Botnets

PID_00255336

Joaquin Garcia Alfaro

Universitat Oberta de Catalunya

Índice

Introducción

Una botnet consiste en una red de equipos informáticos infectados por un atacante remoto, que los controla de manera distribuida con fines tanto malintencionados como lucrativos. Dichos equipos infectados componen una red de robots (agentes software) al servicio del atacante. El atacante se convierte así pues en operador de una compleja y potente red cuyos servicios serán finalmente vendidos a organizaciones de todo tipo. Por ejemplo, los servicios de la red podrán ser vendidos a organizaciones criminales para la ejecución de ataques coordinados a gran escala, tales como denegaciones de servicio distribuidas, campañas de *spamming*, venta de productos ilegales, etc.

En este módulo, presentamos el origen y las técnicas actuales que hacen posible la existencia de estas redes. Introduciremos las fases necesaria para su construcción, estudiaremos las arquitecturas, los protocolos y los modelos de comunicación que las hacen posibles. Mostraremos también algunas de las técnicas empleadas por los operadores de las botnets para que los equipos infectados pasen desapercibidos y sus redes no sean, por lo tanto, desarticuladas. Por último, veremos algunos ejemplos concretos de botnets que han sido descubiertas y mostraremos algunos datos sobre el modelo económico que garantizó su existencia.

Objetivos

Los objetivos que el estudiante debe haber conseguido después de estudiar los contenidos de este módulo son los siguientes:

1. Entender qué son las botnets y saber cómo surgieron.
2. Comprender el modelo de propagación de una botnet tradicional.
3. Conocer el modelo de comunicación y colaboración entre robots y operadores.
4. Ver algunas de las técnicas utilizadas por los operadores de una botnet para proteger sus equipos y evitar que la red de robots sea desarticulada.
5. Conocer el modelo económico que promueve la creación y el mantenimiento de una botnet, así como las actividades asociadas.

1. Antecedentes e inicios de la amenaza

Origen del término

El término *botnet* proviene de la superposición de dos palabras inglesas: roBOT y NETwork. Una traducción al castellano sería, por lo tanto, red de robots.

Una botnet (también conocida como red de equipos robot o red de *zombies*) es entendida, hoy en día, como un conjunto de equipos informáticos conectados a Internet, cuyos recursos (tales como memoria, ejecución de procesos, sistema de ficheros y conexiones de red) son controlados, a distancia, sin que sus usuarios y/o propietarios sean conscientes. Los operadores de una botnet (a menudo conocidos bajo el sobrenombre de *botmasters* o *botherders*) crean la red mediante la propagación de código malicioso, que infecta los recursos de los futuros equipos de la botnet y garantiza el control permanente de los estos.

Una vez infectados, los equipos de la botnet son vistos por sus operadores como un conjunto de robots software al servicio de los clientes de la botnet. Estos clientes podrán alquilar los equipos para realizar actividades tales como campañas de *spam*, denegaciones de servicio distribuidas, almacenamiento de contenidos multimedia ilícitos, etc. La mayoría de los equipos infectados, y futuros robots de la botnet, suelen ser ordenadores domésticos, a menudo activamente conectados a Internet durante largos períodos de tiempo (horas, días, semanas) y con unos níveles de protección (en cuanto a seguridad se refiere) bastante bajos.

El operador de la botnet (generalmente, quien se encargó de encontrar e infectar los equipos) enviará periódicamente mensajes de control vía protocolos tradicionales como, por ejemplo, IRC (*Internet relay chat*) o HTTP (*hypertext transfer protocol*). Los robots de la botnet pueden incluso ser reprogramados por terceras partes (a veces, facciones de comunidades maliciosas diferentes) con el objetivo de ampliar sus dominios o retomar los recursos de botnets ya existentes.

Actualmente, una botnet es considerada una red de equipos infectados que ofrece servicios fraudulentos. Dichos equipos, a menudo citados como *robots*, *zombies*, o simplemente *agentes* de la botnet, son controlados a distancia, de forma distribuida, por uno o varios operadores (referenciados en la literatura como *botmasters* o *botherders*).

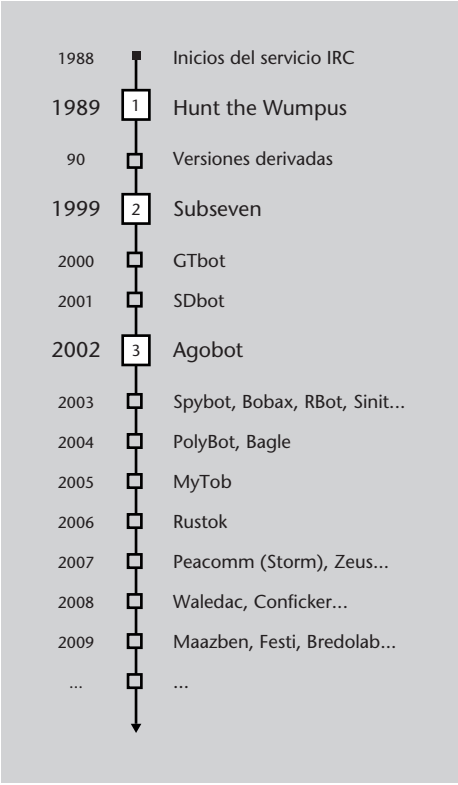
Sin embargo, las botnets no siempre han estado ligadas a servicios ilícitos. En el subapartado siguiente, veremos cuáles fueron los inicios de estas redes fraudulentas. Trataremos, aunque de manera no exhaustiva, la evolución que

han tenido sus servicios a lo largo de los últimos años. Descubriremos que las botnets no siempre han tenido la connotación negativa atribuida en la actualidad. Veremos que, en realidad, las botnets fueron concebidas para automatizar el mantenimiento y la actualización de algunos servicios pioneros de Internet.

1.1. Breve historial sobre botnets conocidas

Las botnets aparecen a finales de la década de los ochenta. La figura 1 resume, de manera no exhaustiva, el nombre y la fecha (aproximada) de aparición de algunos de los despliegues de botnets que más repercusión han tenido desde entonces.

Figura 1



Hunt the Wumpus

Relacionado con la primera botnet de la historia, Hunt the Wumpus es videojuego también famoso por contener algunas de las técnicas pioneras en materia de inteligencia artificial. Se trata de una aventura conversacional en red a través de una consola de comandos que permite lanzar las acciones del usuario en el juego. La versión original del juego, programada en Basic, se remonta a la década de los setenta. El jugador debe recorrer una estructura geométrica (similar a un dodecaedro) compuesta por habitaciones y túneles. Similar al mito del Minotauro, dentro de la estructura se esconde un misterioso monstruo, Wumpus, cuyo objetivo es encontrar a los jugadores y devorarlos. Adicionalmente, algunas de las habitaciones contienen trampas mortales (pozos sin fondo, murciélagos gigantes, etc.).

La aparición de la primera botnet está íntimamente relacionada con la invención del protocolo IRC (*Internet relay chat*) y con la versión en red de *Hunt the Wumpus* a través de canales IRC. De hecho, y como sucede con muchas otras tecnologías fraudulentas asociadas a Internet, el origen de esta primera botnet, totalmente lícita e inofensiva, fue concebida para la automatización de tareas de gestión virtuales de canales IRC. Los equipos de esta primera botnet tenían por objetivo la construcción y el mantenimiento de procesos asociados a juegos para usuarios IRC. Los robots de la plataforma debían estar disponibles las 24 horas del día para ofrecer a los usuarios la posibilidad de jugar con ellos. Rápidamente, y de manera espontánea, redes similares fueron desplegadas para dar soporte a operadores de otros servicios.

Un aspecto importante en el desarrollo de estos precursores de las botnets actuales es la capacidad de crear un canal de comunicación entre los operadores y los robots, así como la inclusión de mecanismos de control de acceso para evitar que terceras partes pudieran tomar el control de los equipos de la red de gestión. Por ello, encontramos en esta época una arquitectura basada en canales de control, a través de los cuales el operador podrá comunicar instrucciones de gestión, tales como inicialización de servicios, reanudación de tareas, operaciones de actualización y mantenimiento de versiones. Estos robots fueron, así pues, evolucionando desde simples programas autónomos capaces de jugar y entrener a internautas, hacia gestores automatizados de tareas y lanzamiento de nuevas aplicaciones al servicio de terceras partes. Es común encontrar en el código fuente de estos robots de finales de los noventa la posibilidad de creación de cuentas de usuarios con privilegios estratificados, así como la inclusión de consolas de comando y la posibilidad de ejecución de macros y *scripts* por parte de usuarios con suficientes privilegios.

Lectura recomendada

Una de las mejores lecturas para entender la evolución de las botnets es el artículo "The Evolution of Malicious IRC Bots", publicado por Symantec y escrito por John Canavan. En ella podréis encontrar un segundo caso parecido al despliegue de Subseven, conocido bajo el sobrenombre de **Pretty Park**. Al igual que el caso relacionado con el despliegue a gran escala de Subseven, Pretty Park se caracteriza por la instalación de *malware* de tipo troyano, que permitirá al operador de la infección un control total sobre los equipos que compondrán la futura botnet.

La primera utilización del término *botnet* se remonta a 1993, basándose en la construcción de redes repetidoras del servicio IRC mediante el control de equipos informáticos ordinarios, conectados a Internet. El control de dichos equipos para formar la red de servidores de IRC se basaba en la utilización de los comandos del propio protocolo. Desde esta primera utilización en 1993, las botnets han evolucionado hoy día hacia completas redes de equipos informáticos infectados y controlados por los operadores que propagaron la infección.

Hasta finales de la década de los noventa no podemos encontrar la aparición de las primeras botnets con connotaciones fraudulentas o malvadas. Uno de los primeros casos relevantes que debemos destacar es el despliegue de robots basado en la infección a gran escala del gusano IRC/Jobbo y la posterior instalación en las máquinas infectadas de la herramienta **SubSeven**. El gusano IRC/Jobbo tuvo como vector de transmisión la explotación remota de errores de programación en clientes de IRC de la época (mayoritariamente, el cliente mIRC para sistemas MS Windows). Mediante la explotación de vulnerabilidades, y la posterior escalada de privilegios, el resultado fue la construcción de una red de equipos controlados mediante la inyección en las víctimas de *malware* de tipo troyano. La herramienta instalada en dichos equipos, Subseven, ofrecerá al operador de la botnet un control de administración total sobre cada máquina infectada.

Además de herramientas tradicionales que podemos encontrar en otros troyanos y *rootkits* de la época (tales como la ocultación de procesos en ejecución), el caso Subseven también se caracterizó por la instalación de nuevas funcionalidades para el robo de contraseñas de correo almacenadas en los equipos infectados (utilizadas posteriormente para propagar la infección, o en campa-

ñas de *spamming*), robo de los nombres de usuario y contraseñas de servicios de mensajería instantánea, capacidad de reconfiguración de los parámetros de red, almacenamiento de ficheros, descarga automática de aplicaciones, incorporación de herramientas de encaminamiento para redireccionar aplicaciones e incorporación de nuevos servidores y clientes IRC en las víctimas (ocultos, naturalmente, del espació de gestión de los administradores de los equipos infectados). Otra de las características importantes que cabe destacar del caso de Subseven es la instalación de herramientas para la construcción y el mantenimiento de canales de control para la ejecución de posteriores ataques distribuidos de denegación de servicio. Dichos ataques requieren un modelo distribuido de equipos dispuestos a ser sincronizados por parte de uno o más atacantes para el envío conjunto de ataques DoS contra un mismo objetivo. El objetivo de la botnet resultante era, así pues, la utilización de equipos armados con suficientes herramientas de ataque, con suficientes privilegios de administración y, a ser posible, con acceso a redes con un gran ancho de banda. El objetivo era igualmente borrar las huellas originales del origen del ataque, haciendo que el tráfico de control que desencadenaba los ataques pasara desapercibido entre cientos o millares de sistemas ejecutando de manera sincronizada las distintas etapas del ataque desde distintas redes.

