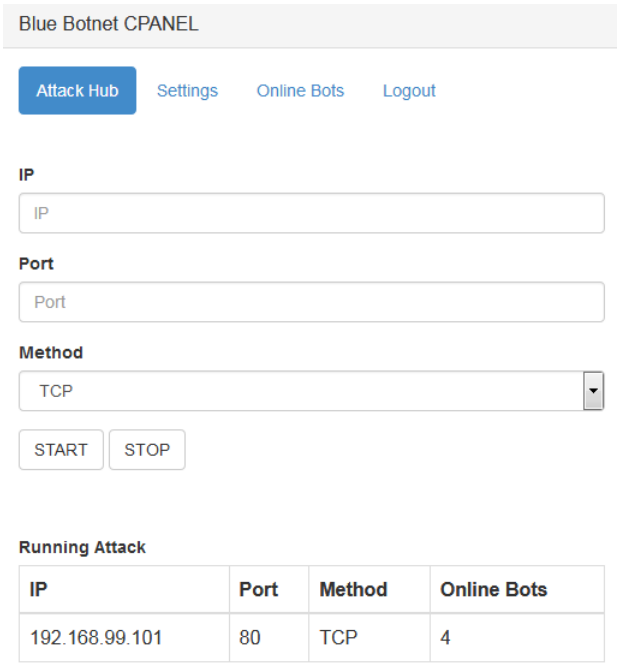


Figure 4.26: Creación bot Blue Botnet.

Ataque SYN [10] Cuando un cliente desea iniciar una conexión contra otro equipo, inicia la conversación con un “SYN”, en el lado del servidor se recibe el SYN y responde con un SYN+ACK, finalmente el extremo que empezó la conexión contesta con un ACK y ya pueden empezar a transmitir datos (ver imagen 4.27).

Un ataque de tipo Syn Flood lo que hace es empezar un numero especialmente alto de inicios de conexión que nunca son finalizados, dejando al servidor a la espera del ack final, y por tanto consumiendo recursos de forma desproporcionada. Por defecto la espera de respuesta suele ser de 180 segundos.

Para generar este ataque debemos ingresar la IP de nuestra vitima y el puerto al que atacar, que en este caso es el puerto 80 donde corre nuestra aplicacion web.

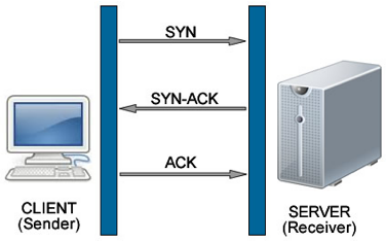


Figure 4.27: 3-way handshake [10].

En la imagen 4.28 de WireShark con el tráfico capturado, una vez que se ha lanzado el ataque, se puede observar que se lanzan paquetes contra el objetivo víctima (192.168.99.101) del tipo SYN desde diferentes puertos desde las máquinas zombis.

Este ataque no ha conseguido colapsar la máquina víctima y provocar el buscado DDoS. Se estima que serian necesarias mas máquinas para lograr esto con este tipo de ataque SYN Flood.

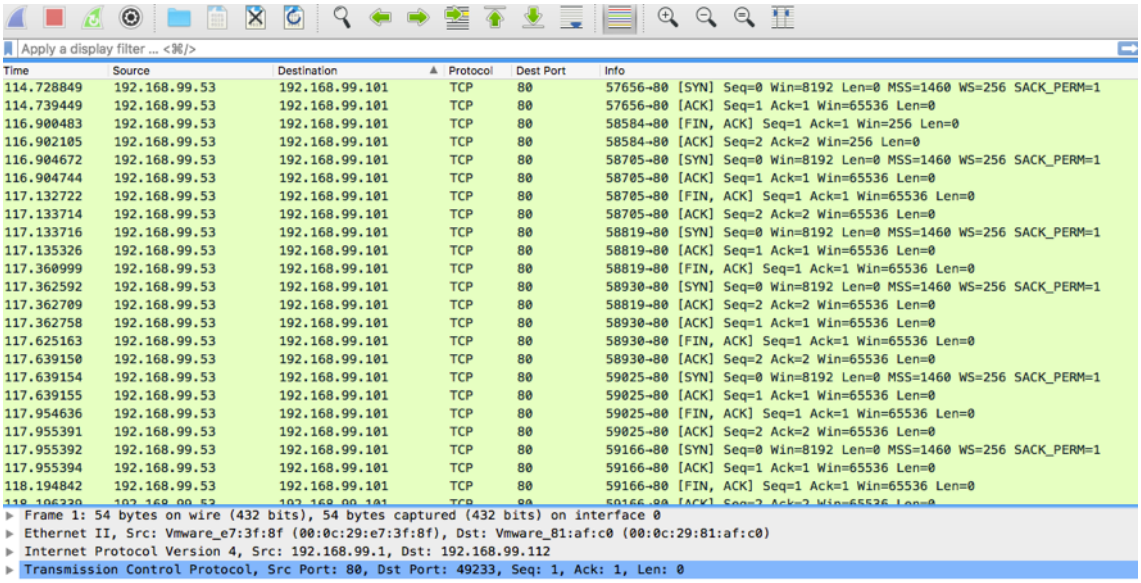


Figure 4.28: Tráfico generado por una ataque SYN.

Ataque TCP [11]

Estos tipos de ataques DDoS explotan algunas de las debilidades de diseño del protocolo TCP/IP. Por lo general, utilizan incorrectamente los seis bits de control (flags), o indicadores, del protocolo TCP/IP. Estos indicadores son SYN, ACK, RST, PSH, FIN y URG que sirven para interrumpir los mecanismos normales del tráfico TCP. Como este protocolo se basa un mecanismo de handshake de tres vías, donde cada solicitud crea una conexión semiabierta (SYN), una solicitud de respuesta (SYN-ACK) y luego un acuse de recibo de la respuesta (ACK). Los ataques que intentan abusar del protocolo TCP/IP a menudo envían paquetes TCP en el orden incorrecto, lo que provoca que el servidor de destino se quede sin recursos informáticos a medida que intenta comprender dicho tráfico anormal.

Otro tipo de ataque con esta técnica se llama “Duplicate ACK Spoofing” y funciona suplantando las solicitudes ACK que ya se han enviado. Ademas, se envía el tamaño máximo posible en los ACKS. Esto produce una saturación del buffer del servidor que, aunque disponga de un protocolo de “ventana deslizante” para la respuesta a los ACKS este se ve desbordado también por la recepción de los ACKS ya recibidos.

La forma en que funciona este ataque es permitiendo incrementar el SMSS (Sender Maximum Segment Size) cada vez que el receptor recibe ACK duplicados no pudiendo distinguir los ACK válidos de los falsificados.

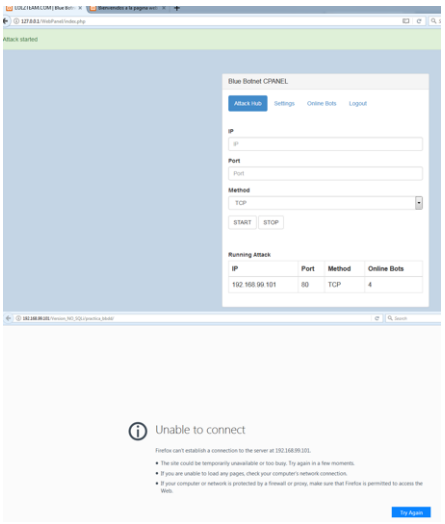


Figure 4.29: Ataque Blue botnet TCP.

