CloudBotNet

MEMORIA TRABAJO FIN DE GRADO

Abdalah Fallaha Sabhan
Jose Luís Góngora Fernández

Directores:

José Luis Vazquez-Poletti
José Manuel Velasco Cabo

Grado en Ingeniería Informática

Facultad de Informática

Universidad Complutense de Madrid

Junio 2016

RESUMEN

La Ciberseguridad es un campo que cada día está más presente en nuestra vida con el avance de la tecnología. Gobiernos, militares, corporaciones, instituciones financieras, hospitales y otros negocios recogen, procesan y almacenan una gran cantidad de información confidencial en sus ordenadores y transmiten estos datos a través de redes a otros ordenadores. Con el creciente volumen y la sofisticación de los Ciberataques, se requiere una atención continua para proteger los negocios sensibles y la información personal así como salvaguardar la seguridad nacional. En el futuro casi todo va a ser informático por lo que con el avance de la tecnología nuevas amenazas aparecen, más peligrosas y sofisticadas. El enfoque de nuestro proyecto es demostrar que con unos pocos conocimientos de redes, seguridad, computación en la nube y unas pocas líneas de código se puede implementar una potente herramienta de ataque que puede poner en peligro la integridad y confidencialidad de los usuarios e instituciones.

Palabras Claves: *Botnet , Fabric , Python , Cloud Computing, SecureShell(SSH), Sysadmin*

ABSTRACT

Cybersecurity is a field that everytime is more present in our lives with the advance of the technology. Governments, military, corporations, financial institutions, hospitals and other businesses collect, process and store a great deal of confidential information on computers and transmit that data across networks to other computers. With the growing volume and sophistication of cyber attacks, ongoing attention is required to protect sensitive business and personal information, as well as safeguard national security. In the future almost everthing is going to be computer-based so with the advance of the technology new threats will appear, more dangerous and sophisticated. The approach of this project is to demonstrate that with a few knowledge of network security, cloud computing and some coding lines it is possible to implement a powerful attack tool that could represent a serious danger for the integrity and confidentiality of users and institutions.

Palabras Claves: *Botnet , Fabric , Python , Cloud Computing, SecureShell(SSH), Sysadmin*

AUTORIZACIÓN PARA LA DIFUSIÓN DEL TRABAJO FIN DE GRADO Y SU DEPÓSITO EN EL REPOSITORIO INSTITUCIONAL E-PRINTS COMPLUTENSE

Los abajo firmantes, alumno/s y tutor/es del Trabajo Fin de Grado (TFG) en el Grado en Ingeniería Informática de la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el Trabajo Fin de Grado (TF) cuyos datos se detallan a continuación. Así mismo autorizan a la Universidad Complutense de Madrid a que sea depositado en acceso abierto en el repositorio institucional con el objeto de incrementar la difusión, uso e impacto del TFG en Internet y garantizar su preservación y acceso a largo plazo.

TÍTULO del TFG: CLOUDBOTNET
Curso académico: 2015 / 2016

Nombre del Alumno/s:

Abdalah Fallaha Sabhan
Jose Luís Góngora Fernández

Tutor/es del TFG y departamento al que pertenece:

José Luis Vazquez-Poletti
José Manuel Velasco Cabo

INDICE

1. INTRODUCCIÓN

1.1 ANTECEDENTES

Existen varias herramientas para la administración de sistemas que tienen como usuarios finales a los administradores encargados de la gestión y el mantenimiento de los sistemas. Muchas de ellas son basadas en gestión remota para poder hacer frente a la administración de los sistemas desde cualquier lugar del mundo y no necesariamente desde el contacto físico con la máquina.
No obstante, no son muchas las herramientas que permiten administración en paralelo de sistemas mediante el protocolo SSH. Y si a esto le añadimos la opción de crear roles para categorizar hosts, almacenamiento externo basado en computación en la nube, y diferentes espacios de trabajo como un menú interactivo o un intérprete de comandos para la gestión y administración de sistemas, es cuando CLOUDBOTNET es la novedad y única en este ámbito.

1.2 OBJETIVOS

El objetivo del proyecto es la creación de una herramienta destinada a la administración de sistemas en paralelo mediante el protocolo SSH en un entorno multi-sesión. Para esto necesitamos una herramienta capaz de hacer diferentes operaciones sobre uno o varios hosts de forma simultánea.

Los objetivos y los requisitos de la herramienta son:

- Listar hosts
- Ejecutar comandos en uno o varios hosts
- Abrir una consola de comandos en un host
- Acceso externo basado en computación en la nube
- Geolocalización de los hosts
- Administración de roles por host

1.3 BOTNET

1.3.1 Definición

Se puede definir como un avanzado software malicioso que incorpora diversas técnicas propias de virus, gusanos, troyanos y Rootkits proporcionando la funcionalidad del sistema comprometido al atacante.

Una de las características que definen a un Bot es que se conectan a un servidor central o a otras máquinas infectadas después de comprometer el sistema con éxito, formando así una red. Esta red es la denominada red de Bots. Los Bots son manejados por una o más personas denominadas Botmasters o Botherders, las cuales transmiten las órdenes a los Bots a través de una infraestructura de red centralizada o distribuida.

Una función típica que las Botnets proporcionan a sus Botmasters incluye la extracción automatizada de las credenciales de la víctima, la distribución organizada de correo no deseado (Spam), la capacidad de participar en ataques de denegación de servicio, o la extensión de la red de Bots mediante la contratación de nuevos Bots. La motivación para estas actividades se debe principalmente a los intereses financieros de los Botmasters.

1.3.2 Historia

Las Botnets llevan conviviendo con nosotros desde hace más de una década. Su historia se remonta como mínimo a 1999 con la aparición del caballo de troya, coloquialmente llamado troyano, el famoso Sub7. Este fue uno de los primeros troyanos que se conocieron en el mundo de internet, y el cual permitía secuestrar la máquina victima e interactuar con la capa de aplicación.