Ataque DoS

Un ataque de **denegación de servicio** (del inglés, *Denial of Service* o DoS) agrede la disponibilidad de acceso de la víctima a un recurso (en el caso de ser un usuario final); o de los usuarios que pretenden acceder a los recursos ofrecidos por la víctima (en el caso de tratarse de un servidor de servicios). Es, por lo tanto, la apropiación exclusiva de un recurso o servicio con la intención de evitar cualquier acceso a terceras partes.

El siguiente caso importante que hay que destacar es **Agobot** (también conocida bajo el nombre Gaobot). Esta botnet se basa, de hecho, en una mejora de dos despliegues anteriores (posiblemente entre el 2000 y el 2001) bautizados bajo el sobrenombre de GTbot y SDbot. La botnet Agobot, cuyo despliegue fue posiblemente a mediados del 2002, sienta las bases definitivas en el desarrollo y ciclo de vida actual de las botnets. Agobot introduce un diseño modular y la mayoría de las funcionalidades que se encuentran aún hoy en las botnets más recientes. Las herramientas relacionadas con la gestión de Agobot se componen de tres módulos independientes: un primer módulo encargado de garantizar comunicaciones IRC de control y la gestión de puertas traseras; un segundo módulo encargado de gestionar las tareas y servicios (por ejemplo, ataques DDoS) proporcionados por los robots de la botnet, y un tercer módulo para garantizar la seguridad del *malware* instalado en las víctimas y evitar una posible desarticulación de los equipos de la botnet.

Lectura recomendada

El artículo "Spamalytics: An Empirical Analysis of Spam Marketing Conversion", publicado por investigadores de la Universidad de California (centros de Berkley y San Diego) trata los posibles beneficios obtenidos por los operadores de la botnet Peacomm (también conocida como Storm), en especial gracias a las campañas de *spamming* encubiertas por los robots de esta botnet.

Agobot también sienta las bases del modelo económico que hace posible la existencia y evolución continua de las botnets. Multitud de estudios han reportado estimaciones sobre las posibles ganancias obtenidas por las actividades encubiertas por Agobot y otras botnet derivadas como Peacomm y Conficker. De hecho, el resto de las botnets aparecidas a partir de Agobot se caracterizan por una mejora continua de las tecnologías empleadas para el mantenimiento y la gestión de los robots. Tales evoluciones son meramente técnicas, pero ponen de manifiesto nuevamente el modelo económico que se esconde tras el despliegue de una botnet, y que permite a las organizaciones que promueven la construcción de estas redes una mejora continua de sus productos. A lo largo de los apartados siguientes trataremos algunas de las técnicas e innovaciones que han introducido dichas botnets durante estos últimos años.

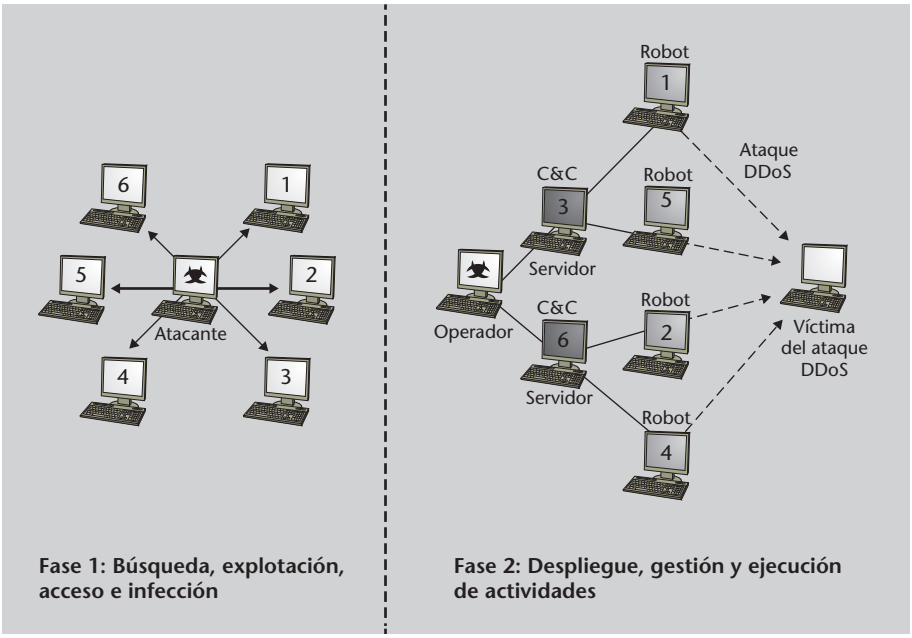
2. Fases previas al despliegue de una botnet

La mayoría de las botnets descubiertas a día de hoy siguen un patrón de despliegue y de actuación muy similar. La figura 2 muestra de manera esquemática la construcción típica de una botnet. El objetivo de la botnet representada en la figura es poner a disposición de un atacante los recursos de multitud de equipos conectados a Internet para finalmente para lanzar un ataque de denegación de servicio contra un equipo concreto. Podemos apreciar dos etapas bien diferenciadas:

1) búsqueda, explotación, acceso e infección de los futuros equipos de la botnet, y

2) despliegue de los canales de comunicación y ejecución de tareas.

Figura 2



En este apartado, nos centraremos en la primera etapa. Veremos algunas de las técnicas utilizadas por operadores de botnets actuales para garantizar una correcta propagación de los futuros equipos que compondrán la red. Dichas técnicas comparten multitud de semejanzas con la mayoría del *malware* ya analizado en módulos anteriores de este material. Por ejemplo, realización de barridos de puertos, de servicios, etc. Estas técnicas son necesarias para inicializar el despliegue, a la vez que para clasificar las vulnerabilidades que permitirán finalmente al atacante infectar y controlar sus futuros equipos. Como

de costumbre, veremos la utilización e inyección de troyanos, de *rootkits* y del código de control necesario para convertir a los equipos en robots de la botnet.

La primera fase en el despliegue de una botnet se puede descomponenr en los pasos siguientes:

1) **Búsqueda e identificación de robots.** En este primer paso, el atacante tratará de recoger información y aprender todo lo que pueda sobre los equipos que tratará de infectar y convertir en robots de la botnet. En especial, tratará de descubrir qué servicios (y sus versiones) son accesibles desde el exterior, con el objetivo de identificar posibles vulnerabilidades y errores de configuración.
2) **Explotación y acceso no autorizado.** En este segundo paso, el atacante tratará de hacerse con privilegios de administrador, abusando de alguna de las deficiencias encontradas durante la etapa anterior.
3) **Infección y toma de control.** Una vez ya producida la explotación de la vulnerabilidad que permitió el acceso al equipo, el atacante tomará el control del equipo y realizará la instalación de aquellas herramientas que permitan que el ataque y futuros acciones/tareas pasen desapercibidas por el propietario/usuario legítimo del sistema.

Dentro de la tercera etapa, de infección y toma de control, se contemplan actividades tales como la eliminación de entradas sospechosas en ficheros de registro, la instalación y modificación de comandos de administración para ocultar la entrada en los sistemas de la red, o la actualización de los servicios vulnerables que ha utilizado para la intrusión (para evitar que terceras partes se introduzcan de nuevo en el equipo), etc. Se contemplan también dentro de esta fase la realización de conexiones necesarias para descargar la imagen completa de nuevas herramientas o actualizaciones de los binarios asociados a la botnet. Una vez realizada esta fase, el equipo infectado pasará a ser considerado un robot (o agente de la botnet) más del sistema controlado por el atacante. Las herramientas instaladas garantizarán también la ejecución de los futuros servicios asociados a la botnet.

A continuación, pasamos a ver con más detalle algunas de las subtareas que acabamos de presentar.

2.1. Búsqueda e identificación de futuros robots

Previamente a la explotación de vulnerabilidades e infección de víctimas, el atacante requiere encontrar y conocer las características de los equipos que

compondrán más adelante la botnet. La fase de recogida de información puede empezar con la simple utilización de aplicaciones de administración que permitan la obtención de información de un sistema, tales como *ping, traceroute, whois, finger, rusers, nslookup, rcpinfo, telnet, dig*, etc.

Por ejemplo, la utilización del comando *ping* en según qué dominios, junto a la existencia de identificadores especiales, podría permitir al atacante determinar la existencia de algunos de los equipos conectados a la red de dicho dominio.

Una vez descubierta la existencia de, como mínimo, uno de los equipos del dominio, el atacante tratará de obtener información relacionada con la topología de red de las víctimas. Herramientas como *traceroute* o *whois* pueden ser utilizadas por el atacante para averiguar el sistema operativo de los distintos equipos de un dominio concreto, así como del resto de los equipos recorridos hasta el sistema de destino. Otras, como *finger* o *rusers* ofrecerán información para descubrir la existencia de usuarios válidos.

El atacante complementará las informaciones obtenidas con el uso de herramientas de *fingerprinting* que buscan la obtención de huellas identificativa de los sistemas encontrados previamente. La huella identificativa que el atacante tratará de obtener de los sistemas encontrados empezará por una referencia a las características de red, en especial de la pila TCP/IP de estos. En primer lugar, esta información le permitirá ajustar o confirmar el sistema operativo que se ejecuta en los equipos encontrados. En segundo lugar, facilitará la clasificación de posibles vulnerabilidades que atacar en la próxima etapa del despliegue.

Las diferentes interpretaciones de un mismo RFC asociado a cualquier protocolo, así como las decisiones finales para su implementación en un sistema operativo concreto, pueden ofrecer información muy valiosa para la identificación de características internas de un equipo conectado a Internet. De hecho, la probabilidad de identificar un sistema operativo mediante un mero análisis remoto de las características de implementación de su pila TCP/IP son muy altas.

El atacante puede, de igual manera, utilizar herramientas existentes para la realización de barrido de puertos, barrido de servicios o escáneres de vulnerabilidades. Así pues, la exploración de puertos TCP o UDP permitirá el reconocimiento de servicios ofrecidos gracias a asociaciones preestablecidas entre identificadores y servicios estándar (por ejemplo, asociación entre el servicio Netbios y el identificador de puerto 139). El atacante podrá utilizar y combinar más adelante esta información en las etapas posteriores para afinar la explotación de vulnerabilidades de aquellos sistemas y servicios que hayan sido encontrados. Es importante tener presente que el atacante realizará par-

Ved también

Para más información sobre los escáneres de vulnerabilidades podéis consultar el apartado 5 del módulo didáctico "Vulnerabilidades en redes".

te de estas actividades desde equipos que han sido previamente infectados. Es incluso probable que el atacante disponga ya de una botnet existente para la realización de su búsqueda de víctimas. Pensemos que la utilización de una botnet previa, compuesta de, aproximadamente, 65.000 robots, sería capaz de realizar un barrido de puertos de una red clase B mediante el simple envío de un único paquete por robot. Por último, no debemos olvidar la posibilidad de realizar la búsqueda de víctimas dentro de una misma red local mediante la utilización de escuchas de red. Las herramientas de tipo *sniffing* contra redes TCP/IP locales son realmente efectivas, puesto que permiten interceptar, almacenar y analizar gran cantidad de información sensible enviada mediante protocolos que no son protegidos mediante cifrado. De esta manera, el atacante puede obtener nuevamente información sobre nombres de equipos y de dominio, cuentas de usuario, claves de acceso o incluso direcciones de usuarios de correo electrónico relacionados con los equipos de la red local (víctimas potenciales para las próximas etapas de explotación de vulnerabilidades e infección complementaria de otros equipos).