Aunque Blue Botnet no especifica las técnicas que

utiliza se ha observado en un análisis de los paquetes con WireShark que utiliza ambas técnicas como puede verse en la imagend e la captura 4.30.

A continuación se realiza un ataque con técnica TCP de DDoS con cuatro máquinas zombi y su resultado (ver 4.29).

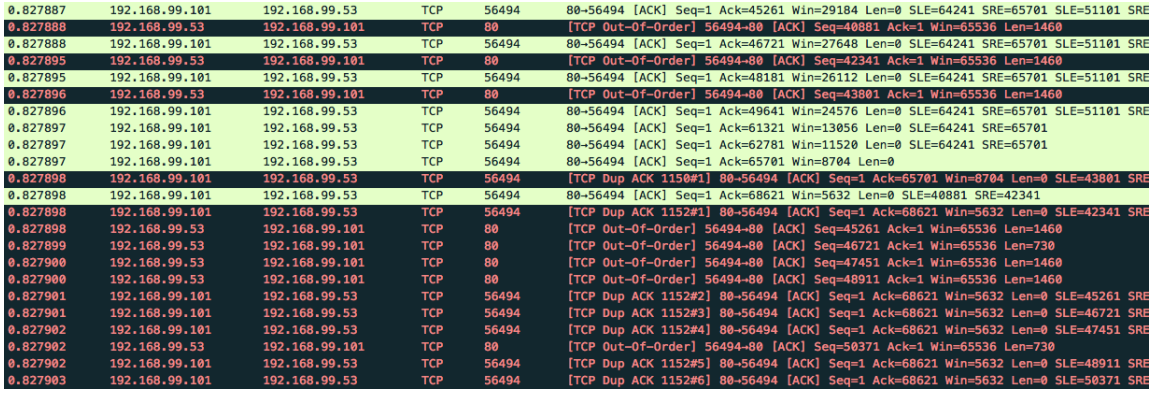


Figure 4.30: Trafico generado con ataque efectivo TCP.

Este ataque ha sido muy efectivo llegando a inutilizar el servicio de la web víctima en menos de 5 segundos después de lanzar el ataque (ver imagen).

En el tráfico capturado con WireShark mostrado en la figura 4.30, vemos la gran cantidad de ACKS que es enviado al servidor víctima (192.168.99.101) y la cantidad de paquetes de ACKS duplicados (TCP Dup ACK XXXX) y de ACKS desordenados (TCP Out-Of-Order).

Para generar este ataque debemos ingresar la IP de nuestra vitima y el puerto al que atacar, que en este caso es el puerto 80 donde corre nuestra aplicacion web.

4.9 Análisis forense de un equipo infectado (Bot)

Se pretende realizar un análisis forense de un equipo infectado [45] [43] [8] (podría tratarse de un honeypot puesto por nosotros). Para ello, se debe obtener una imagen de la memoria RAM de una de las máquinas infectadas por el bot. También se realizará el mismo proceso con otra máquina infectada para comparar los resultados.

Para la generación de la imagen de la memoria RAM hemos empleado la herramienta de Belkasoft (Version Trial) por su facilidad de uso, para luego, realizar un análisis de dicha imagen con Volatility (OpenSource).

Una vez obtenidas las imágenes de las dos máquinas (Bot1 y Bot3) empezaremos a realizar los análisis con la herramienta Volatility.

Lo primero que debemos hacer con volatility es obtener la versión del S.O de la imagen obtenida. Para ello empleamos el plug-in *imageinfo* (ver imagen 4.31), de esta manera podemos realizar el resto de consultas con un PROFILE establecido.

```
MacBook-Pro-de-Guillermo-Calvo-Ortega:volatility_2.6_mac64_standalone guillermosarajevo$ ./volatility_2.6_mac64_standalone imageinfo -f 20171
030_Bot1.mem
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/Users/guillermosarajevo/Desktop/volatility_2.6_mac64_standalone/20171030_Bot1.mem)
                      PAE type : PAE
                           DTB : 0x185000L
                          KDBG : 0x82b74c30L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0x82b75c00L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2017-10-31 01:57:54 UTC+0000
     Image local date and time : 2017-10-30 18:57:54 -0700
```

Figure 4.31: Información del sistema de la imagen RAM.

```
0x3dd13458          TCPv4    192.168.99.51:49184          192.168.99.1:80      CLOSED          -1
0x3dd16008          TCPv4    192.168.99.51:49190          192.168.99.1:80      CLOSE_WAIT      -1
0x3df8a008          TCPv4    192.168.99.51:49180          192.168.99.1:80      CLOSED          -1
0x3dfb09c0          TCPv4    192.168.99.51:21063          192.168.99.1:49260   CLOSED          -1
```

Figure 4.32: Procesos activos -Volatility-.

En segundo lugar realizamos un *netscan* para ver si podemos averigura que proceso (PID) se esta comunicando con una IP sospechosa. Esta IP sospechosa seria la del servidor C&C (192.168.99.1). El comando seria: *ĵvolatility_2.6_mac64_standalone –profile=Win7SP1x86_23418 netscan -f 20171030_Bot1.mem*.

Los procesos que han realizado dicha comunicación me aparecen con el PID -1 (ver 4.32) a que me indica que dicha comunicación esta cerrada (se cierra al detectar un volcado de memoria?).

Ahora realizo ahora un *psxview* (ver imagen 4.34). Este plugin me permite ver cuando los procesos aparecen en un tipo de consultas y no en otras. Psxview determina los procesos y las referencias cruzadas enumerando los procesos de PsActiveProcessHead (pslist), mostrando los procesos con el grupo de etiquetas como activos (psscan), y enseñándonos indirectamente el proceso mediante el escaneo ETHREAD (check_thrdproc), recorriendo PspCidTable, usando la tabla de control csrss.exe (que gestiona las creaciones y detención del resto de procesos) .

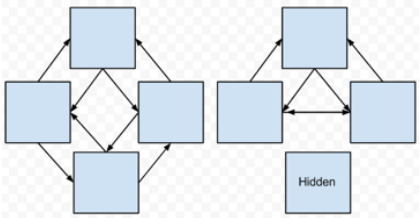
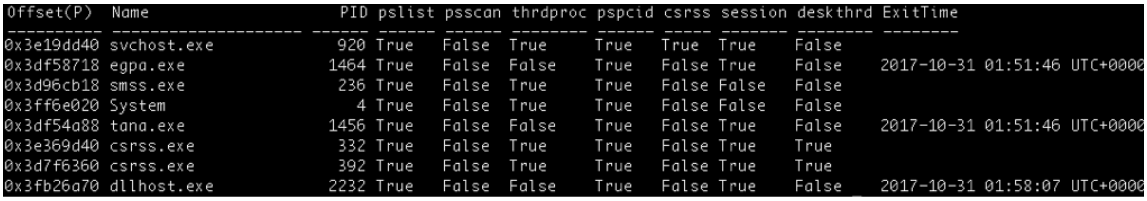


Figure 4.33: Vinculación entre proceso [45].