La primera infraestructura de redes Bots se orientó entorno al protocolo por excelencia de comunicación en tiempo real basado en texto, el conocido IRC (Internet Relay Chat) de los primeros años y auge de internet. Estas primeras Botnets funcionaban de la siguiente manera:

1 – El Botmaster creaba un servidor IRC privado.
2 – Las máquinas infectadas por la Botnet conectarían a ese servidor IRC en segundo plano, de forma que el usuario infectado no se daría cuenta de lo que realmente ocurre.
3 – El Botmaster ejecutaría en el servidor IRC los comandos que quisiera ejecutar en las víctimas.

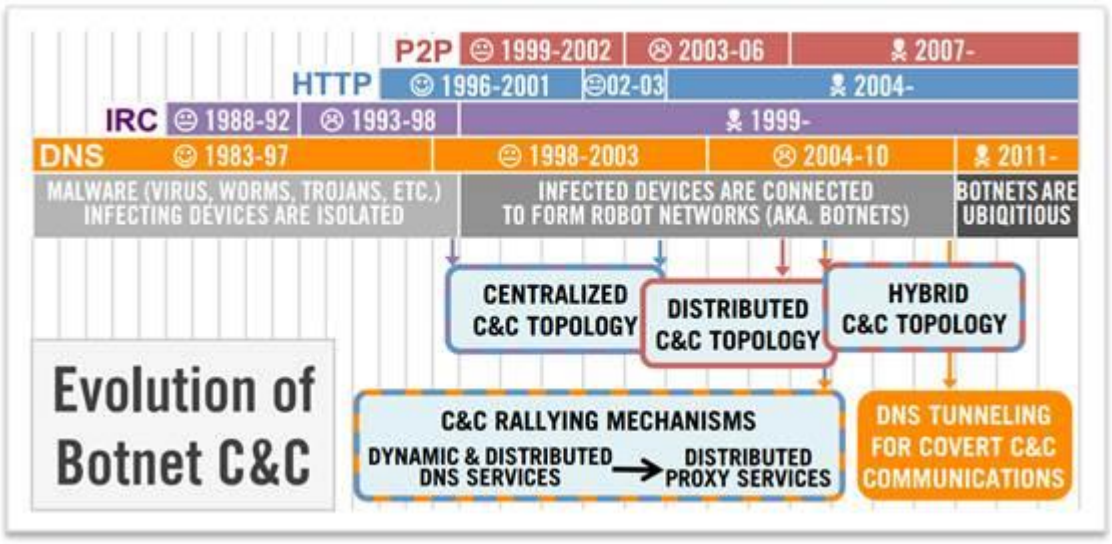
Un troyano\Botnet que hacía esto es el conocido GTBot. Este troyano era descargado por los usuarios que habían sido previamente engañados pensando que estaban instalando algo bueno para sus máquinas. No más lejos de la realidad estaban instalando un troyano que hacía uso de HideWindow para ejecutar mIRC en segundo plano y conectarse al servidor IRC del Botmaster para recibir las órdenes a través de él.

Después de un tiempo las empresas de seguridad se dieron cuenta de que los criminales estaban usando el protocolo de IRC para propósitos malintencionados. Y pusieron su atención en colocar seguridad detrás de ellos a través de firewalls y otras medidas de seguridad. Fué así como los criminales y delincuentes se vieron obligados a evolucionar a otras infraestructuras como HTTP, SSL o ICMP.

En el año 2003 en pleno auge de internet, las redes Botnets poco tenían que ver con las

conocidas en los finales de los años 90. Pasaron de ser redes pequeñas y pruebas de concepto, a redes de distribución de gran alcance con objetivos más concretos. Nacieron las primeras organizaciones criminales basadas en cibercrimen. Se especializaron en Spam, Malware, Adware, Spyware,
Pornware, y en definitiva toda práctica ilícita que reportara beneficios en términos de dinero. Esto pudo llevarse a cabo con las mejoras de rendimiento de los ordenadores CPU y la mejora de la conexión a internet, cada vez más rápidas. Ataques de denegación de servicio (DDos) también comenzaron a ser un práctica muy habitual de estas redes Botnets.

Las redes de Bots también pueden existir sin un servidor command-and-control (C&C) mediante el uso de arquitecturas peer-to-peer (P2P) y otros canales de gestión, lo que hace que sea más difícil interceptar al Botmaster que hay detrás de ella. C&C usa una arquitectura centralizada, es este el primer tipo de Botnets que se crearon. Con el tiempo pasaron a una arquitectura distribuida como puede ser peer-to-peer creando Botnets más potentes y versátiles.



Evolución Command & Control Botnet

1.3.3 Infraestructuras

La parte más importante de una Botnet es la denominada infraestructura de comando y control (C & C). Esta infraestructura consta de los Bots y una entidad de control que puede ser centralizada o distribuida. Uno o más protocolos de comunicación son utilizados por los Botmasters para comandar los sistemas víctima y para coordinar sus acciones.

La infraestructura C & C centralizada se requieren que los Bots mantengan una conexión estable dentro de esta infraestructura con el fin de operar de manera eficiente. Por lo tanto, la arquitectura de la infraestructura C & C determina la robustez, estabilidad y tiempo de