Ved también

Para más información sobre las escuchas de red (o *sniffers* en inglés) podéis consultar el subapartado 2.1 del módulo didáctico "Vulnerabilidades en redes".

2.2. Explotación de vulnerabilidades y acceso no autorizado

La mayor parte de la información obtenida en la etapa anterior (búsqueda e identificación de futuros robots) será utilizada para obtener un acceso remoto, y no autorizado, en los equipos encontrados. A menudo, dicho acceso se materializa mediante la explotación de vulnerabilidades no corregidas por parte de los propietarios o administradores de los equipos en cuestión. La mayoría de las vulnerabilidades explotadas están relacionadas con deficiencias de programación en aplicaciones o servicios de red expuestos a Internet. Estas deficiencias, a pesar de ser en el nivel de sistema operativo o de lenguaje, afectan, por lo general, a la seguridad global de una red. La mayoría del código viral en forma de gusano, por ejemplo, consigue introducirse en nuevos equipos a través de la explotación de estas deficiencias.

Recordemos también que la mayor parte de las deficiencias de programación explotadas durante esta fase se deben a situaciones no previstas por los desarrolladores de la aplicación. Por ejemplo:

- entradas no controladas que pueden provocar acciones malintencionadas y ejecución de código malicioso;
- uso de caracteres especiales que permiten un acceso no autorizado al servidor del servicio;
- entradas inesperadamente largas que provocan desbordamientos dentro de la pila de ejecución y que pueden implicar una alteración en el código que ejecutar, etc.

Se espera, por lo tanto, la explotación remota de la seguridad por medio de técnicas ya vistas en módulos anteriores, como, por ejemplo, desbordamientos de *buffer* y explotación de cadenas de formato. Los ataques que permiten explotar este tipo de deficiencias se presentan generalmente en forma de binarios (programas ejecutables) ya compilados para el sistema operativo en el que se está ejecutando la aplicación vulnerable, y conocidos con el sobrenombre de *exploits* remotos.

Un ***exploit** remoto* es un programa, generalmente escrito en C o ensamblador, que fuerza las condiciones necesarias para aprovecharse de un error de seguridad subyacente en una aplicación de red o servicio. El objetivo final suele ser la obtención de un acceso remoto no autorizado en el equipo atacado.

La siguiente lista muestra, de manera no exhaustiva, algunas vulnerabilidades típicas que son analizadas por el código asociado a una botnet durante la etapa de explotación de vulnerabilidades:

- Servicios RPC (*remote procedure call*) erróneos en objetos DCOM (*distributed component object model*) en sistemas operativos Windows XP. La explotación remota es posible mediante el acceso a los servicios vulnerables a través de puertos TCP (por ejemplo, puertos 135, 139, 445 y 593, entre otros).

- Servicios web basados en IIS5 WEBDAV. Explotación remota a través de puertos TCP (por ejemplo, puerto 80).

- Versiones vulnerables de Windows Messenger. Explotación remota a través de puertos TCP (por ejemplo, puerto 1025).

- Implementación ASN.1 vulnerable en servicio Kerberos para sistemas operativos Windows. Explotación remota a través de puertos UDP (por ejemplo, puerto 88).

- Servicio HTTP vulnerable del sistema operativo CISCO IOS en su gama de encaminadores (*routers*). Explotación remota a través de puertos TCP (por ejemplo, puerto 80).

- Versiones vulnerables del sistema gestor de bases de datos MSSQL. Explotación remota a través de puertos TCP (por ejemplo, puerto 1433).

Se consideran también parte de esta etapa la explotación de deficiencias de autenticación de los servicios encontrados durante la etapa inicial. Por ejemplo, en el caso de que los equipos encontrados tengan servicios abiertos basados en una autenticación por combinación de nombre de usuario y contraseña (tipo

Netbios, Miscrosoft-DS, FTP, VNC, etc.), un ataque manual o automatizado en forma de gusano puede tratar de acceder a los recursos del servicio mediante *password cracking*, *rainbow tables*, ataques de diccionario o simplemente fuerza bruta. En la mayoría de los casos, aun desconociendo los nombres de usuario asociados al equipo que atacar, es normal tratar de forzar el uso de cuentas por defecto, y combinaciones conocidas, tales como:

- guest/letmein, invited/client, recovery/temp,
- staff/changeme, faculty/qwerty, student/1234,
- tech/public, ftp/default, manager/friend,...

Estas combinaciones, aunque puede que ofrezcan únicamente accesos remotos con privilegios mínimos (en general, solo lectura y con perfil de invitado), pueden más adelante dar lugar a la instalación local de *exploits* en el sistema operativo, que permitan una escalada de privilegios final, para obtener control total sobre el equipo atacado.

Por último, es en esta etapa cuando se realizan, también, algunas de las partes más críticas en el despliegue de una botnet: la instalación de troyanos y puertas traseras, inicialización de servicios y futuros canales de comunicación/gestión, corrección de vulnerabilidades (para evitar a otros atacantes u operadores tomar control sobre los equipos), etc. Por ello, multitud de autores consideran que el ciclo de vida de una botnet empieza precisamente durante esta parte de la etapa de explotación, acceso e infección. La siguiente lista (no exhaustiva) muestra algunos ejemplos de puertas traseras y troyanos que son (o han sido) con frecuencia utilizados por botnets durante la etapa de infección de equipos:

Ved también

Para más información sobre troyanos, rootkits y puertas traseras podéis ver el apartado 4 del módulo didáctico "Vulnerabilidades de bajo nivel y software malicioso".

- *Subseven* (puerto por defecto, 27347)
- *NetDevil* (puerto por defecto 903)
- *Optix* (puerto por defecto 3140)
- *Bagle* (puerto por defecto 2745)
- *Kuang* (puerto por defecto 17300)
- *Mydoom* (puerto por defecto 3127)

Las herramientas anteriores suelen ser instaladas de manera automática; pero, configuradas de manera manual por parte de los operadores (humanos) de la botnet. Esto es, generalmente, necesario para garantizar la configuración apropiada de los equipos infectados. También, para evitar errores que podrían ser utilizados por los propietarios genuinos de los equipos para descubrir y eliminar (por ejemplo, por medio de herramientas de detección de código viral) parcial o globalmente las herramientas instaladas. Por último, puede ser también necesaria la interacción directa de los operadores con el objetivo de inicializar contraseñas y listas de control de acceso para evitar que terceras partes se apropien de los equipos recién infectados. Veremos con más detalle

parte de estas acciones en los apartados siguientes, en relación con las técnicas para la gestión, coordinación y protección de recursos de una botnet.

2.3. Ataques e infecciones complementarias

Concluimos este apartado con un resumen de técnicas complementarias, que suelen realizarse en paralelo a las etapas anteriores. Se trata de estrategias menos convencionales y menos estudiadas desde un punto de vista académico, pero igual de efectivas a la hora de encontrar e infectar víctimas. La mayoría de ellas propician, además, una forma asíncrona e implícita para que las víctimas se descubran y expongan a sí mismas ante los ataques preparados por el operador de la botnet. La mayoría son simples vectores de transmisión utilizados por *malware* tradicional (tipo gusanos, troyanos y virus). Puesto que la mayoría corresponden a técnicas relacionadas con la ingeniería social, veremos un simple resumen de las tres técnicas más comunes y representativas:

Ved también

Para más información sobre las técnicas de ingeniería social podéis ver el módulo didáctico "Ingeniería Social".

1) Distribución de correos electrónicos que contienen el *malware*. Como para la distribución de muchos otros tipos de *malware* en general, los operadores de una botnet suelen utilizar equipos ya infectados, junto a técnicas de ingeniería social, para la distribución de código malicioso en correos electrónicos (tanto a través de servicios SMTP, POP3 o IMAP; como por medio del uso de nuevos servicios de mensajería para redes sociales). Un código malicioso será asociado a los mensajes con el objetivo de explotar vulnerabilidades de sistema o de red en las aplicaciones cliente de la víctima del ataque. Para ello, el código será asociado al mensaje como un documento o fichero adjunto (o simplemente referenciado en forma de enlace en el cuerpo del mensaje). A continuación, el uso de técnicas de ingeniería social garantizará que la víctima ejecute el código (o visite el enlace) desde la aplicación o el servicio vulnerable que el atacante espera explotar.

Ejemplos

Los siguientes son algunos ejemplos de ataques relacionados con esta categoría:

- envío de enlaces tipo XSS, CSRF, phishing, tratando de engañar y robar información proporcionada por el usuario (por ejemplo, nombres de usuario y contraseñas);
- código oculto dentro del mensaje electrónico para la posterior instalación de macros o código interpretado (imágenes o estructuras de datos con contenido malicioso, tipo vbasic, javascript, flash, etc.) que tratará, más adelante, de explotar vulnerabilidades locales o remotas en la víctima, y
- propaganda sobre lanzaderas web (por ejemplo, falsos noticiarios en línea o sitios web con contenidos ilícitos, que esconden en su código HTML la inyección y ejecución en la víctima de código viral).

Ved también

Para más información sobre *phising* y sus variantes podéis ver el apartado 4 del módulo didáctico "Ingeniería social".

Sea cual sea la técnica, el objetivo será siempre el mismo: bien de manera explícita por parte del usuario, bien de manera implícita debido a errores de programación, que se ejecute en el sistema operativo de la víctima algún proceso

que acabará explotando una vulnerabilidad local o remota para autoinstalar el código (o parte del código) asociado con los binarios de la botnet.

2) Infección a través de mensajería instantánea. De modo similar al correo electrónico tradicional o de redes sociales, los operadores de botnets utilizan también con frecuencia la explotación y el despliegue de *malware* por cuentas de mensajería instantánea. De nuevo, nos encontramos con la combinación de técnicas de ingeniería social con el objetivo de engañar y forzar a los usuarios de equipos vulnerables (bien en el nivel de aplicación o de red) a pulsar sobre ficheros adjuntos o referencias/lanzaderas malintencionadas. Es común, además, la inundación de múltiples cuentas en paralelo mediante SPIM (del inglés, *spam in instant messaging SPIM*).

3) *Malware* **camuflado en el intercambio de ficheros vía P2P**. Por último, viene siendo común la explotación (una vez más, a través de técnicas de ingeniería social) de la confianza de los usuarios de herramientas para el intercambio de ficheros vía redes par a par (del inglés, *peer-to-peer* o P2P), tipo eMule, bittorrent, etc. El código asociado a la botnet será, por norma general, camuflado entre contenidos populares de estas redes con el objetivo de ser transmitido y ejecutado en equipos vulnerables.