Por otro lado, PspCidTable contiene controladores para todos los procesos y subprocesos que están presentes en el sistema. El planificador de subprocesos lo usa para realizar un seguimiento de qué subprocesos se gestionarán.

Hay que tener en cuenta que *pslist* es un plugin de volatility que muestra información similar que si solicitara la lista de tareas. Como windows emplea listas doblemente vinculadas de estructuras EPROCESS, eso significa que cada elemento apunta al elemento anterior y posterior a él (ver imagen 4.33).

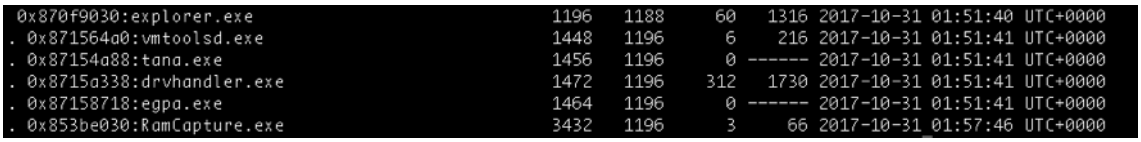
El malware puede desvincular estas listas y esconder un proceso malicioso del comando *pslist*.



Offset(P)	Name	PID	pslist	psscan	thrdproc	pspcid	csrss	session	deskthrd	ExitTime
0x3e19dd40	svchost.exe	920	True	False	True	True	True	True	False	
0x3df58718	egpa.exe	1464	True	False	False	True	False	True	False	2017-10-31 01:51:46 UTC+0000
0x3d96cb18	smss.exe	236	True	False	True	True	False	False	False	
0x3ff6e020	System	4	True	False	True	True	False	False	False	
0x3df54a88	tana.exe	1456	True	False	False	True	False	True	False	2017-10-31 01:51:46 UTC+0000
0x3e369d40	csrss.exe	332	True	False	True	True	False	True	True	
0x3d7f6360	csrss.exe	392	True	False	True	True	False	True	True	
0x3fb26a70	dllhost.exe	2232	True	False	False	True	False	True	False	2017-10-31 01:58:07 UTC+0000

Figure 4.34: Referencias cruzadas de procesos -Volatility-.

El malware realiza esto a través de la manipulación directa de objetos Kernel (DKOM). Para ello, posiblemente, el malware encuentra un proceso actual a través de PsGetCurrentProcess, luego encuentran el EPROCESS para ocultarse, y luego cambie FLINK y BLINK para que apunte a otra región de memoria válida.



0x870f9030:explorer.exe	1196	1188	60	1316	2017-10-31 01:51:40 UTC+0000	
. 0x871564a0:vmtoolsd.exe	1448	1196	6	216	2017-10-31 01:51:41 UTC+0000	
. 0x87154a88:tana.exe	1456	1196	0	------	2017-10-31 01:51:41 UTC+0000	
. 0x8715a338:drvhandler.exe	1472	1196	312	1730	2017-10-31 01:51:41 UTC+0000	
. 0x87158718:egpa.exe	1464	1196	0	------	2017-10-31 01:51:41 UTC+0000	
. 0x853be030:RamCapture.exe	3432	1196	3	66	2017-10-31 01:57:46 UTC+0000	

Figure 4.35: Árbol de procesos -Volatility-.

El comando introducido es: *./volatility_2.6_mac64_standalone –profile=Win7SP1x86_23418 psxview -f 20171030_Bot1.mem*.

Aquí se observa (ver imagen 4.34) que existen solo 3 procesos que se encuentre como “TRUE” en *pslist* y “FALSE” en thrdproc. La función ThreadProc (thrdproc) es una función definida por la aplicación que lo llama y que sirve como dirección inicial para un hilo. Este comportamiento es cuando menos “raro”.

Ademas, es extraño el nombre de dos de ellos “tana.exe” y “egpa.exe”. Estos procesos ni siquiera se ha encontrado en internet.

Ahora realizamos un *pstree* para ver la relación de que proceso depende de otro (ver imagen 4.35). Es decir en nuestro caso, vemos que el ejecutable tana.exe y egpa.exe son arrancados por explorer.exe (tiene lógica si quiero ocultar el arranque de un malware)

El comando introducido es: *./volatility_2.6_mac64_standalone –profile=Win7SP1x86_23418 pstree -f 20171030_Bot1.mem*

De hecho, si buscamos algún tipo de “traza” de nuestro malware en el arranque “winlogon” con el plugin *printkey*, encontramos que no se ha añadido ningún otro ejecutable (como si hacia versiones previas de Zeus que añadia sdr64.exe) en el arranque del dispositivo.

Se ha empleado también el plugin *mutantscan* que nos muestra los mutex de acceso al kernel. Un mutex no es mas que un sistema de sincronización entre procesos (semáforo). El que nos ha llamado la atención es “_SHuassist.mtx” que en búsquedas en la red nos los relacionan con varios troyanos como W32/VB.BKX y TROJ_SMALL.LJF.

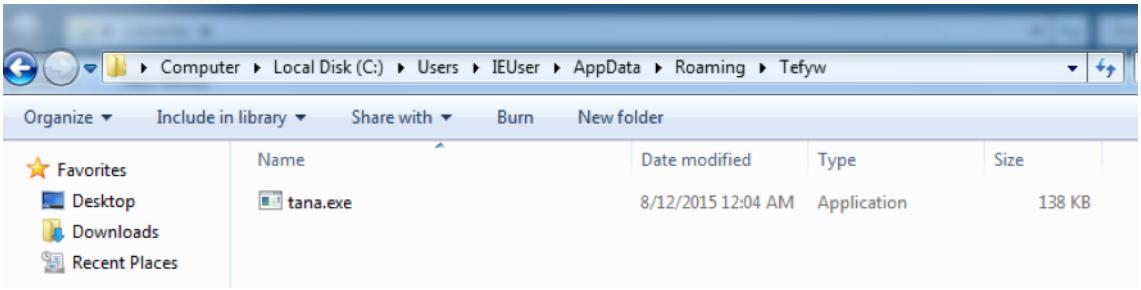


Figure 4.36: Archivos maliciosos.

Sin embargo, se sabe [8] que esta versión de Zeus utiliza GUID (Los GUID strings son únicos y aleatorios se podría manejar un mutex refiriendose a él) en lugar de mutex fijos con lo que resulta dificil o imposible detectarlos.

También se conoce por [8] que las versiones nuevas de Zeus (a partir de la 2.0) estan preparadas para poder "correr" varias instancias de Zeus de diferentes "Builders". Esto dificulta su limpieza (podría haber mas de uno en funcionamiento) y explica por que se usan nombres aleatorios en la generación de los archivos del bot y del uso de GUID en los mutex.

Los demás plugins no nos han arrojado mas información en nuestra investigacion.