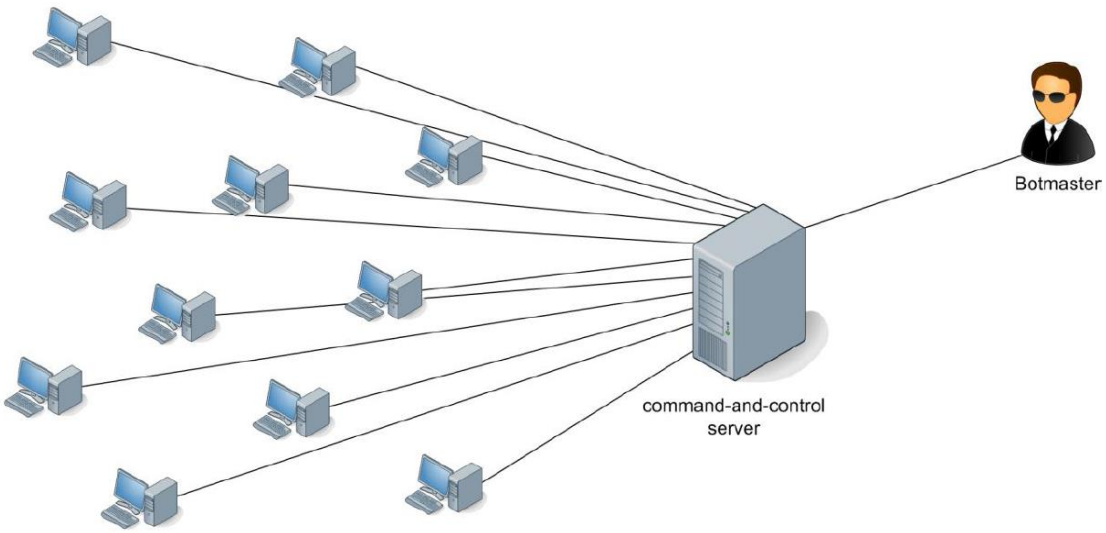
reacción. En general, los enfoques centralizados y descentralizados pueden ser distinguidos. El enfoque centralizado es comparable con el modelo clásico de red cliente-servidor. En estas redes, los Bots actúan como clientes y se conectan a uno o más servidores centrales, de las que reciben sus órdenes.
Modelos descentralizados de C & C, por ejemplo basados en arquitectura P2P, a menudo requieren que los Bots actúen al menos parcialmente de forma autónoma. Los Bots deben mantener la conectividad con otros Bots y emitir solicitudes de nuevos comandos a la Botnet. Debido a que no existe un único conjunto de servidores de comando de control que pueden servir como un punto único de enviar órdenes, el Botmaster puede ocultarse dentro de la Botnet para enviar las órdenes.

1.3.3.1 Infraestructura C&C Centralizada

En una infraestructura C&C centralizada, todos los Bots tienen que conectarse a uno o varios servidores centrales de comando y control, el cual está bajo el control del Botmaster. El Botmaster se conecta al servidor de comando y control y desde este envía las órdenes a todos los Bots que se conectaron a dicho servidor. Debido a que todos los Bots son conectados al mismo servidor de control, esto genera algunas ventajas y desventajas.

Ventajas:
- Botmaster pueden comunicarse de forma simultánea con todos los Bots
- Latencia baja
- Buen medio de coordinación

Desventajas:
- Fáciles de mitigar para el sysadmin (administrador del sistema)
- Fácilmente trazable para encontrar el C&C y al Botmaster
- Fáciles de perder y/o robar la Botnet



Modelo C&C centralizado

Aunque ya no es la infraestructura preferida para los Botmaster por su fácil mitigación, C&C centralizada sigue ocupando un buen porcentaje de las Botnets actuales. El protocolo IRC todavía sirve como una tecnología importante para el control de Botnets y permite un modelo de comunicación centralizado. De acuerdo con el informe de Symantec Internet Security 2010, el 31% de los servidores centralizados C & C que se observaron usan IRC como un protocolo de comunicación.

En cuanto a protocolos de comunicación usados, está muy extendida entre los programadores de Botnets la utilización del protocolo HTTP (Hypertext Transfer Protocol). El 69% de las Botnets con infraestructura centralizada hace uso del protocolo HTTP como protocolo de comunicación según el informe de Symantec Internet Security 2010. Esto es debido a que cualquier ordenador conectado a internet, raramente filtrará el puerto 80, porque es el puerto que utiliza para hacer la navegación web un usuario del sistema. Esta es una buena práctica para un Botmaster con el objetivo de que firewall no corte su comunicación con los Bots.

En algunos casos, una Botnets se organiza en múltiples niveles. Por ejemplo, en lugar de un solo servidor central C & C, no puede haber una infraestructura de servidor. Esta infraestructura por lo general tiene un diseño jerárquico y puede incluir, por ejemplo, los servidores dedicados para la orquestación de los Bots en subgrupos, similar a un equilibrador de carga, y más servidores para la distribución de contenidos, como las plantillas de spam.

1.3.3.2 Infraestructura C&C Descentralizada

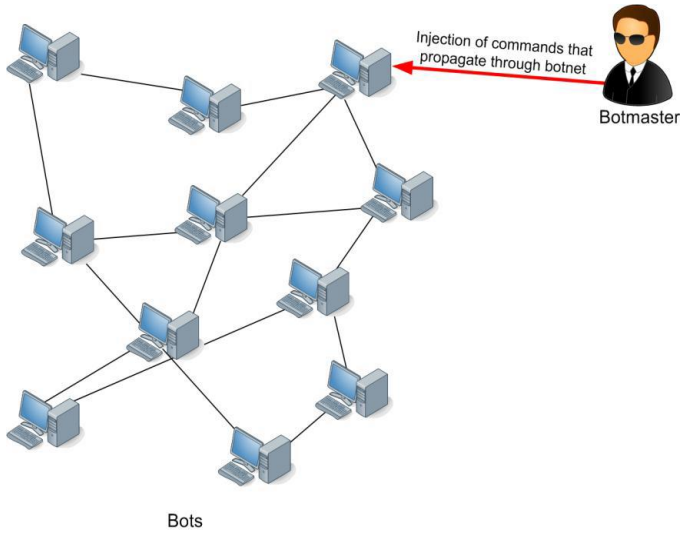
En las arquitecturas C&C descentralizadas, son los enlaces entre los propios Bots los que permiten la comunicación dentro de la red de Bots, proporcionando la base para su organización. El modelo de red usado para esta arquitectura es el conocido peer-to-peer (P2P). El conocimiento de los Bots participantes se distribuye a través de la propia red formada por todos los Bots. En consecuencia, la información no puede ser dirigida a todos los Bots al mismo tiempo, y los comandos tienen que ser inyectados a un Bot para que posteriormente se propage por toda la Botnet. La inyección de un comando se hace desde un punto arbitrario (Bot) como ya hemos comentado, por lo que la localización de la Botmaster es casi imposible. Esto proporciona un alto grado de anonimato.
Con respecto a la robustez, las Botnets peer-to-peer tienen la gran ventaja de que no hay un servidor central puede ser atacado directamente para mitigarlos. Por otra parte, basándose en la auto-propagación de comandos a través de la Botnet significa un tiempo de reacción alto.