3. Coordinación y gestión básica de robots

En este apartado tratamos sobre la fase de coordinación y gestión de los robots de la botnet. En dicha fase, los operadores de la botnet deben desplegar los recursos necesarios para realizar, más adelante, la ejecución de los servicios para los que la botnet fue concebida (por ejemplo, venta de información fraudulenta, ejecución coordinada de ataques o creación de otras botnets). Para todo ello, el operador de la botnet desplegará una serie de mecanismos de comunicación, a menudo referenciados en la literatura como canales C&C*. Dichos canales permitirán al operador controlar y coordinar sus robots. Esos mismos canales deben, también, permitir a los robots devolver el resultado de sus operaciones al operador de la botnet.

Del inglés, *command* & *control

Veremos, en primer lugar, esquemas tradicionales, la mayoría construidos sobre arquitecturas centralizadas con el apoyo de protocolos como IRC y HTTP. Estos primeros esquemas se caracterizan por una gran flexibilidad en el despliegue y mantenimiento de canales C&C, pero sufren fuertes limitaciones ligadas a la redundancia de sus servicios y a la dificultad de protección de sus nodos centrales. Dejaremos para el apartado siguiente el uso de esquemas más avanzados que tratan de paliar estas limitaciones mediante el uso de esquemas descentralizados basados en protocolos P2P (del inglés *peer-to-peer*) o esquemas preventivos que tratan de proteger los servidores de canales C&C con el apoyo del protocolo DNS.

3.1. Gestión centralizada basada en servicios IRC

Las primeras botnets de la historia realizan la gestión de sus equipos a través de canales C&C basados en el protocolo IRC. Para ello, el código asociado a la botnet requiere que cada robot (una vez finalizada la explotación de vulnerabilidades y la inyección de código viral) se suscriba de manera automática a uno, o varios, canales específicos de servidores IRC. Estos servidores estarán, por supuesto, controlados por los operadores de la botnet. Los temas (o *topics*) de dichos canales IRC son utilizados para almacenar e intercambiar comandos con los robots. El intérprete asociado al código del robot simplemente requiere ir analizando e identificando las temáticas de los canales suscritos para ir obteniendo nuevos comandos.

La utilización de la tecnología IRC para la construcción de canales C&C ofrece multitud de ventajas. En primer lugar, ofrece una alto nivel de interacción a partir de un protocolo relativamente sencillo. Posibilita, además, la utilización

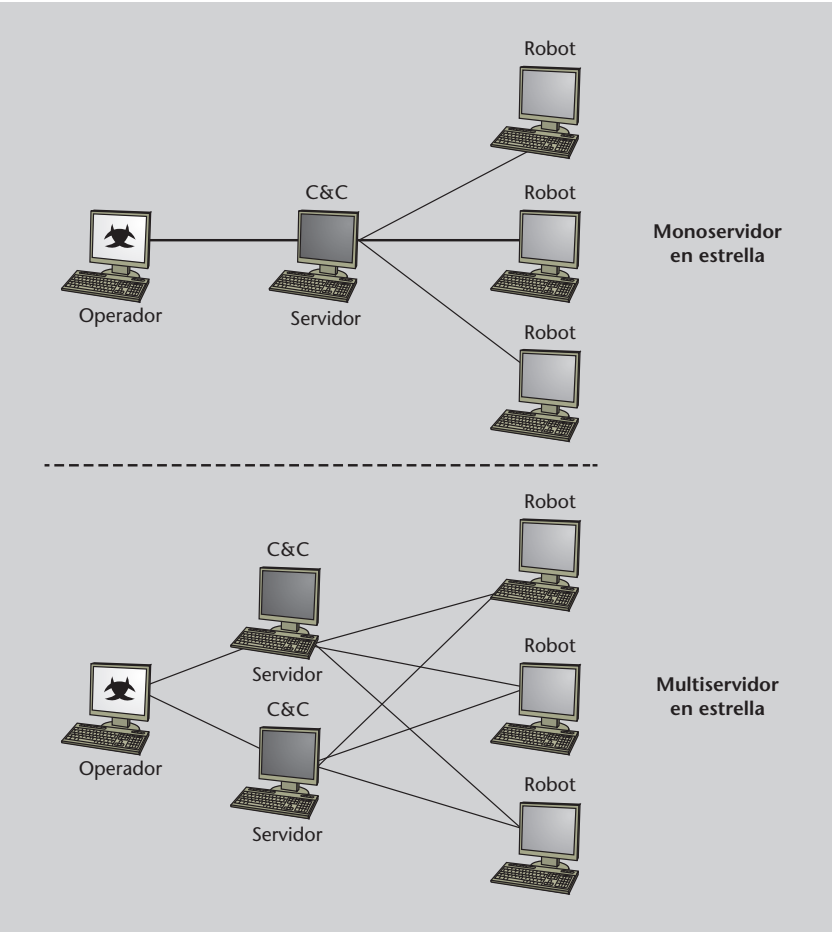
de comunicaciones de tipo *full-duplex* y una alta eficiencia de respuesta en ambas partes de la comunicación (tanto clientes como servidores). El despliegue y la gestión del mantenimiento de los servidores es también relativamente sencilla, ya que existen desde sus inicios multitud de implementaciones para un vasto abanico de lenguajes y plataformas. El protocolo IRC ofrece, además, soluciones para garantizar un mínimo de seguridad en el servicio. De hecho, el protocolo IRC ofrece de forma nativa el concepto de usuario, alias de usuario, gestión de contraseñas, etc.

En cuanto a la redundancia y robustez de los modelos C&C basados en IRC, es importante destacar que se basan en arquitectura centralizadas, caracterizadas por la existencia de uno o varios nodos centrales, encargados de redirigir los mensajes del operador entre los robots. La principal desventaja de esta centralización es la facilidad de detectar y atacar a dichos nodos centrales con el objetivo de desarticular el sistema de comunicación que garantiza la gestión de la botnet. Veremos más adelante algunas de las estrategias de redundancia utilizadas a menudo entre servidores y clientes C&C de botnets actuales con el objetivo de remediar este problema.

Ejemplo

La figura 3 destaca dos topologías de ejemplo que son, o han sido, comúnmente utilizadas en arquitecturas centralizadas para la gestión de robots a partir de tecnologías IRC. Ambas topologías se caracterizan por utilizar servidores IRC que se encargan de centralizar la comunicación de los robots hacia el operador que desplegó la botnet.

Figura 3



Sea cual sea la topología, podemos caracterizar el intercambio de comandos entre robots y operadores mediante el uso de dos tipos de modelos:

- **Mediante modelos tipo PUSH**. Con la utilización de un modelo PUSH, el operador de la botnet será el encargado de enviar (empujar) el primer comando de control que requieren los robots de la botnet. Dicho comando, almacenado comúnmente como el tema de los canales del servidor (o servidores) para los que los robot han sido programados a visitar. Así pues, y de manera asíncrona, los robots de la botnet comprobarán periódicamente (según cómo su agenda haya sido programada) para acceder a dichos canales, para descargar nuevos comandos almacenados por el operador. Cada vez que un nuevo equipo infectado se una a la botnet, recogerá de esta manera la operación inicial que debe ejecutar para completar la fase de despliegue.

- **Mediante modelos tipo PULL**. Con la utilización de un modelo PULL, el mismo comando será ejecutado por todos los robots sin necesidad de interacción directa con el operador. En contrapartida, el operador necesitará conocer a priori dónde almacenar o realizar el envío de los comandos para garantizar una correcta inicialización de los robots del sistema. Para ello, codificará, por ejemplo, un conjunto de direcciones IP o de nombre de dominio asociado a los servidores IRC que los robots deben visitar tras su primera ejecución. Estas listas serán actualizadas periódicamente a cada interacción con los servidores IRC. En el caso de utilizar un modelo PULL, los robots requieren iniciar una comunicación directa con el operador, a la espera de recibir nuevas instrucciones o comandos. Este modelo requiere también que los robots obtengan durante la fase de infección una agenda de tareas, programada por los operadores, y almacenada en los ficheros de configuración de los robots. El robot se encargará de enviar una petición (en inglés *query*) al operador, almacenándola como comentario en algunos de los canales de los servidores IRC desplegados por el operador. Al descubrir la petición, el operador responderá a la consulta mediante alguna aplicación automática con el fin de proveer una respuesta instantánea a los robots.

La comunicación entre robots y operadores, por medio de los servidores de canales C&C basados en tecnologías IRC, se caracteriza también por la dirección de las sesiones que guían el intercambio de los comandos de control, así como por la presencia, o ausencia, de datos devueltos por los robots. Podemos hablar de comunicaciones unidireccionales cuando el operador, siguiendo un modelo de tipo PUSH, empuja los comandos hacia los robots, sin necesidad de que estos últimos proporcionen confirmación o datos asociados a la ejecución del comando solicitado. Este tipo de comunicación está relacionado, en general, con la ejecución de ataques por parte de los robots, para los cuales, el operador no requiere confirmación directa de la ejecución de los comandos solicitados. Por otro lado, el modelo PULL suele ir ligado a la necesidad de comunicaciones bidireccionales que permiten a los robots devolver los resultados asociados a los comandos ejecutados. Este segundo tipo está íntimamente ligado con la necesidad de confirmación por parte de los operadores (por

ejemplo, para informar del estado de las operaciones iniciadas), así como en servicios de recogida de información por parte de los robots (por ejemplo, operaciones relacionadas con campañas de robo de identidades o de intercambio de materiales ilícitos).

3.2. Gestión centralizada basada en conexiones HTTP

El uso de canales C&C basados en HTTP es actualmente (en el 2011) el segundo tipo más común por detrás de IRC. HTTP ofrece a los operadores un uso más intensivo de comunicaciones tipo PULL. Como en el caso IRC, los robots solicitarán información del operador enviando peticiones, y el operador se encargará de recoger, tratar y responder dichas peticiones. En el caso IRC es común la utilización de servidores desplegados directamente por el operador de la botnet. Sin embargo, en el caso HTTP, es común el uso de servidores web desplegados por terceras partes que, de manera inconsciente, se convierten en puntos de almacenamiento de peticiones y respuestas entre robots y operadores. Por ejemplo, sitios web que permitan a los usuarios enviar y consultar información (tipo foros, blogs, redes sociales y gestores web de correo electrónico) pueden ser utilizados para el intercambio de información. Robots y operadores deberán simplemente definir un código y lugar donde depositar tanto peticiones como respuestas. A continuación, los robots almacenarán sus peticiones codificadas apropiadamente, por ejemplo, en algún servidor web accesible públicamente por cualquier usuario.

Es posible también esquemas indirectos de comunicación entre operadores y robots mediante conexiones HTTP unidireccionales. En este caso, los robots realizarán una conexión, sin necesidad de obtener respuesta inmediata. Más adelante, el operador de la botnet tratará de recoger dichas conexiones (por ejemplo, a través de *logs* reportados por los servidores de dichos servicios) y descubrir, por ejemplo, las direcciones IP asociadas a cada uno de los robots. El operador no siempre requerirá un control directo sobre los servidores web accedidos por los robots. La posibilidad de encontrar dicha información publicada inconscientemente por los administradores de los servicios accedidos (por ejemplo, históricos de visitas o estadísticas similares accesibles desde el exterior) será suficiente. El operador se encargará, más adelante, de proporcionar los comandos a los robots que se hayan dado conocer por esta vía.