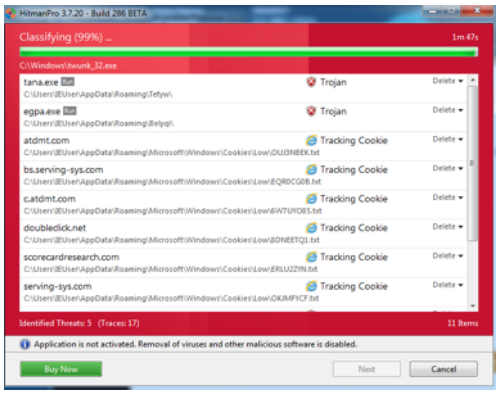


Figure 4.37: Detección malware.

Para corroborar los hechos, hemos realizado las mismas investigaciones con una imagen de la memoria de otro bot (Bot3) en el que hemos encontrado las mismas pistas con una salvedad, los archivos que en Bot1 eran "tana.exe" y "egpa.exe en el Bot3 se llaman "fuef.exe" y "oguc.exe". Ello nos llama la atención en el sentido de que se puede ver que esta versión de Zeus crea nombres aleatórios de los ejecutables aunque les aloja en el mismo directorio con la sintaxis *%User Profile%\Application Data\[random folder]\[random name].exe*.

En la imagen 4.36 puede verse uno de los ejecutables maliciosos detectados en el path *C:\Windows\Users\IEUser (usuario empleado)\AppData\Roaming*. Queremos reseñar que en todos los casos el malware se ha alojado en dicho path.

Hemos empleado dos programas de detección de malware para que nos muestren sus descubrimientos en la máquina zombi donde esta alojado el Bot1. Hay que tener en cuenta que estas herramientas se basan en "firmas" (hashes) y suelen requerir una conexión a internet para su empleo.

La primera ha sido *malwarebytes* que nos ha dicho que nuestro dispositivo no tenia ninguna

infeccion o malware, aunque mas adelante (al quedarse instalada en el equipo) si detectó el software malicioso y lo puso en cuarentena.

La segunda (HitmanPro) los ha detectado y ha coincidido con mis sospechas anteriores. En la imagen 4.37 se puede ver una muestra de lo encontrado por *HitmanPro*.

La aplicación nos ha detectado como archivos maliciosos *tanga.exe*, *egpa.exe*, *drvhandler.exe* y las copias que teníamos del Bot Zeus y el Bot de Blue Botnet. Así pues esta herramienta demuestra su eficacia y confirma nuestras sospechas con “tana” y “egpa”.

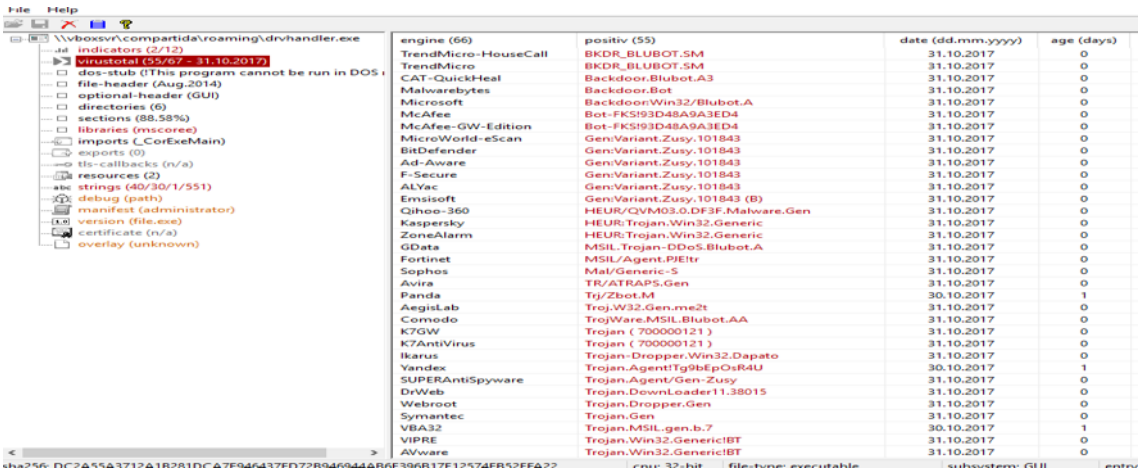


Figure 4.38: Detección de malware -PSEstudio-.

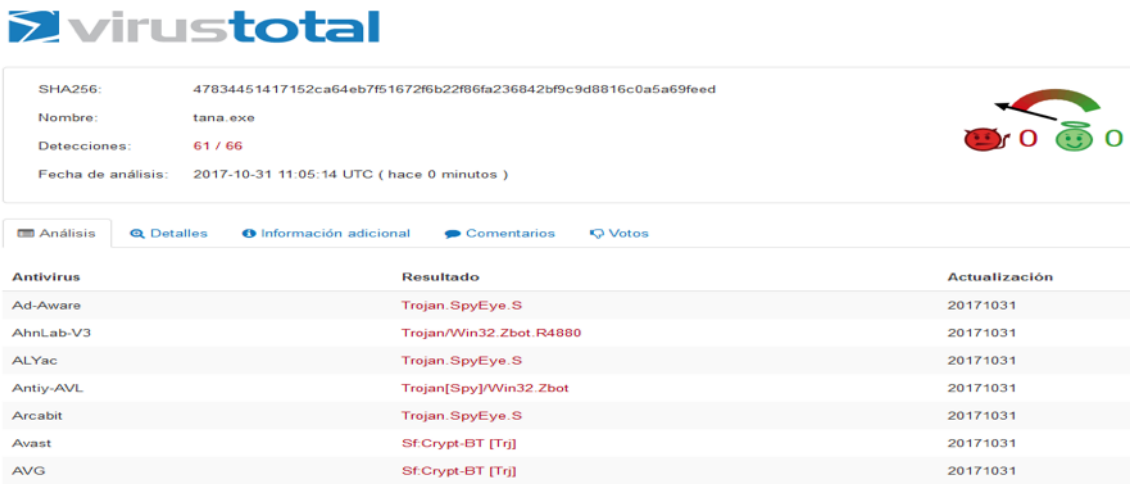


Figure 4.39: Detección de malware -Virustotal.com-.

Vamos a emplear dos herramientas mas una relacionada con la dependencia entre archivos y dll y la otra con análisis de malware de un archivo (Dependency Walker y PSEstudio). Ambas se ejecutan de manera portable y solo requieren del archivo sospecho (en nuestro caso tana.exe, egpa.exe y drvhandler.exe).

Dependency Walker es una herramienta gratuita que escanea cualquier módulo de Windows de 32 bits o 64 bits (exe, dll, ocx, sys, etc.) y crea un diagrama de árbol jerárquico de todos los módulos dependientes. Para cada módulo encontrado, enumera todas las funciones que exporta ese módulo, y cuáles de esas funciones están siendo llamadas por otros módulos. Otra vista muestra el conjunto mínimo de archivos necesarios, junto con información detallada sobre cada archivo, incluida una ruta completa al archivo, la dirección base, los números de versión, el tipo de máquina, la información de depuración y más cosas. Para nuestros fines no hemos extraído mucha información de ella, ya que no es el objetivo de este trabajo el realizar un análisis en profundidad de este malware.

PEStudio nos ha parecido una herramienta muy útil pues tiene incluso un vinculo con virustotal.com para el análisis de archivos. PEStudio implementa un amplio conjunto de características que está especialmente diseñadas para recuperar cada detalle de cualquier archivo ejecutable. Los resultados se comparan con las especificaciones de Microsoft. Además, el contenido del archivo que se analiza se compara con varias listas y umbrales (blacklist and withelist).