Ventajas:

- Anonimato
- Robustez
- Difícil de mitigar

Desventajas:

- Mayor latencia
- No permite coordinación simultanea



Modelo C&C distribuida

1.3.3.3 Infraestructura C&C Híbridas

Son Botnets como su nombre bien indica hacen uso de las dos arquitecturas conocidas, centralizada y distribuida.

Hay al menos un caso conocido, la Botnet SpamThru, en el que se utiliza la arquitectura peer-to-peer como un canal de respaldo. En el caso de SpamThru, los servidores centralizados opcionales se utilizan, además, para el mando. Esto hace de SpamThru una Botnet híbrida.

1.3.4 Tendencia Actual

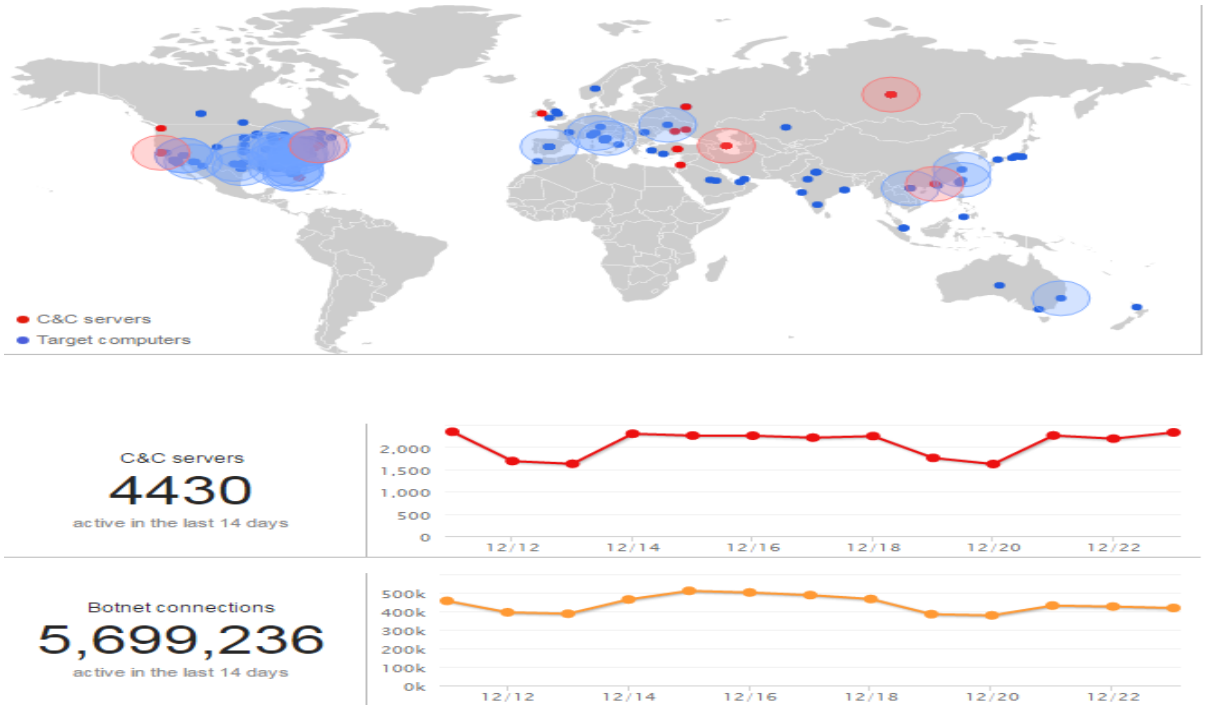
En un principio las Botnets se creaban como un juego entre la comunidad underground, ganaba el que tuviera la Botnet con mayor número de zombies y no se usaban para un fin en concreto. Con el paso del tiempo las Botnets se han convertido en un arma de hacer dinero. Se pueden alquilar Botnets por horas para hacer una ataque de denegación de servicio, para hacer Spam, o para un sin fin de objetivos. En la actualidad los dueños de estas redes de Bots ganan millones de dólares al año haciendo publicidad invasiva, por ejemplo de la viagra. Es un negocio muy rentable, que mueve miles de millones al año en todo el planeta.

A continuación vamos a detallar algunas de las características más importantes que tienen las Botnets de nuestros días:

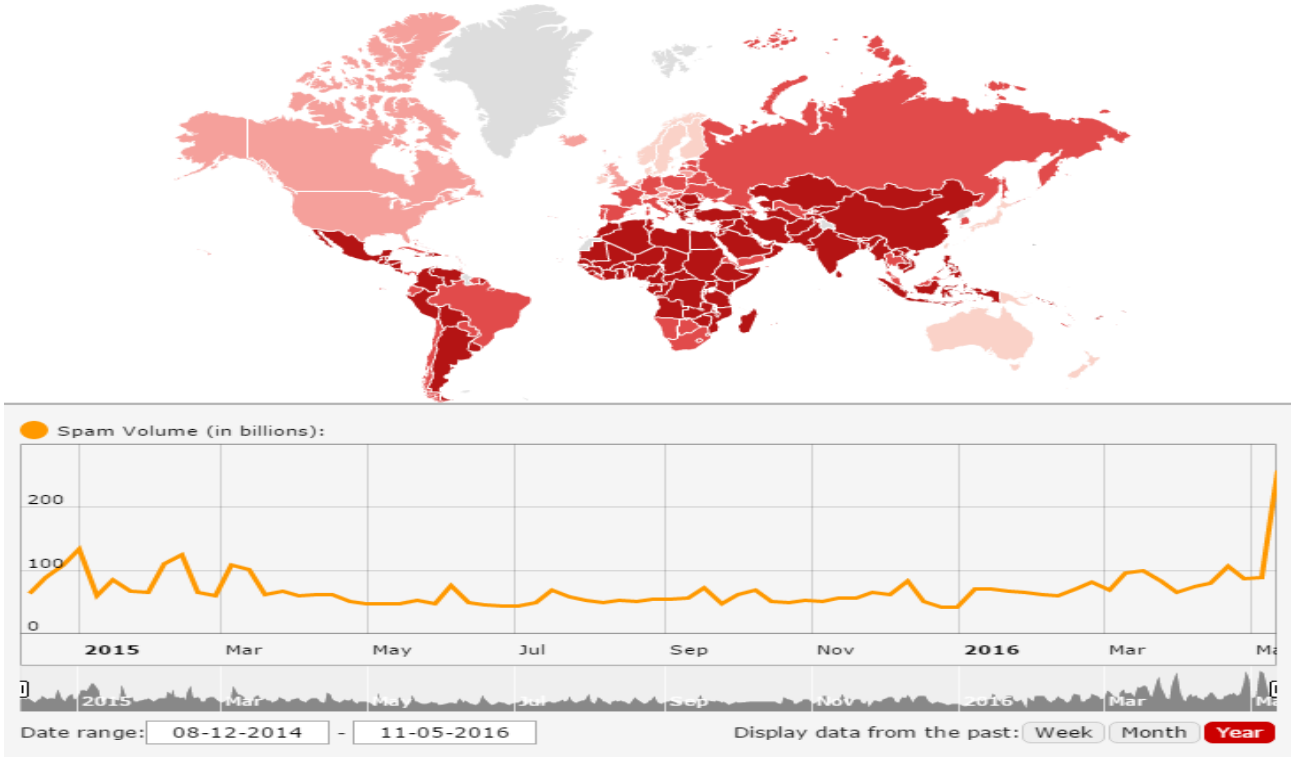
- Diferentes modelos de arquitectura:
C&C centralizado: son fáciles de programación, fáciles de localizar, interceptar e inhabilitar.
P2P distribuido: permiten al dueño recuperar la Botnet en caso de pérdida, dificultan el rastreo de la persona que hay detrás de ella, y evita que otras personas le puedan robar la Botnet.
- Backdoors: lo que permite ejecutar cualquier acción sobre las máquinas. Esto implica la posibilidad de hacer ataques Ddos, robo de información, minería de bitcoins, etc.
- Diferentes protocolos de comunicación: comenzaron con IRC, actualmente se suele usar HTTP (S) porque no suelen ser bloqueados por los firewalls.
- Ocultación de tráfico mediante cifrado: evita que un administrador de sistemas pueda detectar una infección de sus equipos fácilmente, para eso usan SSL o VoIP.
- Mecanismos de propagación: Botnets versátiles son capaces de añadir más zombies a sus redes de forma automatizada.
- Rootkits: algunas Botnets tienen la capacidad de modificar el comportamiento del sistema, haciéndose ocultas para cualquier administrador de sistemas.

Las Botnets siguen evolucionando, y son una de las grandes preocupaciones de la comunidad de la seguridad. Grandes empresas de seguridad y analistas siguen el rastro de las nuevas amenazas y buscan maneras de combatir a este enemigo silencioso que no duda en poner en jaque a gobiernos, instituciones y usuarios domésticos.

A continuación vamos a ver dos gráficas que nos permitirán hacernos una idea del impacto de estas redes de zombies. La primera muestra el número de ordenadores infectados detectados, junto la localización de sus C&C. La segunda muestra el volumen de Spam por continente. Ambos gráficos son del escáner de tráfico de TrendMicro. Obviamente este no es el total de Botnets y Spam en todo el mundo, es solo una pequeña parte de lo que se ha podido detectar por solo una empresa de seguridad. Es una muestra insignificante con respecto al volumen real.



Sistemas infectados (TrendMicro)



Volumen de Spam (TrendMicro)

2. CLOUD COMPUTING

2.1 INTRODUCCION

A menudo los periódicos, revistas y telediarios mencionan información relacionada con la "nube" por lo que se ha hecho evidente la conciencia que empieza a tener la población sobre la importancia de este tema. El término 'En la nube' ha entrado en nuestro lenguaje coloquial. Es posible que hayamos escuchado que el gobierno estadounidense haya puesta en marcha "una iniciativa de nube" o que casi el 75 por ciento de los desarrolladores de Microsoft están trabajando actualmente en productos "relacionados con la nube", o de que un teléfono o servicio almacena sus datos en la nube. Lo que hace que la nube represente el futuro de la computación moderna.

En los últimos años la computación en la nube ha pasado de ser un concepto de negocio prometedor a uno de los segmentos con mayor crecimiento en la industria del IT pero con el aumento de información y datos almacenados en la nube sobre individuos y empresas, las preocupaciones están empezando a crecer sobre la seguridad del entorno. A pesar de todo el bombo que rodea el Cloud los clientes siguen siendo reacios a desplegar su negocio en la nube.
El problema de la seguridad en la computación en nube ha jugado un papel importante en ralentizar su aceptación. Desde un punto de vista la seguridad en la nube se podría mejorar debido a la centralización de datos y un mayor número de recursos enfocados en la seguridad, por otra parte las preocupaciones persisten sobre la pérdida de control sobre datos sensibles y la falta de seguridad en los núcleos almacenados en la nube y encomendados a los diferentes proveedores de Cloud. Si estos proveedores no hacen el trabajo necesario para asegurar sus entornos sus clientes podrían estar en un serio problema. La medición de la calidad de los proveedores de nube en cuanto a la seguridad es un tema difícil porque muchos de los proveedores de nube no exponen sus infraestructuras a los clientes por lo que el cliente tiene que tener una total confianza en el proveedor elegido.

2.2 DEFINCIÓN

Uno de los mayores retos es la definición del Cloud Computing (computación en la nube) ya que al igual que otros términos dentro de la industria de la tecnología todo el mundo tiene su propia definición. Una definición objetiva y oficial de este término es la que nos aporta el Instituto Nacional de Estándares y Tecnología (NIST), una agencia de la Administración de Tecnología del Departamento de Comercio de EE. UU.:

"*Cloud Computing es un modelo para habilitar acceso conveniente por demanda a un conjunto compartido de recursos computacionales configurables, por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios, que pueden ser rápidamente aprovisionados y liberados con un esfuerzo mínimo de administración o de interacción con el proveedor de servicios*"

Con otras palabras se podría definir al Cloud Computing como un conjunto de servicios mediante el cual los recursos compartidos, software e información pueden ser accedidos por ordenadores y otros dispositivos como un servicio medido a través de Internet.