Los canales C&C implementados mediante HTTP corresponden, en general, a modelos de comunicación tipo PULL, a través de los cuales los **robots** descubren peticiones y actúan en consecuencia. Los robots serán programados para consultar periódicamente los servicios web proporcionados por los operadores con la finalidad de obtener las órdenes. Sin embargo, es posible también utilizar HTTP para la construcción de canales C&C tipo PUSH, por medio de los cuales el operador (que controlará en todo momento los servidores del servicio) se encargará de suministrar los comandos a los robots.

Esquemas más avanzados contemplan la posibilidad de intercambiar información codificada en forma de URL que contendrán, por ejemplo, credenciales necesarias para que el operador se conecte a las puertas traseras instaladas por los robots. Nuevamente, el intercambio de información se realizará de manera indirecta a través de conexiones HTTP en servidores web controlados o accesibles por el operador. Dichas URL codificarán la información de la manera siguiente:

```
http://mybotnet.canal.to/get?puerto=2001&clave=1234
```

La URL anterior deberá ser interpretada por el operador para extraer la información relevante (puerto y contraseña, por ejemplo). Posteriormente, será utilizada para conectarse a los robots y proporcionarles las instrucciones o respuestas correspondientes.

3.3. Gestión centralizada basada en protocolos de aplicación similares

Aunque IRC y HTTP son los dos protocolos más comunes a la hora de implementar arquitecturas centralizadas para controlar y gestionar los robots, es posible encontrar también el uso de otros protocolos de aplicación. La mayoría, en su versión PULL, ofrecen modos interactivos e indirectos para el intercambio de peticiones y respuestas entre robots y operadores, por medio de ficheros almacenados en servidores FTP secuestrados por los operadores, o públicos en general, así como secuestros de identidades de servicios públicos de mensajería instantánea. Sin embargo, a fecha de hoy, son mínimas las botnets que han sido descubiertas haciendo uso de tecnologías como FTP o mensajería instantánea para la comunicación a través de arquitecturas centralizadas. Posiblemente, este hecho radica en la necesidad de crear y controlar cuentas asociadas a estos servicios.

4. Mayor redundancia y protección en las comunicaciones

A medida que las botnets han ido avanzando, en especial hacia redes para la automatización de tareas ilícitas, el rendimiento y la protección de recursos en el despliegue de canales C&C empieza a tomar mayor importancia. El uso de esquemas centralizados incrementa las amenazas de secuestro o desarticulación de una botnet por terceras partes (en general, por los operadores legítimos de los recursos de la botnet, aunque también son posibles ataques provenientes de otras comunidades de atacantes). Aunque los protocolos usados en arquitecturas centralizadas permiten un mínimo de seguridad (por ejemplo, mediante listas de control de acceso por perfiles de usuario y protección de canales mediante contraseñas), no es posible garantizar la resistencia contra localización y desarticulación de servidores C&C.

Mostraremos en este apartado dos alternativas, cuyo objetivo es mejorar la redundancia de los recursos ofrecidos por los robots, así como la seguridad de sus servidores C&C. Esto último, para evitar la localización final de los operadores y la desarticulación total de la botnet. Veremos, en primer lugar, el uso de esquemas totalmente descentralizados, la mayor parte inspirados en el uso de estructuras par a par (P2P, del inglés *peer-to-peer*) ya existentes para el intercambio de ficheros a escala global; y, en segundo lugar, el uso de esquemas jerárquicos que combinen estrategias híbridas entre centralización y descentralización total. Este último caso tratará de explotar al máximo las posibilidades que ofrece el protocolo DNS para renovar periódicamente el enlace entre referencias de dominio lógicas y físicas de servidores intermedios.

Las primeras botnets utilizaban robots software para la automatización de tareas legítimas. Los canales de control de estas primeras botnets, basados en general tanto en IRC como en HTTP, tenían como objetivo mejoras en la funcionalidad de sus redes. Sin embargo, la seguridad de los servicios ofrecidos, o la facilidad de desarticulación de dichos servicios, no es ni mucho menos uno de los objetivos principales de tales diseños. Las botnets actuales, en general desplegadas para la ejecución de tareas ilícitas, presentan una evolución continua respecto a sus canales de comunicación y gestión, en busca de nuevas alternativas que garanticen una gestión más redundante y segura.

4.1. Necesidad de estrategias alternativas

Antes de pasar a presentar las alternativas que parecen estar abriéndose camino en cuanto a la construcción de nuevos canales C&C para botnets, re-

pasaremos una vez más las deficiencias que esquemas anteriores, mayoritariamente basados en tecnologías como IRC y HTTP, pueden ser utilizadas para comprometer la seguridad de una botnet. En primer lugar, está el problema de centralización de las comunicaciones entre operadores, servidores de canales C&C y los robots. Por definición, la botnet trata de distribuir un conjunto de operaciones entre un gran ejército de equipos remotos. Al mismo tiempo, se pretende que el control resida, cuanto más mejor, en manos de unos pocos usuarios (puesto que las actividades son, en general, ilícitas). Una centralización expone la seguridad del operador, u operadores, ya que puede ser reportada a las autoridades pertinentes para que procedan a la denuncia y desarticulación de los recursos asociados. La mayoría de las botnets conocidas hoy en día han sido ya desarticuladas. Para ello, es imprescindible localizar el centro de operaciones de la botnet y reportarlo a las autoridades y proveedores de servicio pertinentes, quienes buscarán y bloquearán el tráfico generado tanto por los servidores C&C como por los robots.

Actualmente, los proveedores de servicio de Internet dedican grandes presupuestos para rastrear y detectar operaciones sospechosas en sus redes. Una vez obtengan las pruebas necesarias, serán ayudados por las autoridades de seguridad competentes para finalmente tratar de interrumpir las operaciones de una botnet, realizar las denuncias oportunas y encarcelar a los responsables que promovieron o desplegaron la botnet. En la mayoría de los países, las actividades asociadas a una botnet se encuentran ya tipificadas apropiadamente para poder realizar las intervenciones policiales y los procedimientos judiciales necesarios para perseguir y detener estas actividades.

Desde una perspectiva puramente técnica, el uso de protocolos como IRC y HTTP, sin protección criptográfica de origen, facilitará también el trabajo a las autoridades que tratan de desarticular una botnet. Aquellos servidores de canales C&C que no dispongan de protecciones criptográficas serán susceptibles de escuchas digitales, por medio de *sniffers de red*, lo cual permitiría analizar y rastrear los comandos proporcionados a través de, por ejemplo, un simple canal IRC compartido entre operadores y robots para el lanzamiento de tareas, y la entrega de resultados. No solo la intercepción de estas informaciones pone en riesgo a los operadores de la botnet, sino que también posibilita que terceras partes puedan emular o tomar el control de los robots, bien con fines bienintencionados, por parte de las autoridades que tratarán de desarticular la botnet, o bien por otras comunidades de atacantes, con el fin de ampliar el número de robots de sus botnets.

Lectura recomendada

Brett Stone-Gross; Marco Cova; Bob Gilbert; Richard Kemmerer; Christopher Kruegel; Giovanni Vigna. "Analysis of a Botnet Takeover". *IEEE Security & Privacy*, vol. 9, núm. 1, pág. 64–72, enero/febrero 2011.

Todos estos elementos dificultan las tareas de control de los operadores de botnets, que buscan a diario nuevos esquemas que garanticen la supervivencia de sus sistemas, evitando al máximo el riesgo de ser descubiertos y perseguidos por las autoridades (del mismo modo que la desinfección de los robots que están bajo el control de una botnet). En el pasado, servidores de canales C&C basados en IRC permitieron a las autoridades interceptar las actualizaciones

de *malware* solicitados por los robots con el objetivo de restablecer las operaciones normales de los equipos infectados. Hoy en día, los nuevos esquemas buscan también detectar el riesgo de falsos robots, potencialmente controlados por investigadores o fuerzas de seguridad, con el objetivo de introducirse en una botnet para recoger información que más adelante pueda servir para desarticular la infraestructura al completo.

Los esquemas vistos en el apartado anterior son potencialmente vulnerables a estas actividades lícitas (pero peligrosas a ojos de los operadores de una botnet). Todos estos motivos han sido, con certeza, estudiados y analizados por las organizaciones que se esconden detrás del *malware* diseminado por los controladores de las botnets, y explica el nivel de perfección en las comunicaciones C&C de botnets actuales.

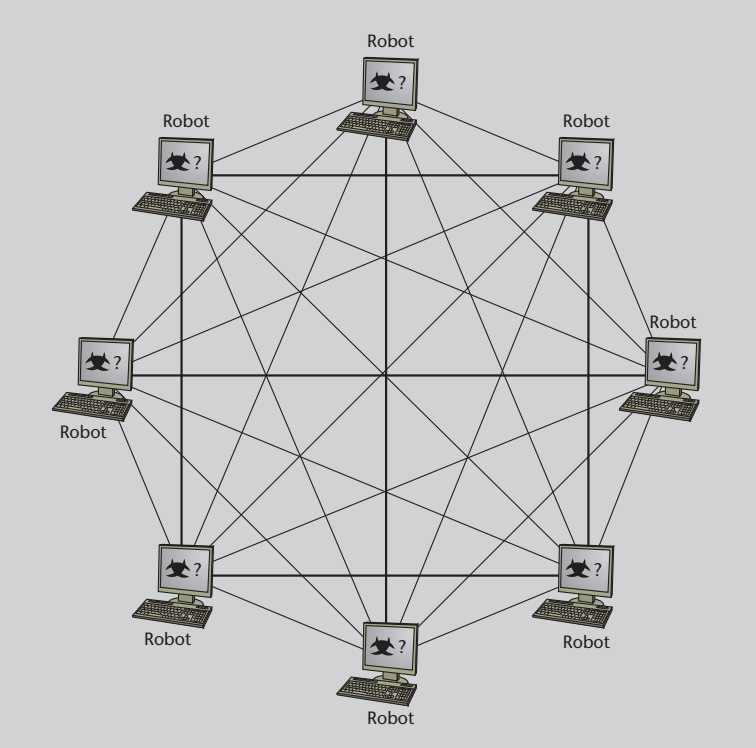
4.2. Comunicación descentralizada mediante redes P2P

Un primer esquema que busca solucionar las limitaciones vistas anteriormente es la descentralización total en las comunicaciones entre robots y operadores. Un ejemplo práctico es la adaptación de redes P2P, utilizadas con éxito para el intercambio de ficheros, y que ha sido ya experimentado en botnets recientes.

Ejemplo

La figura 4 muestra un ejemplo sencillo del aspecto que tendría una arquitectura basada en P2P para la comunicación entre robots y operadores.

Figura 4



La figura muestra una botnet con estructura descentralizada sin servidores C&C propiamente dichos. El operador (u operadores) de la botnet elegirá uno de los robots (equipo comprometido) al azar y lo utilizará temporalmente como servidor C&C. De manera dinámica, irá cambiando y repitiendo este proceso, lo cual le permitirá mejorar la protección de su localización real, siempre pivotado por parte de uno de los equipos que habrá comprometido durante la fase de explotación. Podemos también apreciar en la figura que ninguna de las máquinas tiene, a priori, un papel crítico en la arquitectura. Por lo tanto, no es posible detener o desarticular la red a través de la localización de un conjunto de equipos particulares. Cada equipo, durante la fase de explotación y acceso, recibirá una lista (inicialmente bastante limitada) de direcciones IP de otros robots de la botnet.