Esta herramienta nos muestra como peligrosos los archivos *tana.exe*, *egpa.exe* y *drhhandler.exe* (vemos la imagen 4.38 del análisis de drhhandler.exe), también nos muestra las dependencias que este ejecutable tiene con sus dll. Esta herramienta compara los hashes obtenidos con los encontrados en virustotal.com.

Por ultimo, se han analizados los archivos ejecutables también en la web virustotal.com. En la imagen 4.39 vemos el análisis de tana.exe.

Para concluir nuestro análisis forense hemos realizado una búsqueda de las claves de registro en busca de tana.exe, egpa.exe y/o drhhandler.exe y hemos encontrado que los archivos relativos a Zeus tienen sus claves de registro en los siguientes path (como se puede ver en 4.40):

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\tana.exe

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\egpa.exe

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\drhhandler.exe

Asi pues para esta versión de Zeus las variables de registro a eliminar o borrar estarán siempre contenidas en "*HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run*".

Por otro lado, hemos encontrado trazas de "Bot.exe" (Bot de Blue Botnet) y de "drhhandler.exe" en los registros que están relacionados con la tecnología RADAR que existe en Windows 7. Dichos registros son:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\RADAR\HeapLeakDetection\DiagnosedApplications\drvhandler.exe

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\RADAR\HeapLeakDetection\DiagnosedApplications\Bot.exe

RADAR [20] es una tecnología de detección de fugas de memoria integrada que permite a los equipos de productos de Microsoft y a terceros descubrir y corregir fugas de memoria al principio del ciclo del producto y después de su lanzamiento (podrían ser versiones Beta).

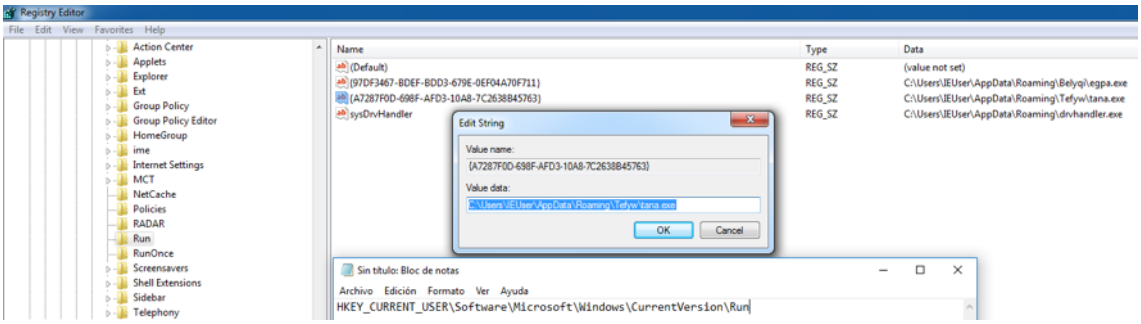


Figure 4.40: Claves de registro maliciosas.

Esto nos indica que para el sistema operativo “algo no va bien” con estos dos ejecutables. Mas concretamente, y sabiendo que una fuga de memoria (memory leaks) se produce cuando un bloque de memoria reservada no es liberada en un programa de computación y que, además, se le suele asociar a programación en C o C++ que realizan esa liberación de memoria nivel manual (otros lenguajes lo hacen automáticamente) se presupone que la programación del Bot se ha podido realizar en C o C++.

4.9.1 Formas de ocultación de Zbot descubiertas

Después de todo el análisis forense realizado hemos verificado que esta versión de Zeus oculta muy bien sus archivos y sus librerias. Para realizar esta ocultación, habitualmente este tipo de malware usa técnicas que se conocen como “Hooking”.

La idea de “Hooking” [15] es la de conseguir que otra función legítima llame al kernel durante el proceso para el malware. Este tipo de comportamiento hace que las aplicaciones e hilos (threads) que son lanzados parezcan provenir de procesos “limpios” haciendo difícil su descubrimiento incluso a través de técnicas forenses. Las formas para poder detectar este tipo de comportamiento seria a través de la comprobación de dichas librerías y procesos con sus hashes (integridad) o a través de ingeniería inversa.

Las técnicas de Hooking que se podrían aplicar son:

- Sustitución de la dirección de la función real (modificando las tablas IAT -Import Address Table- o las tablas SSDT/IDT -Tablas de los servicios de sistema de windows o de interrupciones-).

- Cambio de la función legítima (-splicing- intercepción en el modo kernel con la modificación del cuerpo de la función).

- Sustitución directa de todo el componente de la aplicación del sistema (por ejemplo, sustitución de una biblioteca por una función maliciosa).

También podemos dividirlos por el modo de ejecución de la siguiente manera:

- Métodos de usuario: modificación de tablas IAT -splicing-. Su peculiaridad es que no se puede cambiar nada en el comportamiento del kernel del sistema operativo y sus extensiones.
- Modo Kernel: modificación de tablas SSDT/IDT, intercepción en el modo kernel con la modificación del cuerpo de la función. Con esto, puede modificar la estructura de datos y el código de cualquier parte del sistema operativo y las aplicaciones.

Hay que reseñar que (como se verá mas adelante durante la desinfección con ZbotKiller) Zeus emplea IAT y splicing.

4.9.2 Conclusiones del análisis forense

En el apartado 4.5.3 de este texto se señalaban los emplazamientos de los archivos relacionados con el malware Zeus para las variantes 1, 2, 3 y 4, sin embargo, después de su búsqueda en el equipo infectado, según esas indicaciones, no aparecieron ninguno de los archivos mencionados en dicho apartado. Es por ello que se pueden realizar las siguientes conclusiones:

- Se confirma que la versión de Zeus empleada (2.1.0.1) no coincide con las variantes documentadas 1, 2, 3 y 4, por lo que se trata de una versión mas actual y no documentada (al menos tras las búsquedas de ello realizadas por este alumno).
- Esta version de Zeus funciona (al menos) con los navegadores Explorer, Chrome y Firefox.
- Pueden existir en la misma máquina varias versiones del bot Zeus de diferentes compilaciones (Builder).
- Los dos ejecutables (tana.exe y egpa.exe) son bots Zeus (Zbot) ya que se probó la ejecución de dos builders diferentes para comprobar el punto anterior. Ademas, se ha constatado que en otras máquinas infectadas solo existe un archivo de los dos y el bot funciona perfectamente. Sin embargo, drhhandler.exe es común a todas las máquinas infectadas.
- Zeus utiliza técnicas de “Hooking” avanzadas que le permiten la ocultación y ejecución de sus procesos a nivel kernel. Esto dificulta su limpieza y eliminación.
- Se podria inferir que el leguaje de programación utilizado en la programación de los bots es C o C++.
- Los nombres de carpeta y archivos de este malaware son diferentes según el bot analizado y aleatoriamente creados excepto (como ya se comentó) para drhhandler.exe.