El uso de la palabra nube hace referencia a dos términos principales:

- **Abstracción**: La abstracción corresponde a olvidar los detalles de la implementación por parte de los usuarios y los desarrolladores, tomando este concepto desde un enfoque en donde las aplicaciones se ejecutan sobre una maquina física que no está especificada, los datos son almacenados en ubicaciones desconocidas, la administración de los sistemas está bajo responsabilidad de un tercero y finalmente los usuarios tienen exceso a esta infraestructura desde cualquier lugar con acceso la red.

- **Virtualización**: se refiere a la habilidad del sistema para crear sistemas que parezcan independientes ante los usuarios a través de mecanismos de compartir y asignar periodos de uso a los recursos que cada unidad necesita.

2.3 HISTORIA

El Cloud Computing no ha sido una tecnología descubierta de manera repentina, sino que ha evolucionado a través de una serie de fases que incluye tecnologías como la computación Grid, la computación Utilitaria, prestación de servicios de aplicaciones y software como servicio, etc. Sin embargo, el concepto global de lanzar recursos de computación a través de una red global se inicia en los años sesenta.

En el año 2020 se prevé que el mercado del Cloud Computing supere los 241 mil millones de dólares. Pero cómo hemos llegado hasta aquí y cómo empezó todo es la historia del Cloud Computing.
La historia del Cloud Computing no es tan antigua, la primera página web de servicio de negocio de Cloud Computing (Salesforce.com y Google) fue lanzada en 1999. La computación en la nube va ligada directamente al desarrollo de Internet y el negocio de

tecnología ya que el Cloud Computing es la solución al problema de cómo Internet puede ayudar a mejorar el negocio de tecnología.

El negocio de tecnología tiene una larga y fascinante historia, pero el desarrollo que más influyó en la historia del Cloud Computing empieza con la aparición de los ordenadores como proveedores de soluciones reales a los negocios.

- **Principios del 1960**

El científico de computación John McCarth, aparece con el concepto de "Timesharing"(Tiempo compartido), permitiendo a las organizaciones usar simultáneamente un Mainframe. Este tipo de computación se describe como una significante contribución al desarrollo de Internet y a posteriori el Cloud Computing.

- **1970**

Apareció la virtualización y el uso de Software de virtualización como VMware. Llegó a ser posible arrancar más de un sistema operativo simultáneamente en un entorno aislado, a su vez fue factible arrancar completamente un ordenador diferente(Máquina Virtual) dentro de un sistema operativo.

- **1990**

Se lanza la World Wide Web para uso general (1991). Sun Microsystems inicia el slogan "The Network IS the Computer", transformando la idea de World Wide Web a World Wide Computer. Salesforce.com introdujo el concepto de entrega de aplicaciones empresariales a través de una página web. (1999)

- **1997**

La primera definición y el uso académico de la expresión "Cloud Computing", fue durante una conferencia de Ramnath Chellappa (actual profesor de la universidad Emory) en Dallas en el año 1997.

- **1999**

La llegada de Salesforce.com en 1999 fue pionera en el concepto de ofrecer aplicaciones empresariales a través de una simple página web, lo que se denominó posteriormente software como servicio (SaaS). Este servicio abrió el camino para que otras empresas especialista en software pudieran ofrecerlo.

- **2000**

Después de la burbuja del dot-com , Amazon jugó un papel clave en desarrollo del Cloud Computing restaurando sus centros de datos. Habiendo constatado que la arquitectura

Cloud dio un significante avance en la eficiencia interna de la compañía, Amazon inició un programa de desarrollo de productos nuevos para ofrecer Cloud Computing a clientes externos.

- **2006**

En 2006, Amazon expandió su servicio Cloud. Primero fue con su proyecto denominado "Elastic Compute Cloud EC2" que permitió a los usuarios el acceso a ordenadores y la ejecución de sus propias aplicaciones en ellos, todo ello en el Cloud. Posteriormente lanzaron un servicio de almacenamiento, Simple Storage Servicio (S3). Esto introdujo el modelo "pago sobre la marcha" tanto a los usuarios como a la industria en su conjunto, lo que ha llevado a que se convierta en una práctica actual.

- **2008**

Eucalyptus, una API compatible con la plataforma AWS se convierte en el primer software de código abierto para el despliegue de nubes privadas .Más tarde aparece OpenNebula ,el primer software de código abierto para desplegar nubes privada e híbridas. OpenNebula es un kit de herramientas de Cloud Computing para la gestión de centros de datos heterogéneos y distribuidos

- **2013**

El mercado de servicios de nube pública en todo el mundo asciende a 131.000 millones de dólares, un 18,5 mayor respecto al año 2012. La Infraestructura como servicio (IaaS) ha sido el servicio con crecimiento más rápido en el mercado.

2.4 CARACTERISTICAS

Una gran cantidad de empresas y proveedores de servicios han estado tratando de sacar provecho de la popularidad de la nube. Muchos proveedores afirman que ofrecen servicios en la nube, aunque en realidad no lo hacen. El hecho de que una aplicación sea basada en la Web no quiere decir que se trate de una aplicación en la nube. Las aplicaciones y los servicios alrededor de la nube deben prestar ciertas características antes de que puedan ser consideradas como una verdadera implementación en la nube. La definición NIST del Cloud Computing describe cinco características fundamentales: Auto-servicio por demanda, Acceso amplio desde la red, Conjunto de recursos, Rápida elasticidad y Servicio medido. Estas cinco características deben estar presentes en una oferta para que sea pueda tener en cuenta como una verdadera oferta de Cloud.