La utilización del modelo P2P para la gestión de recursos de una botnet hace desaparecer el concepto de centralismo. Todos los nodos tendrán las mismas responsabilidades de hacer pasar peticiones y respuestas hacia los operadores de la botnet, quienes, además, cambiarán con periodicidad su posición (lógica) en la estructura desplegada de modo aleatorio. Si el esquema es implementado de manera apropiada, las tareas de localización y desarticulación son extremadamente complejas. Por un lado, los operadores aumentarán su anonimato, al poder esconderse tras cualquier robot con igual probabilidad. Por otro lado, ningún equipo de la botnet desempeña, a priori, un papel lo suficientemente importante como para desarticular la red tras su desconexión.

Peacomm

Peacomm es comúnmente presentada en los medios como **Storm**, debido al nombre del gusano que se encargaba de desplegar la infección asociada a la botnet.

Una de las botnets con mayor repercusión mediática, y que estructuraba sus comunicaciones a través de tecnologías P2P, fue Peacomm. La comunicación entre los equipos de la botnet se basó en el protocolo Overnet, adaptado a su vez de Kademlia, un protocolo P2P de gran importancia en aplicaciones P2P para intercambio de ficheros. Además de descentralizar sus comunicaciones, Peacomm las protegía también mediante técnicas de cifrado. Esto permitió proteger la confidencialidad de los comandos y resultados intercambiados entre operadores y robots. Otras redes quizá con menos resonancia, pero que también aplicaron el modelo P2P con anterioridad a Peacomm fueron Sinit y Nigache. De nuevo, sus protocolos para comunicar a los pares se inspiraron en protocolos como BitTorrent y Waste. El resultado final, y en comparación con diseños centralizados vistos anteriormente, fue una mayor protección para los operadores de la botnet o, lo que es lo mismo, mayor dificultad en la labores de localización y desarticulación de los servidores de comunicación C&C utilizados por los operadores para gestionar los robots de la botnet.

Pese a las ventajas inherentes de una estructura P2P para la gestión de botnets, sin complementos adicionales, también presentará deficiencias. El motivo principal es la complejidad y dificultad de desplegar los servicios, lo cual está íntimamente ligado con problemas de latencia y de pérdida de comandos y/o respuestas, ya que no siempre será posible garantizar la entrega de los mensajes entre los equipos que compondrán la botnet.

Una de las primeras botnets con un modelo de comunicaciones descentralizado basado en P2P fue **Sinit** (en torno al año 2003). Sinit utilizaba técnicas de barrido aleatorias para encontrar al resto de los pares de la botnet. Dicho barrido de direcciones provoca un volumen de tráfico fácilmente caracterizable, que permitía detectar y aislar con facilidad equipos infectados por Sinit. Como consecuencia, la red fue desarticulada con relativa facilidad.

La búsqueda de pares también puede entorpecer, e incluso reducir la eficiencia de protección de la tecnología P2P. Una mala implementación del protocolo de inicialización de pares, basado en escaneos aleatorios, puede ser identificado con facilidad por equipos de detección de botnets y código viral, y ser utilizado por los proveedores de acceso a Internet para aislar dichos equipos para desarticular la red. Por lo tanto, la utilización de P2P por si sola, sin mecanismos adicionales de protección, seguirá sin solventar las principales limitaciones ya reportadas en esquemas centralizados.

4.3. Protección basada en renovación cíclica de referencias DNS

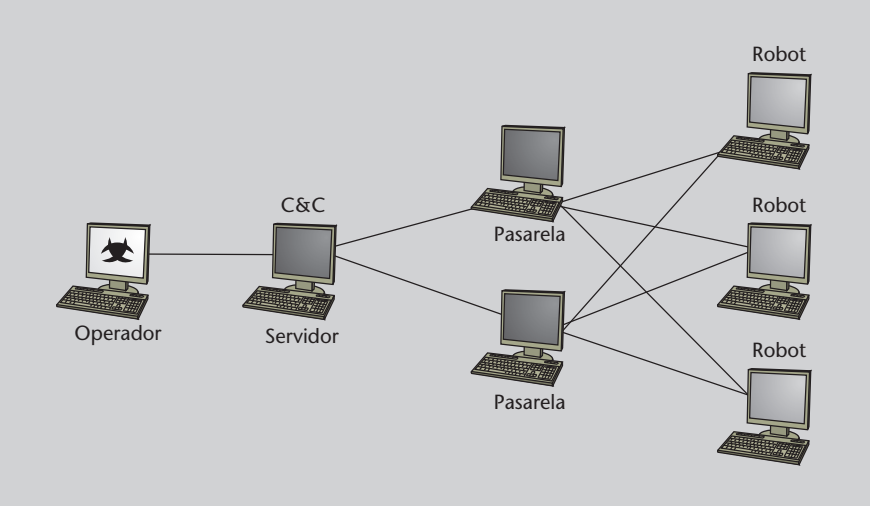
Como vimos al inicio de este apartado, una centralización de las comunicaciones facilita el despliegue de la botnet, pero puede también facilitar la localización de los servidores C&C y agilizar la desarticulación de la botnet. Por otro lado, una descentralización total del sistema de comunicaciones puede ayudar inicialmente a proteger los recursos de la botnet, pero complica el despliegue y la eficiencia de las comunicaciones. Además, si no es utilizada correctamente, puede acabar siendo utilizada para caracterizar el comportamiento de los robots de la botnet y, nuevamente, facilitar su desarticulación. Operadores de botnets han estado trabajando estos últimos años en estructuras que combinan lo mejor de ambas opciones con el objetivo de encontrar un compromiso híbrido que flexibilice tanto el despliegue de robots como la protección de los servidores C&C. El uso de topologías jerárquicas basadas en federación de servidores C&C a través de pasarelas de aplicación (en inglés, *proxies*) parece ser la clave para encontrar dicho compromiso.

Un ejemplo de botnet con un sistema de comunicaciones basado en topología jerárquica es **Waledac** (en torno al año 2008). Los robots de Waledac comunican directamente con un conjunto de nodos repetidores que se comportan como pasarelas. Dichos repetidores reenviarán las peticiones y respuestas de los robots a servidores de nivel superior mediante comunicaciones cifradas con el protocolo TSL (*transport security layer*). Los nodos raíz de la jerarquía actuarán finalmente como servidores C&C para responder a las peticiones de los robots, o recoger sus resultados.

Ejemplo

La figura 5 muestra un ejemplo sencillo de topología jerárquica. En la figura podemos ver una estructura multiservidor que utiliza comunicaciones estratificadas en diferentes niveles y enlazados con los robots a través de pasarelas. Los robots comunicarán en primera instancia con las pasarelas, que redirigirán los mensajes hacia los niveles superiores, donde se encontrarán protegidos los nodos que actúan como servidores C&C (en última instancia, en comunicación directa con el operador de la botnet). Generalmente, las comunicaciones entre pasarelas y servidores de nivel superior utilizarán técnicas criptográficas para proteger sus comunicaciones.

Figura 5



Recordemos que todos los esquemas vistos hasta el momento pueden beneficiarse del uso del protocolo DNS para obtener comunicaciones de tipo *multihoming*. El concepto de *multihoming* es utilizado por servicios tradicionales (y legítimos) para asignar múltiples direcciones IP a un mismo nombre asociado con DNS. Por ejemplo, la configuración de un servidor DNS puede asignar una configuración como la siguiente:

```
acmeBotnet.servers.com pointing to 10.0.0.1
acmeBotnet.servers.com pointing to 10.0.0.2
acmeBotnet.servers.com pointing to 10.0.0.3
...
```

Al mismo tiempo, el uso de DNS también facilita que una serie de equipos principales gestionen dichos localizadores de referencias, incluso antes de que la conexión con el servidor C&C que les pondrá en contacto se haga efectiva. El hecho de descubrir e informar sobre dominios inexistentes o comprometidos por los operadores de una botnet supone un riesgo mucho menor (para los operadores) que, por ejemplo, descubrir e informar las direcciones IP finales de los servidores C&C. Esta característica presenta grandes ventajas y parece ser la estrategia común en botnets actuales para conseguir la máxima protección de los equipos más cercanos a los operadores de la botnet, tanto los servidores finales como los equipos intermediarios entre el operador y los robots. Para beneficiarse de dicha protección, los robots dirigirán sus peticiones hacia un conjunto de servidores DNS establecidos a priori por el operador de la botnet, y recibirán como respuesta el conjunto de direcciones IP asociadas a los servidores C&C. Combinado con el uso de técnicas como FastFlux, el sistema final

mejorará la redundancia y resistencia frente a una posible desarticulación por parte de los proveedores de acceso a Internet donde se encuentran localizados los robots. De este modo, cada robot recibirá un conjunto de referencias que resolver, cuyas direcciones IP se irán alternando a lo largo del tiempo.

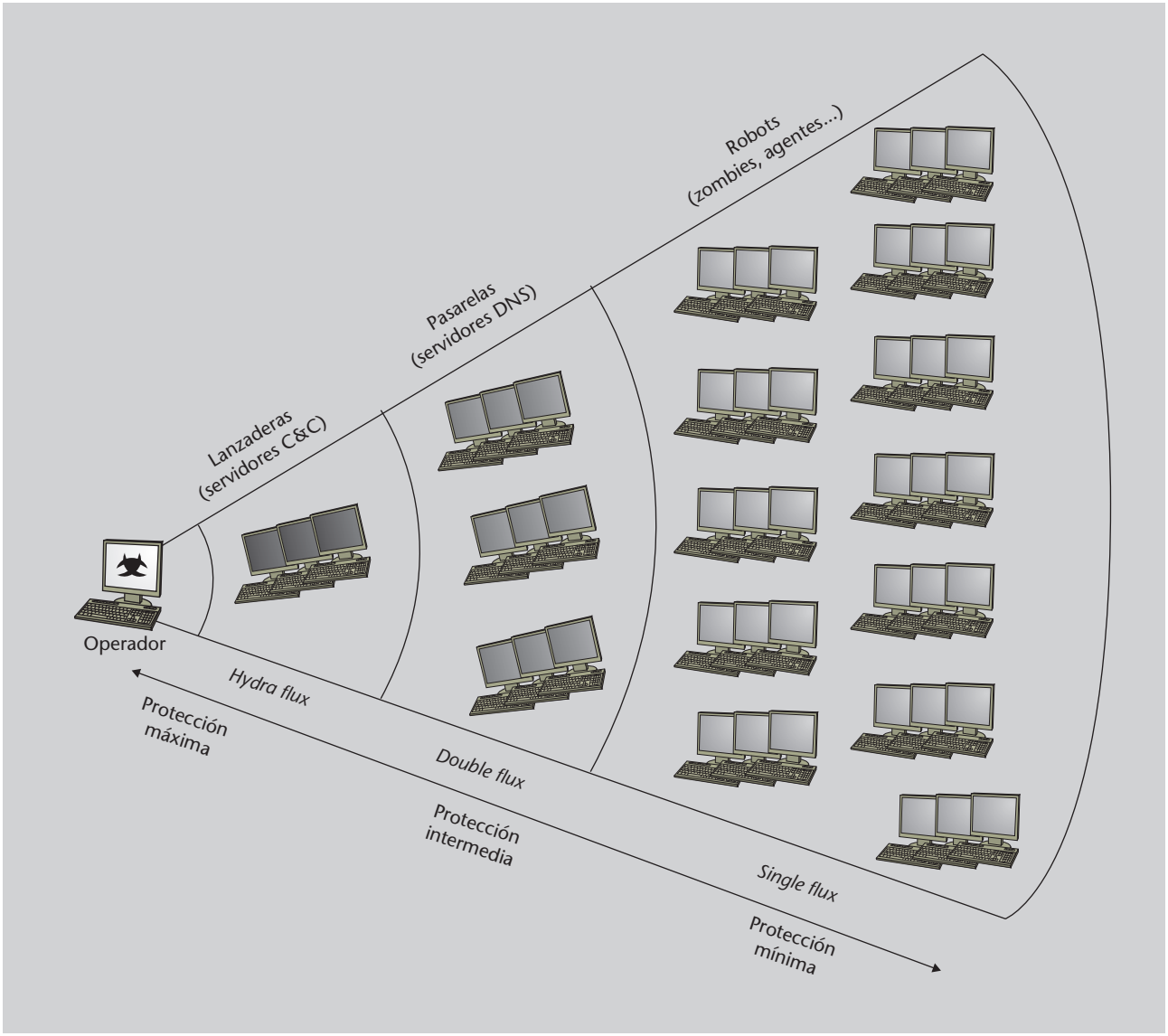
Variantes y mejoras del FastFlux tradicional pueden ayudar a complicar aún más el rastreo y la desarticulación de las infraestructuras C&C de una botnet. El uso de FastFlux avanzado, sobrenombre con el que se refiere a menudo en la literatura a la composición jerárquica de FastFlux encadenado, recoge la mayor parte de estas variantes. La estrategia consiste en colocar los recursos más cercanos al operador (tales como la consola de operaciones y los servidores C&C) en la parte más alta de la jerarquía, renombrados ahora con la etiqueta de lanzaderas (del inglés, *motherships*). A continuación, se colocarán en la segunda parte de la jerarquía los servidores de DNS, etiquetados en general como pasarelas, y que realizarán el enlace final entre robots y lanzaderas.