- Esta versión de Zeus no cambia las claves de registro habituales como winlogon.
- Este malware se ejecuta a través de una aplicacion como explorer.exe que es siempre arrancada con la máquina. Es decir, hace que explorer.exe arranque los archivos maliciosos.
- Zeus 2.1.0.1 intenta ocultarse eliminando vínculos entre los procesos.
- Zeus crea o llama multitud de librerías y crea mutex específicos para su funcionamiento.
- Ciertas herramientas anti-malware no lo detectan.
- Por todo ello su eliminación es mas compleja que con anteriores versiones. Ademas, no necesita privilegios de usuario para infectar un equipo.

4.10 Detección, proteccion e inhabilitacion de Zeus

Ya se comento las medidas de detección , protección e inhabilitación de malware tipo botnet en el apartado 3.5.2. Aquí queremos añadir solo las especificas para la botnet Zeus en el caso práctico que se ha realizado.

Los mayores actores de las medidas de detección e inhabilitación son las partes afectadas (como bancos o multinacionales), los CSIRTs que existen a nivel mundial, gobiernos y empresas dedicadas a las seguridad entre otros. Algunas de las mas interesantes son:

- https://zeustracker.abuse.ch/index.php (Específica para Zeus con mapas interasantes)
- http://securityengineer.pro/index.html
- https://www.spamcop.net/
- http://www.malwaredomainlist.com/
- http://www.team-cymru.org/
- https://www.evilfingers.com/index.php
- http://www.rootkitanalytics.com/
- https://www.autoshun.org/
- https://www.shadowserver.org/wiki/ (Una de las mejores)
- https://support.kaspersky.com/sp/viruses/disinfection

4.10.1 Desinfeción de un equipo infectado con Zbot

En este apartado se quiere realizar una desinfección de manera manual (con la información obtenida del análisis de las máquinas infectadas) y otra a través de una herramienta como ZbotKiller. Esta última herramienta no necesiata conexión a internet para su funcionamiento.

4.10.1.1 Desinfección manual

Para realizar esta desinfección se debe de seguir los siguientes pasos:

- Arranque del dispositivo “en modo seguro” (en windows con F8 al arrancar).
- Búsqueda y borrado de todos los archivos en el path *C:\Windows\Users\IEUser (usuario empleado)\AppData\Roaming*
- Archivos a eliminar –> [ramdom].exe y drhhandler.exe.
- Búsqueda y borrado de las claves de registro relacionadas con esos archivos en el path *HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run*.
- Comprobación de que la máquina no está infectada con un antivirus o antimalware o a través delintento de conexión con el servidor C&C.

La desinfección realizada fué satisfactoria, no observándose traza del malware.

4.10.1.2 Desinfección automática -herramienta-

Para realizar esta desinfección se ha empleado la herramienta “ZbotKiller” [19] desarrollada por Kaspersky y orientada a la bśqueda de bots Zeus.

Esta herramienta se puede utilizar de manera local o remota y una vez ejecutada actúa automaticamente, buscando y eliminando no solo los archivos infectados sino detctando los “IAT Hooks” y “Spliy Hooks” y asi desligando esos “Hooks” realizados y limpiando los “Threads infectados”. Los elementos de enlace (Hooks) ya fueron explicados anteriormente. En la imagen 4.41 se ve el proceso de desinfección y todos los elementos infectados encontrados y limpiados.

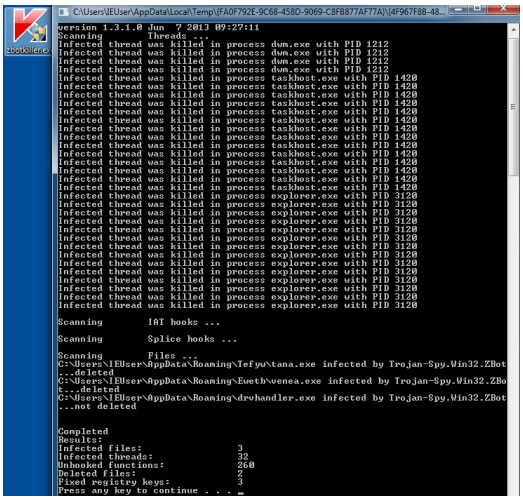


Figure 4.41: Limpieza Zbot con ZbotKiller [19].

debe de seguir los siguientes pasos:

- Descargar la herramienta (ejecutable .exe). La ultima version libre y gratuita es la 1.3.1.0 de 07/12/2015.
- Ejecutar la herramienta con privilegios de administrador.

- Comprobación de que la máquina no está infectada con un antivirus o antimalware o a través delintento de conexión con el servidor C&C.

La desinfección realizada fué satisfactoria, no observándose traza del malware.

4.11 Conclusiones del estudio práctico

Este capitulo se ha basado en la realización práctica de ensayos para conocer el funcionamiento de la botnet Zeus (Zeus 2.1.0.1/Server C&C), su empleo, su infección, sus posibiliddes maliciosas y su análisis a través de un volcado de memoria RAM.

La realización práctica ha sido complicada, ya que de la versión de Zeus empleada apenas hay documentación y pocos o ningún análisis en profundidad.

Sin embargo, gracias al empleo de escenarios simulados hemos aprendido no solo la taxonomía y funcionalidades una botnet tan importante y potente como Zeus sino a realizar un estudio de manera forense de una máquina infectada que nos ayuda a comprender mejor el funcionamiento "a bajo nivel" de este malware tan complejo.

La versión de la botnet empleada (Zeus 2.1.0.1) puede ser detectada y eliminada accediendo y borrando los archivos con nombres aleatorios que se encuentran en el path C:\Windows\Users\IEUser (usuario empleado)\AppData\Roaming. Los archivos implicados y que hay que eliminar son *drhhandler.exe* y los que se encuentra en carpeta aleatorias como "Tefyw". Además, es aconsejable eliminar las claves de registro como ya se ha explicado anteriormente.

CHAPTER 5

CONCLUSIONES Y TRABAJOS FUTUROS

Saltar rápidamente a conclusiones rara vez conduce a felices aterrizajes. S. Siporin.

Al principio de este trabajo se propuso una serie de objetivos a alcanzar durante todo el desarrollo del mismo. Los objetivos planteados eran:

1. Explicar la taxonomía de infección, empleo y gestión de las botnets, así como su impacto en internet.

2. Realizar un ejemplo práctico (a través de la implementación de un escenario) verificando su potencial, forma de ejecución y posibles medios de protección a través de datos empíricos.

3. Obtener unas conclusiones finales en las que se puedan reseñar los tipos de defensas que se podrían establecer ante este tipo de ataques y/o empleos de métodos de detección.

4. Por último, obtener vías de estudio para un trabajo futuro vinculado a los botnets.

Es de reseñar que se han cumplido todos ellos y que, además, durante el desarrollo de este trabajo se propuso uno adicional, sería el realizar un análisis de un equipo infectado para comprender mejor como se oculta este malware y como podría ser detectado. Este objetivo también se cumplió durante la parte práctica.

5.1 Conclusiones obtenidas a través del ensayo práctico

Las conclusiones obtenidas en este trabajo son innumerables, dado el hecho que durante este texto se ha comenzado por un "estado del arte" de las botnets para ir conociendo como actúan este tipo de redes. Mas adelante hemos conocido casos reales y su impacto en la sociedad, llevándonos del terreno de lo intangible o teórico al terreno real. En los siguientes capítulos se ha explicado de manera analítica los tipos de redes, formas de comunicación y su implementación desde sus primeras apariciones hasta nuestros días. La ultima fase de este trabajo ha sido la implementación práctico de varios ensayos en los que no solo se ha probado el uso de estas redes en un entorno simulado y próximo a uno real, sino que se ha aprendido sus formas de ocultación, la flexibilidad en su empleo, la peligrosidad de sus acciones y los medios de infección usados asi como la defensa ante esta amenaza.