- **Auto-servicio por demanda**

Un consumidor puede solicitar y recibir acceso a una oferta de servicios sin que un administrador o algún tipo de personal de apoyo tengan que interactuar con la solicitud manualmente. Los procesos de solicitud y cumplimiento están automatizados. Esta característica en los servicios da una ventaja tanto al proveedor como al consumidor.
La implementación de un auto-servicio permite a los clientes acceder de manera fácil y rápida al servicio que ellos desean, es una práctica muy atractiva en la nube ya que en los entornos tradicionales las peticiones tardan días o semanas en ser cumplidas , causando retrasos en los proyectos e iniciativas.
El auto-servicio libera a los administradores de las tareas rutinarias diarias en cuanto a la creación y gestión de las peticiones de los usuarios, esto permite el personal de TI de una organización centrarse en otras actividades más estratégicas.
La construcción de este tipo de servicios puede ser difícil y costoso, pero para los proveedores les vale la pena esta inversión en tiempo y dinero. Es un servicio que funciona a través de portales de usuarios, donde hay varios portales fuera de la caja que se pueden utilizar para proporcionar la funcionalidad requerida, pero en algunos casos se necesitan un portal personalizado. En la parte de "front-end" se les presenta a los usuarios una plantilla que les permite insertar la información apropiada y en la parte "back end " el portal se interconectará con las API's correspondientes publicadas por las aplicaciones y servicios.
Cuando se interactúa con auto-servicios hay que tener una cierta precaución en el potencial de cumplimiento y las cuestiones reglamentarias, es decir, hay que tener un control para evitar que un usuario tenga la posibilidad de utilizar ciertos servicios o realizar ciertas acciones sin previa aprobación. Como resultado, algunos procesos no pueden ser automatizados completamente.

- **Acceso amplio desde la red**

Los servicios Cloud tienen que ser fácilmente accesibles, a los usuarios solo les requiere que tengan una conexión de red básica para conectarse a servicios o aplicaciones. Podría ser una conexión de Internet, aunque el ancho de banda de una conexión de internet sigue creciendo, siguen siendo relativamente lentas comparadas con las redes de área local (LAN).Por lo tanto el proveedor no debería exigir a los usuarios un gran ancho de banda para utilizar el servicio.
Los servicios Cloud deben ser accesibles por los clientes a través de una amplia variedad de dispositivos. Portátiles y ordenadores de escritorio no son los únicos dispositivos que tienen conexión a redes y a Internet, los usuarios también se conectan a través de Tablets y Smartphones y todos estos dispositivos deberían ser soportados también.

- **Conjunto de recursos**

El conjunto de recursos ayudan a ahorrar costes y permiten flexibilidad en el lado del proveedor. Esta propiedad está basada en el hecho de que los clientes no tienen una necesidad constante de que todos los recursos estén disponibles para ellos. Cuando los

recursos no se están usando por un cliente, en vez de estar improductivos, se podrían usar por otro. Esto da la capacidad a los proveedores de servir muchos más clientes.
Normalmente el conjunto de recursos se logra mediante la Virtualización, se alojan múltiples sesiones virtuales en un único sistema, donde los recursos de un único sistema físico se pueden compartir con múltiples sistemas virtuales.

- **Rápida elasticidad**

La rápida elasticidad describe la capacidad de un entorno Cloud a crecer fácilmente para satisfacer la demanda del usuario. El despliegue en la nube ya debería tener la infraestructura necesaria en lugar de ampliar la capacidad del servicio. Si el sistema está diseñado correctamente, esto sólo podría implicar la adición de más recursos como ordenadores, discos duros, y similares. La clave está en que a pesar de que los recursos están disponibles, no se utilizan hasta que se necesite. Esto permite al proveedor ahorrar en costes de consumo (es decir, en energía y refrigeración).
Esta característica se logra generalmente a través de un mecanismo de disparadores. Cuando el uso de recursos alcanza un cierto punto un disparador se pone en marcha, este disparador inicia automáticamente un proceso de expansión de la capacidad, una vez que el uso haya disminuido la capacidad se reduce según sea necesario para garantizar que los recursos no se desperdicien.

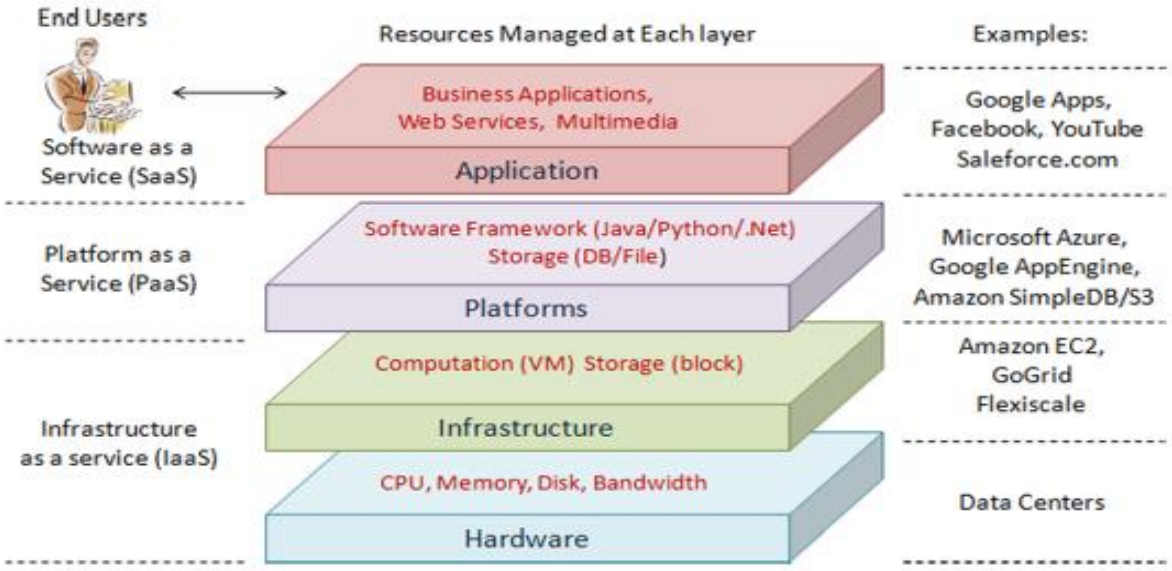
- **Servicio medido**

Los servicios Cloud deben tener la capacidad de medir su uso. El uso se puede determinar con diferentes métricas como el tiempo empleado, el ancho de banda gastado y los datos manejados. La característica del servicio medido es lo que hace que la función “pago por uso” del Cloud Computing sea posible. Una vez que una métrica apropiada ha sido identificada, se determina una tasa, esta tasa se utiliza para fijar la cantidad que debe ser cargada de un cliente. De esta manera el cliente factura basándose en los niveles de consumo. Si el servicio no se usa en un día particular, al cliente no se le carga nada por este tiempo.