La figura 6 muestra los distintos niveles de protección que pueden obtenerse con el uso de FastFlux avanzado.

Los servidores C&C, ahora con mayores responsabilidades que en esquemas anteriores, continúan en contacto directo con el operador de la botnet. Entre sus nuevas responsabilidades, los servidores C&C deberán gestionar también la infraestructura de servidores DNS y hacer efectiva la protección que FastFlux avanzado ofrece a la infraestructura global de la botnet. La gestión de servidores DNS corresponde a tareas de registro, actualización de nombres dominios y mantenimiento de los servidores de nombre de cada dominio. El registro y la actualización de nombres de dominio es realizado, en general, en forma de registros DNS de tipo NS (*name server*) que almacenarán el enlace del servidor (o servidores) encargados de responder a las peticiones DNS de los robots por cada tipo de dominio establecido por los operadores de la botnet. Es necesario gestionar, también, los ficheros de configuración de las distintas zonas existentes en cada dominio establecido por el operador. Dichos ficheros contienen la lista de los equipos existentes en un dominio, así como el enlace correspondiente entre nombre y dirección IP (en forma de registros de dirección).

La botnet **Conficker** (en torno al año 2008) basó su estrategia de protección en la combinación de FastFlux y la introducción de algoritmos propios para la generación aleatoria de nombres de dominio asociados a los servidores C&C. De manera simplificada, los robots de Conficker requerían generar una lista de posibles dominios DNS asociados a sus servidores C&C. Más adelante, el operador se encargaría de registrar dicha lista de dominios para asociarlos a la localización final de los servidores C&C. Periódicamente, la localización (dirección IP) de los servidores C&C se adaptaba para reducir el riesgo de una desarticulación de la botnet.

Figura 6



Los ficheros de configuración de cada zona contendrán también el valor TTL (tiempo de vida, del inglés, *time-to-live*) asociado con cada registro de dirección. Dicho valor establece el tiempo (en general, en segundos) durante el cual la asociación es almacenada por el cliente en su memoria caché. Transcurridos los TTL segundos, el cliente dará por caducada dicha asociación y volverá a solicitar a su servidor de DNS que interrogue nuevamente a los servidores raíz la nueva dirección IP. Mediante la utilización de valores TTL pequeños, los operadores asegurarán que todos los equipos de la botnet refresquen sus enlaces de manera periódica. De nuevo, el objetivo final es encontrar el mejor compromiso para que ningún punto de la red tenga dependencias directas con nodos principales. En consecuencia, el descubrimiento de cualquier nodo de la red elegido al azar supondrá el mismo efecto para la botnet, aumentando la robustez de la botnet contra una posible desarticulación.

Lectura recomendada

Es interesante la lectura del artículo "Know Your Enemy: Fast-Flux Service Networks, An Ever Changing Enemy", de los autores del proyecto Honeynet, donde se puede ver la utilización de FastFlux básico y avanzado. El artículo está disponible en la siguiente dirección: http://honeynet.org/papers/ff/

5. Modelo económico asociado a las botnets

De modo muy breve, concluiremos este módulo mostrando algunos detalles sobre el modelo económico que explica la existencia actual de los despliegues de botnets. Este modelo es también clave para entender la progresiva evolución de estas redes hacia nuevas técnicas que dificulten las investigaciones que tanto proveedores de servicios de Internet, como autoridades y fuerzas policiales de distintos países realizan a diario en su guerra particular contra las organizaciones que promueven la existencia de las botnets (y a sus potenciales clientes).

5.1. Primeras generaciones

Como ya hemos visto en los primeros apartados, las botnets iniciales se construyeron sobre esquemas muy básicos, generalmente basados en un código ejecutado por parte de robots que contenían multitud de errores de programación (es decir, repletos de *bugs* software) y que presentaban facilidad de detección y desarticulación de los servidores C&C. También, hemos visto que el uso de arquitecturas centralizadas, común en las primeras botnets, facilitaba el trabajo a investigaciones policiales con el objetivo de encontrar el origen de los operadores y desarticular las botnets.

Los años 2002 y 2003 fueron los años de mayor auge de las botnets. En especial, con el despliegue de Gaobot y sucesores. Aunque su descubrimiento y desarticulación fue relativamente rápido, sientan las bases respecto a nuevos métodos de reproducción, búsqueda y explotación de vulnerabilidades de equipos poco protegidos (pero localizados en redes con grandes capacidades). Estas técnicas permitirán a sus operadores la expansión de actividades típicas de economías sumergidas del mundo real sobre el plano digital (a través de Internet). Estas primeras generaciones no se caracterizan por una alta calidad de productos, sino por la fuerza potencial de sus recursos. Efectivamente, la posibilidad de gestionar a distancia redes de más de 65.000 robots permiten recorrer con facilidad gran parte de los equipos conectados a Internet, generando por cada robot un volumen relativamente bajo de tráfico.

Las tendencias actuales se caracterizan por unos operadores de botnets cada vez mejor formados, con amplios conocimientos en construcción de protocolos robustos y utilización de comunicaciones cifradas. De hecho, las botnets esconden actualmente ingenieros con gran experiencia tanto en *networking* como en seguridad. Estudios recientes muestran mejoras constantes de canales C&C e incorporación de técnicas de anonimato para continuar dificultando la detección y desarticulación de robots y servidores asociados.

A medida que las técnicas de detección de robots o servidores C&C avanzan, los operadores de botnets tratan también de adoptar nuevas medidas que dificulten la detección de firmas o patrones que lideren la desarticulación de sus equipos.

Pero pasemos a repasar a continuación, antes de concluir este módulo, algunas de las actividades principales que se esconden tras una botnet, así como las previsiones de futuro que explican la mejora constante de las botnets.

5.2. Actividades asociadas a botnets actuales

Hemos destacado en el primer apartado de este módulo que una de las primeras actividades que dieron a conocer las botnets fue precisamente la ejecución de ataques. Más concretamente, ataques de tipo DDoS. Se trata de ataques a la disponibilidad de servicios ofrecidos por equipos o redes de terceros, que desempeñarán el papel de víctimas de la botnet. El total, o un gran número, de los robots de la botnet tratarán de consumir los recursos de las víctimas de manera simultánea, anonimizando, además, el origen real del ataque. En efecto, los comandos y las peticiones originales del operador de la botnet pasarán desapercibidos a ojos de la víctima y de las investigaciones posteriores al ataque.

También es conocido por todos el uso de las botnets para la puesta en práctica de campañas de *spamming* para la diseminación de anuncios deshonestos. Como la mayoría de las amenazas a Internet, las campañas de *spamming* suelen lanzarse desde botnets controladas por operadores anónimos. Nuevamente, los comandos y las peticiones originales por el operador pasarán desapercibidas por las víctimas de estos ataques de venta ilícita de información, garantizando una diseminación anónima de productos en línea, en busca de compradores potenciales. Como en el caso anterior, los robots actúan como repetidores o pasarelas de los mensajes originales orquestados por los operadores de la botnet.

Quizá, menos conocido por el público en general, es el uso de las botnets como lanzadera de diseminación de código *malware* para botnets ya existentes, o para nuevas botnets que se están aún desplegando. De manera paralela a la diseminación de mensajes considerados por las víctimas como *spam*, los robots son utilizados con frecuencia para diseminar código *malware*. El objetivo es contaminar y continuar el despliegue de la misma botnet, o de terceras partes, garantizando la existencia de bases de código *malware* distribuido a lo largo de Internet. Multitud de ataques relacionados con vulnerabilidades de servicios web, tales como XSS, CSFR, *phishing*, etc., dependerán en gran medida de la existencia de botnets paralelas encargadas de garantizar el correcto despliegue del código final de los ataques correspondientes.

Ved también

Para más información sobre vulnerabilidades web podéis ver el módulo "Ataques a aplicaciones web".

Otro ejemplo, hasta hace poco desconocido por el público en general, es el uso de botnets para espionaje, tanto de sectores públicos y gubernamentales como dentro de sectores privados (industrias de sectores como el del automóvil, el aeronáutico y el de las nuevas tecnologías). Puesto que los robots de la botnet suelen ser albergados en dichos sectores (pensemos en ordenadores o equipos de sobremesa de trabajadores y ejecutivos asociados), es frecuente el despliegue de ataques sobre vulnerabilidades de red que permitirán la ejecución de escuchas de red para recoger y reenviar a los operadores de la botnet cualquier información que pase por dichos equipos sin la protección adecuada (por ejemplo, sin capacidades de protección criptográfica).

Por último, es importante tener también presente que las botnets son utilizadas actualmente para el almacenamiento y la distribución de contenidos audiovisuales. De hecho, la falta de madurez de un modelo económico real basado en la distribución de contenidos audiovisuales a través de Internet abre a los operadores de botnets un nicho perfecto para almacenar y distribuir contenidos audiovisuales obtenidos de manera ilícita. Se trata, por lo tanto, del almacenamiento y de la distribución de películas, series televisivas, libros electrónicos y música, sin el consentimiento de autores o instituciones que ostentan de manera legal el derecho de copia. Podríamos incluir también en esta categoría la utilización de recursos para albergar servidores de juegos ilícitos, tales como casinos en línea y servicios de apuestas ilegales. El uso de sistemas de ficheros de gran capacidad por parte de los robots de una botnet, así como el acceso a recursos de red con grandes anchos de banda y baja latencia, facilita la distribución de estos elementos y dificulta las investigaciones posteriores sobre el origen real de los equipos que albergan los ficheros.