Antes de empezar a detallar las conclusiones obtenidas durante todo este trabajo quiero poner de manifiesto la satisfación obtenida durante la realización de este trabajo, el cual me ha llevado no solo a tener una perspectiva mas real de este tipo de malware, sino a conocer mas en profundidad medios de ocultación en sistemas operativos, formas de ocultación en al red a través de técnicas fast-flux, técnicas de análisis forense, conocimiento de herramientas diversas de malware y anti-malware, conocer la labor de los cibercriminales y el mercado negro existente con las botnets, pero sobre todo la percepción real de algo latente que existe, que no es una quimera o algo oculto como un fantasma (como reza el titulo) sino que esta entre nosotros desde hace ya muchos años, que vino para quedarse y que promete quedarse mucho mas tiempo.

Entre las conclusiones obtenidas podemos destacar las siguientes:

- Las botnets son muy desconocidas en la vida cotidiana y casi se asocia a algo teórico que no afecta a la mayoría de los equipos (o personas).

- Este tipo de redes casi nacieron con la aparición de internet a través del uso de las redes IRC, aunque su uso inicial fué con objetivos legítimos.

- Los medios de comunicación empleados por las botnets se basan en los mismos estándares que cualquier comunicación legítima e incluso en módulos o partes de código obtenidos de aplicaciones legales diseñadas para otros fines.

- El concepto de botnet y el de troyano, a veces, es difuso sobre todo en cuanto se refiere a los bots desplegados en las máquinas zombis.

- Las botnets puede emplear cualquier tipo de topología utilizada actualmente, como son el modelo cliente-servidor y el modelo P2P.

- El mundo criminal emplea activamente, bien a través de su implementacion o bien a través de su compra o alquiler, este tipo de redes para cometer sus crímenes.

- Nadie está a salvo de esta amenaza. Siempre que un usuario sea descuidado o que existan bugs en los equipos informáticos toda máquina es susceptible de ser infectada por una botnet.

- Este tipo de herramientas son interesantes no solo en el mundo criminal sino en el empresarial y por países que pueden emplearlas para fines comerciales, políticos o estratégicos. A este respecto se sospecha que ciertas botnets hayan podido ser auspiciadas por gobiernos.

- Como cualquier otro malware una botnet puede utilizar para propagarse vulnerabilidades de día 0 (zero-day) que son las mas temidas y posiblemente generadas o conocidas e incluso ocultadas por empresas y/o gobiernos.

- Es muy difícil poder encontrara al "BotMaster" pero aun mas difícil al creador de la botnet. Incluso si la red es detectada e inhabilitada el localizar a los autores resulta una labor muy compleja y, a veces, el poder demostrar sus acciones ante la justicia es imposible.

- El diseño e implementación "desde cero" de un artefacto como una botnet requiere de un periodo muy largo de tiempo y (posiblemente) del empleo de un equipo de personas. Sin embargo, hoy en día los cibercriminales reutilizan parte del trabajo hecho y no empiezan la labor desde su inicio. Tanto en uno como en otro caso el diseñador y realizador no solo debe de conocer muy bien los protocolos de comunicación en redes, sino, además, el funcionamiento de los sistemas operativos infectados en profundidad.

- La forma de ocultación de estas redes en internet cada día es más compleja poniendo en serías dificultadas no solo su detección sino incluso su eliminación.

- El cifrado que estas redes utilizan, dificulta saber su forma de trabajo, su comunicación y su funcionamiento a nivel del S.O. Las técnicas de ingeniería inversa se complican al estar la información cifrada.

- La forma en la que este malware se oculta y esconde sus actividades en los equipos infectados evoluciona constantemente y solo con técnicas de análisis forense somos capaces de su descubrimiento.

- La mayoría de las herramientas de detección y desinfección se basan en firmas y búsqueda de tráfico sospechoso en la máquina infectada. Esto nos pone de manifiesto la necesaria colaboración entre organismos, empresas, gobiernos y particulares en la detección y conocimiento de este tipo de malware.

- La detección y eliminación de estas redes se produce por que son previamente "conocidas". Cualquier botnet desconocida o con patrones de comunicaciones difíciles de detectar podría estar operando durante años sin percibirlo.

- Este tipo de redes no siempre son el fin en si mismo, sino que a veces se utilizan como medio de transporte de otro malware asociado como pasó con cryptolocker.
- El mundo de la descarga ilegal, de las aplicaciones gratuitas y del correo electrónico favorece su difusión.
- El software malicioso empleado en la parte práctica no puede ser verificado por medio de hash con lo que puede ser una vía para la propia infección del que lo emplea. El uso de hashes para la comprobación de software gratuito es crucial, aunque la mayor parte de los usuarios de internet no saben emplear esta herramienta.
- Aunque ya existe una colaboración manifiesta, queda mucho trabajo por desarrollar para poder ganar la lucha contra este tipo de malware, si bien estimo que nunca dejarán de operar estas redes. Pero, sin duda la única manera de ir venciendo en esta lucha se basa en la colaboración entre las partes.
- Por último, queremos remarcar que el conocimiento técnico para desarrollar una nueva botnet requiere de tiempo, esfuerzo, apoyo (personal y económico) y mucho, mucho tiempo.

5.2 Trabajos futuros sobre botnets

Este alumno a desarrollado un trabajo relacionado con las botnets a nivel teórico y práctico pero con una alcance delimitado debido al limite de tiempo y de recursos (muy importante). Aunque está muy satisfecho de lo obtenido, la realidad es que las vías de trabajo futuro con respecto a las botnets son enormes quedando, por tanto, mucho camino que recorrer en este campo. Como propuesta de trabajos futuros a desarrollar en este mundo se quieren proponer, entre otros, los siguientes:

- Realizar un estudio práctico de una botnet de tipo P2P (para sistemas de 64 bits).
- Realizar un estudio práctico de una botnet híbrida como Necurs (para sistemas de 64 bits).
- Realizar un estudio práctico de una botnet a través de dispositivos de seguridad como firewall e IDS/IPS para poder adaptar, crear y configurar reglas especificas para su detección.
- Estudio práctico de propagaciones “laterales” en una botnet con la creación de scripts o batchs mas avanzados que se ejecuten en las máquinas zombis.
- Estudiar prácticamente la infección (si es posible) entre máquinas virtuales o dockerizadas por una botnet, o de al menos, el impacto que pueden producir en el resto de los servicios virtualizados que corran en el mismo equipo anfitrión (como por ejemplo el bloqueo de comunicaciones entre todas las máquinas virtuales).