5.3. Perspectivas y garantías de mejoras continuas

Podemos afirmar que la época en la que aficionados de la informática se dedicaban a programar código malicioso por simple diversión, o para dar a conocer sus habilidades técnicas, ha terminado. Hoy en día, la programación de código para la construcción de botnets es un auténtico negocio. Organizaciones de todo tipo (gubernamentales, comerciales e incluso criminales) se dedican a buscar y contratar a especialistas en la materia para que desarrollen nuevos esquemas y estrategias.

En la actualidad, la motivación principal de los operadores de una botnet suele ser de tipo económica. A diferencia de actividades similares en el mundo real, como por ejemplo el robo o atraco a personas o instituciones físicas, el robo de recursos electrónicos y su utilización con fines deshonestos, además de entrañar muchos menos riesgos físicos y jurídicos, es automatizable y paralelizable. Una vez construida la red, dichos recursos pueden ser alquilados a terceras partes. Estos ingresos económicos explican la evolución y mejora técnica continua de los productos asociados a las botnets actuales.

La mayoría de los estudios actuales sobre las ganancias económicas asociadas con el mantenimiento de una botnet no deja lugar a dudas sobre la viabilidad de su modelo económico. Algunas de las cifras que describimos a continuación, basadas en un estudio realizado en el 2004 por Peter Haag y Alain Hugentobler, ayudan a entender la continua evolución y mejora de las tecnologías asociadas:

- El alquiler de una cuenta de usuario, con acceso no exclusivo a los recursos del robot, asciende a los 15 céntimos de euro mensuales.

- El alquiler de una cuenta de usuario, con acceso exclusivo a los recursos de un robot de la botnet, asciende a los 30 céntimos de euro mensuales.

- El alquiler de una zona parcial de una botnet, con hasta 500 robots, puede alcanzar los 380 euros mensuales.

- La utilización puntual de un volumen mayor de robots para, por ejemplo, la realización de un ataque de tipo DDoS contra una víctima que determinar puede alcanzar entre 40 y 700 euros.

- El alquiler de volúmenes mayores (de un orden superior a los veinte mil robots) para, por ejemplo, realizar una campaña de publicidad mediante el uso de *spam* se comercializa a unos 75 euros por semana.

Informes elaborados por criminalistas y especialistas en delincuencia, tanto en el mundo real como en su versión electrónica, arrojan a la luz cifras similares. La mayoría de estos estudios se basan, además, en resultados y predicciones de más de 5 años de antigüedad, por lo que las cifras actuales pueden ser mucho más elevadas e inquietantes. La mayoría de los especialistas en la materia parecen estar de acuerdo sobre la gravedad de la situación, así como el estado de madurez de los cimientos sobre los que se apoyan hoy en día las botnets.

En relación con las organizaciones que hay detrás de estas redes (tanto hoy en día, como potencialmente las que las usarán en el futuro) se habla a menudo de organizaciones criminales relacionadas con mafias del este de Europa, al igual que carteles relacionados con contrabando de productos desde países africanos y americanos (tanto del norte como del sur). También se habla a menudo de organizaciones gubernamentales en países asiáticos, que podrían valerse de los recursos de las botnets para obtener ventajas industriales en relación con las industrias occidentales. Posiblemente, la realidad esconda a muchos otros actores que financian, directa o indirectamente, mejoras sustanciales para que futuros operadores de botnets puedan pasar desapercibidos y sus recursos difícilmente desarticulados.

Casos recientes de criminales, comerciales, ingenieros y desarrolladores asociados con las botnets están empezando a salir a la luz, dando a conocer al

público general la realidad y potencia de esta gran amenaza. Todos estos casos demuestran una vez más que el objetivo final de las botnets, y el código *malware* asociado a estas redes, no es la destrucción masiva de equipos o recursos informáticos, o el simple acto de personas aisladas con pretensiones deshonestas, sino la obtención de beneficios. La asociación de los primeros servicios, la mayoría relacionados con simples ataques, junto con las tendencias actuales de diseminar publicidad, campañas de venta de productos, e intercambio de transacciones financieras, muestran que no son actividades ingenuas, ni al azar, sino más bien negocios bien organizados y reflexionados.

Resumen

Las botnets constituyen en la actualidad la mayor amenaza conocida contra Internet. Las botnets son el resultado de una infección a gran escala de equipos informáticos que, una vez infectados, pasan a ser controlados por un mismo atacante (o por una misma organización de atacantes), sin que los auténticos propietarios lo descubran y, por lo general, con fines tanto malintencionados como lucrativos. Así pues, los equipos infectados componen la botnet resultante, que puede ser finalmente definida como una red de robots al servicio del atacante. El atacante se convierte en operador de una compleja y potente red cuyos servicios serán finalmente vendidos a organizaciones de todo tipo. La siguiente tabla resume la mayor parte de los aspectos que han sido tratados en este módulo.

Botnets		
Descubrimiento, explotación e infección	Búsqueda de víctimas	Barrido de puertos + escáner de vulnerabilidades Distribución de mensajes corruptos, P2P, IM... Ingeniería social, servicios secretos...
	Explotación de vulnerabilidades	Desbordamientos de pila Condiciones de carrera Robo de contraseñas, fuerza bruta...
	Toma de control de los equipos	Modificación de servicios internos Instalación de código malicioso Obertura de puertas traseras
Coordinación y gestión de los robots	Despliegue de recursos y canales C&C	Arquitecturas centralizadas - Topología monoservidor en estrella - Topología multiservidor en estrella Arquitecturas descentralizadas - Topología aleatoria Arquitecturas híbridas - Topología jerárquica
	Protocolos mayoritariamente utilizados	IRC, HTTP, IM, FTP... P2P (bittorrent, kademlia, Waste...) DNS (resolución y protección de recursos)
	Inicialización y características de la comunicación	Modelo PUSH monodireccional Modelo PULL monodireccional Modelo PULL bidireccional
	Técnicas de redundancia y protección	Descentralización total de los servidores C&C Federación de servidores C&C Uso de FastFlux y FastFlux avanzado
Servicios y actividades asociadas al modelo económico de las botnets actuales	Denegaciones de servicio distribuidas Campañas de venta ilícita Servicios de espionaje Hospedaje de aplicaciones ilícitas Diseminación de aplicaciones ilícitas	Servicios de *spamming* Hospedaje de contenidos ilegales ...

Ejercicios de autoevaluación

1. ¿Cuál de las siguientes afirmaciones es correcta?
a) Las primeras botnets de la historia fueron utilizadas por administradores de operadoras de servicios ilícitos para proteger sus campañas de venta de productos ilegales y para agilizar sus servicios de espionaje y de control de compañías de la competencia.
b) Los ataques de una botnet son controlados por el operador a través de los servidores C&C. Una vez seleccionada una víctima, los robots se limitarán a seguir órdenes y dirigir las acciones hacia el equipo o las redes seleccionadas por el operador.
c) La utilización de protocolos de tipo P2P por parte de los robots de una botnet facilita la identificación de sus servidores C&C, asegurando un despliegue de resursos más sencillo y flexible para los operadores de la red.
d) Las botnets actuales son operadas por *amateurs* con el objetivo de mostrar a amigos y conocidos sus habilidades técnicas a la hora de programar código malicioso.

2. ¿Cómo garantiza el operador que los robots continúen bajo su control?
a) Mediante la modificación de los registros o guiones de iniciazalición de los equipos infectados, asegurando que el código malicioso se ponga en marcha tras cada nueva reinicialización.
b) Con el uso de ingeniería social, haciendo que los usuarios de los equipos infectados continúen ejecutando las aplicaciones relacionadas con la botnet en cada reinicialización.
c) Mediante el uso de incentivos económicos, ofreciendo una participación de la botnet a los usuarios legitimos de los equipos infectados.
d) Por medio del uso de aplicaciones P2P y estructuras aleatorias para la gestión de canales C&C que mantienen los propios usuarios.

3. ¿Qué tipo de equipo informático es infectado y transformado con mayor facilidad en forma de robot de una botnet?
a) Los teléfonos móviles, en especial aquellos que se caractericen por una mayor autonomía y libertad de movimiento.
b) Cualquier equipo informático conectado a Internet que presente vulnerabilidades o deficiencias de seguridad no corregidas.
c) En general, todos aquellos equipos informáticos con sistema operativo de la familia Windows.
d) Cualquier equipo que ofrezca extensiones de programación basadas en código libre.

4. ¿Por qué todos los protocolos para la gestión de robots de una botnet requieren conexiones transportadas sobre TCP?
a) Porque TCP ofrece los mecanimos de redundancia y protección necesarios para evitar una desarticulación total de la botnet.
b) Porque las primeras botnets de la historia así lo hicieron, y las botnets actuales se limitan a extender otras funcionalidades, pero no cambian los protocolos de transporte originales.
c) Porque los operadores de las botnets requieren el uso de sesiones tipo TCP para codificar los comandos de control.
d) No es cierto. También existen casos de botnets cuyos canales C&C se construyen sobre otros protocolos de transporte como, por ejemplo, UDP.

Solucionario

1. b; **2.** a; **3.** b; **4.** d;

Glosario

bug *m* Error de programación que puede desencadenar una deficiencia de seguridad.

caballo de troya *m* Programa, aparentemente inofensivo, que contiene en su interior un ataque contra una vulnerabilidad no corregida.

denegación de servicio *f* Ataque que tratará de saturar recursos de la víctima, tales como memoria o capacidad de cálculo y procesamiento (en inglés, *denial of service*).

DDoS *f* Denegación de servicio distribuida (en inglés, *distributed denial of service*).

DoS *f* Véase **denegación de servicio**.

exploit *m* Técnica (en general, de tipo software) que permite utilizar una vulnerabilidad, aún no corregida, con fines deshonestos.

exploración de puertos *f* Técnica utilizada para identificar los servicios que ofrece un sistema o un equipo en particular.

escáner de vulnerabilidades *m* Aplicación que permite comprobar si un sistema es vulnerable a un conjunto de deficiencias de seguridad.

huella identificativa *f* Información precisa que permite identificar un equipo o una red en concreto (en inglés, *fingerprinting*).

malware *m* Programa con fines malintencionados.

requests for comments *m* Conjunto de documentos técnicos y notas organizativas sobre Internet.

RFC *f* Véase **requests for comments**.

robot *m* Programa deshonesto que permite al operador de una *botnet* controlar a distancia los recursos de un equipo infectado.

sniffer *m* Aplicación que intercepta toda la información que pase por la interfaz de red a la que esté asociada.

troyano *m* Véase **caballo de troya**.

Bibliografía

Filiol, Eric (2009). *Les virus informatiques: théorie, pratique et applications*. (2ª ed.). París: Springer-Verlag France.

Graham, James; y otros (2011). *Cyber Security Essentials*. Boca Ratón: Taylor & Francis Group.

Paget, François (2005). *Vers & Virus. Classification, lutte anti-virale et perspectives*. París: DUNOD.

Schiller, Craig A.; y otros (2007). *Botnets: the killer web app*. Waltham: Syngress Media Inc.