- Realizar un estudio práctico de la información cifrada que envían los bots para poder ser generada en texto claro.
- Estudio práctico y mas en profundidad sobre el desarrollo de mutex o código propio y su forma de despliegue en la máquina victima para pode ver patrones de infección detectables.
- Realizar un estudio práctico de una implementación de un sistema de comunicación a través del empleo de la técnica DGA (Domain Generation Algorithm). Una forma de realizar la parte práctico sería a través de honeypots que hagan (en redes P2P) de nodos de una botnet.
- Búsqueda de información e investigación sobre el empleo de técnicas de Inteligencia Artificial (Data Mining) para poder clasificar patrones en las comunicaciones globales de internet con el fin de encontrar botnets.
- Hacer un estudio práctico sobre los patrones de comunicación detectables de este tipo de redes en internet.

Bibliography

[1] *Advanced ddos attack & corresponding technical trends*.
https://ettrends.etri.re.kr/ettrends/162/0905002171/0905002171.html.

[2] *An advanced hybrid peer-to-peer botnet*.
https://www.usenix.org/legacy/event/hotbots07/tech/full_papers/wang/wang_html/.

[3] *Algoritmo de generación de dominios*.
http://www.davidromerotrejo.com/2015/03/algoritmo-de-generacion-de-dominios.html.

[4] *Atrivo/intercage's disconnection briefly disrupts spam levels*.
http://www.zdnet.com/article/ivointercages-disconnection-briefly-disrupts-spam-levels/.

[5] *Botnet communication topologies*.
http://www.technicalinfo.net/papers/PDF/WP_Botnet_Communications_Primer_(2009-06-04).pdf.

[6] *Botnets*.
https://www.shadowserver.org/wiki/pmwiki.php/Information/Botnets.

[7] *Botnets unearthed, the zeus bot*.
http://resources.infosecinstitute.com/botnets-unearthed-the-zeus-bot/.

[8] *A brief look at zeus/zbot 2.0*.
https://www.symantec.com/connect/blogs/brief-look-zeuszbot-20.

[9] *Creative commons españa*.
https://creativecommons.org/licenses/by-nc-sa/3.0/es/.

[10] *Ddos análisis de ataques coordinados*.
https://www.dragonjar.org/ddos-analisis-de-ataques-coordinados.xhtml.

[11] *Ddos attack*.
https://nets.ec/Ddos_attack.

[12] *Detección de botnets basada en dns*.
https://www.certsi.es/blog/botnets-dns.

[13] *Detección de botnets mediante análisis de flujo*.
https://www.certsi.es/blog/botnets-analisis-flujo.

[14] *Detección de botnets mediante análisis de paquetes*.
https://www.certsi.es/blog/botnets-analisis-paquetes.

[15] *Driver to hide processes and files*.
https://www.apriorit.com/dev-blog/235-splicing-to-hide-files-and-processes.

[16] *Global botnet threat activity map*.
http://vn.trendmicro.com/vn/security-intelligence/current-threat-activity/global-botnet-map/index.html.

[17] *The history of the botnet*.
http://countermeasures.trendmicro.eu/the-history-of-the-botnet-part-i/.

[18] *Honeypots*.
https://www.shadowserver.org/wiki/pmwiki.php/Information/Honeypots.

[19] *How to secure your computer from malicious programs of trojan-spy.win32.zbot family*.
https://support.kaspersky.com/viruses/disinfection/2020#block3.

[20] *Inside windows 7: Radar - windows automatic memory*.
https://channel9.msdn.com/Shows/Going+Deep/RADAR-Windows-Automatic.

[21] *Introduction to botnets*.
https://secure.nic.cz/files/labs/CSIRT-20091102-OS-Botnet-CEPOL.pdf.

[22] *La firma electrónica*.
http://roble.pntic.mec.es/jprp0006/tecnologia/4eso_informatica/tramites_online/firma_electronica.htm.

[23] *Live cyber attack threat map*.
https://threatmap.checkpoint.com/ThreatPortal/livemap.html.

[24] *The malicious top ten*.
http://apac.trendmicro.com/apac/security-intelligence/current-threat-activity/malicious-top-ten/.

[25] *Mapping europe's top"botspots"*.
https://uk.norton.com/tools/bots/index.html.

[26] *Necurs.p2p - a new hybrid peer-to-peer botnet*.
https://www.malwaretech.com/2016/02/necursp2p-hybrid-peer-to-peer-necurs.html.

[27] *New zeus source code based rootkit available for purchase on the underground market*.
https://www.webroot.com/blog/2013/03/14/new-zeus-source-code-based.

[28] *'operation tovar' targets 'gameover' zeus botnet, cryptolocker scourge*.
https://krebsonsecurity.com/tag/p2p-botnet/.

[29] *Peer-to-peer botnets for beginners*.
https://www.malwaretech.com/2013/12/peer-to-peer-botnets-for-beginners.html.

[30] *Proofpoint uncovers internet of things (iot) cyberattack*.
https://www.proofpoint.com/us/proofpoint-uncovers-internet-things-iot-cyberattack.

[31] *Red de bots mariposa*.
https://www.pandasecurity.com/spain/mediacenter/malware/red-de-bots-mariposa/.

[32] *Reversal and analysis of zeus and spyeye banking trojans*.
https://ioactive.com/pdfs/ZeusSpyEyeBankingTrojanAnalysis.pdf.

[33] *Servicio antibotnet*.
https://www.osi.es/es/servicio-antibotnet.

[34] *Spam volumes: Past & present, global & local*.
https://krebsonsecurity.com/2013/01/spam-volumes-past-present-global-local/.

[35] *Técnicas emergentes de evasión: Domain shadowing*.
https://www.certsi.es/blog/domain-shadowing.

[36] *Top 10 de ataques ddos (denial of service)*.
https://openwebinars.net/blog/top-10-de-ataques-dos-denial-of-service.

[37] *The top 10 most dangerous malware that can empty your bank account*.
https://heimdalsecurity.com/blog/top-financial-malware/.

[38] *Ufonet - ddos botnet via web abuse*.
https://ufonet.03c8.net/.

[39] *Zeus banking trojan report*.
https://www.secureworks.com/research/zeus/.

[40] *Zeus complete tutorial-building bot & install cpanel*.
http://www.freetrojanbotnet.com/node/11.

[41] *Zeus crimeware kit*.
https://www.akamai.com/uk/en/multimedia/documents/state-of-the-internet/zeus-zbot-malware-crimeware-kit-cybersecurity-threat-advisory.pdf.

[42] *Zeus: King of the bots*.
http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/zeus_king_of_bots.pdf.

[43] *Zeus trojan - memory forensics with volatility*.
http://www.behindthefirewalls.com/2013/07/zeus-trojan-memory-forensics-with.html.

[44] *Zeus trojan virus, chronological background*.
https://sensorstechforum.com/remove-zeus-trojan-virus/.

[45] *Zeus v2 malware analysis*.
http://sysforensics.org/2012/04/zeus-v2-malware-analysis-part-ii/.

[46] E. ALPARSLAN, A. KARAHOCA, AND D. KARAHOCA, *Botnet detection: Enhancing analysis by using data mining techniques*, in Advances in Data Mining Knowledge Discovery and Applications, A. Karahoca, ed., InTech, Rijeka, 2012, ch. 17.

[47] R. J. ENBODY, A. K. SOOD, AND R. BANSAL, *Cybercrime: Dissecting the state of underground enterprise*, IEEE Internet Computing, 17 (2013), pp. 60–68.