2.5 ARQUITECTURA CLOUD COMPUTING

En esta sección se describe los diferentes modelos del Cloud Computing:

2.5.1 Modelo de Capas



Hablando en términos generales, la arquitectura del entorno Cloud Computing se puede dividir en 4 capas:

- **Capa Hardware:** esta capa es responsable de la gestión de los recursos físicos de la nube, incluyendo los servidores físicos, routers, conmutadores, sistemas de energía y enfriamiento. En la práctica, la capa de hardware es normalmente implementada en los centro de datos. Un centro de datos en general contiene miles de servidores que se organizan en bastidores y están interconectados a través de conmutadores y routers. Los problemas típicos en la capa de hardware incluyen la configuración del hardware, la gestión del tráfico, la energía y la gestión de los recursos de refrigeración.

- **Capa de Infraestructura:** conocida también como la capa de virtualización, la capa de infraestructura crea una agrupación de almacenamiento y recursos informáticos mediante la partición de los recursos físicos usando tecnologías de virtualización como Xen, KVM, y VMware entre muchas . La capa de infraestructura es un componente esencial del Cloud Computing ya que muchas de las características claves como la

asignación dinámica de recursos solo están disponibles a través de tecnologías de virtualización.

- **Capa de Plataforma:** construida por encima de la capa de infraestructura, la capa de plataforma consiste en sistemas operativos y Frameworks de aplicaciones. El propósito de la capa de plataforma es reducir al mínimo la carga de las aplicaciones directamente en contenedores VM (Máquinas virtuales). Por ejemplo, Google App Engine opera en la capa de plataforma para proporcionar soporte API para llevar a cabo el almacenamiento, bases de datos y la lógica de negocio de una aplicación web.

- **Capa de Aplicación:** Es el nivel más alto de la jerarquía, la capa de aplicación se compone de las aplicaciones que se ejecutan en la nube. A diferencia de las aplicaciones tradicionales, las aplicaciones en la nube pueden aprovechar la función de escala automática para lograr mejor rendimiento, disponibilidad y menores costes de operación. En comparación con los entornos tradicionales de servicios de alojamiento como las torres de servidores dedicadas, la arquitectura del Cloud Computing es más modular. Esta arquitectura modular permite al Cloud Computing soportar una amplia gama de aplicaciones al tiempo que reduce los requisitos generales de gestión y mantenimiento.

2.5.2 Modelos de Servicio

Según el Instituto Nacional de Estándares y Tecnología (NIST) hay tres principales modelos de servicio Cloud: Software como Servicio (SaaS), plataforma como servicio (PaaS) e Infraestructura como Servicio (IaaS). Estos son los tres modelos de servicio Cloud originales. Pero una cosa que hay que recordar es que, dado que se trata de proveedores de servicios, casi todo es negociable. Los servicios existentes se cambian y se añaden nuevos servicios para satisfacer las necesidades del cliente. A medida que el mercado del Cloud crece, tenemos que reconocer la existencia de muchos otros modelos de servicio en la actualidad. En este apartado se cubren los más prevalentes:

2.5.2.1 Software como Servicio (SaaS)

Es un modelo de distribución de software en el que las aplicaciones son implementadas por un proveedor de servicios y puestas a disposición de los clientes a través de Internet.
Mucha gente considera SaaS como el servicio inicial en la nube. El modelo SaaS es similar al modelo antiguo de Proveedor de Servicios de Aplicaciones (ASP), aunque existen unas diferencias claves. En primer lugar en las aplicaciones ASP se necesitan una infraestructura

y cliente especial para acceder a las aplicaciones sin embargo la mayoría de las aplicaciones SaaS de hoy en día son aplicaciones basadas en la web y no requieren ningún cliente especial para acceder a ellas. Por otra parte, en el modelo ASP, por lo general los clientes acceden a diferentes instancias de una aplicación, sin embargo, en SaaS los clientes acceden a la misma aplicación, simplemente hay diferentes particiones o vistas de la aplicación.

Características

Dependiendo del proveedor de servicios y del servicio ofrecido, las características pueden variar ligeramente, pero se cubren a continuación los escenarios más comunes:

- **Personalización:** con las implementaciones SaaS, el proveedor de servicios normalmente controla todo lo relacionado con la aplicación. En muchos casos, esto limitará cualquier personalización que se puede hacer. Sin embargo, dependiendo de la aplicación, es posible que se pueda solicitar que la interfaz de usuario (UI) o la apariencia de la aplicación puedan modificarse ligeramente. En un entorno SaaS, permitir la personalización puede ser muy costoso para el proveedor de servicios y, por lo tanto, para el cliente.

- **Soporte y mantenimiento:** en los entornos SaaS las actualizaciones de Software son centralizadas y realizadas por el proveedor de servicios. Esto puede presentar un pequeño problema ya que cuando el proveedor decide que es hora de actualizar pocas opciones que hacer tiene el usuario , si hay tiempo de inactividad asociado con la actualización simplemente hay que aceptarlo. Algunas actualizaciones presentan cambios, por lo cual los usuarios se tienen que adaptar al nuevo diseño o cambio.

- **Analítica:** la analítica y las estadísticas de uso pueden proporcionar información sobre el valor de uso de las aplicaciones. En implementaciones SaaS, el proveedor tiene la capacidad de ver las actividades del usuario y determinar las tendencias. En muchos casos, esta información se comparte con los clientes. Para las grandes organizaciones, esta información puede ser muy valiosa. Es importante comprender las tendencias de uso ya que les ayuda a entender cuándo se puede tener un aumento en el uso y, por tanto, un aumento en los ingresos.

Ciclo de vida de un SaaS : 1. Análisis de Requisitos 2. Arquitectura y reglas de negocio 3. Gestión de recursos y tiempo 4. Desarrollo de la aplicación 5. Despliegue y soporte

2.5.2.2 Plataforma como Servicio (PaaS)

Es un servicio Cloud en el que se les ofrece a los clientes una plataforma para satisfacer sus necesidades informáticas. A la capacidad computacional y de almacenamiento, hay que añadir un conjunto integrado de software que incluye todo lo que necesita un desarrollador para construir una aplicación.

Características

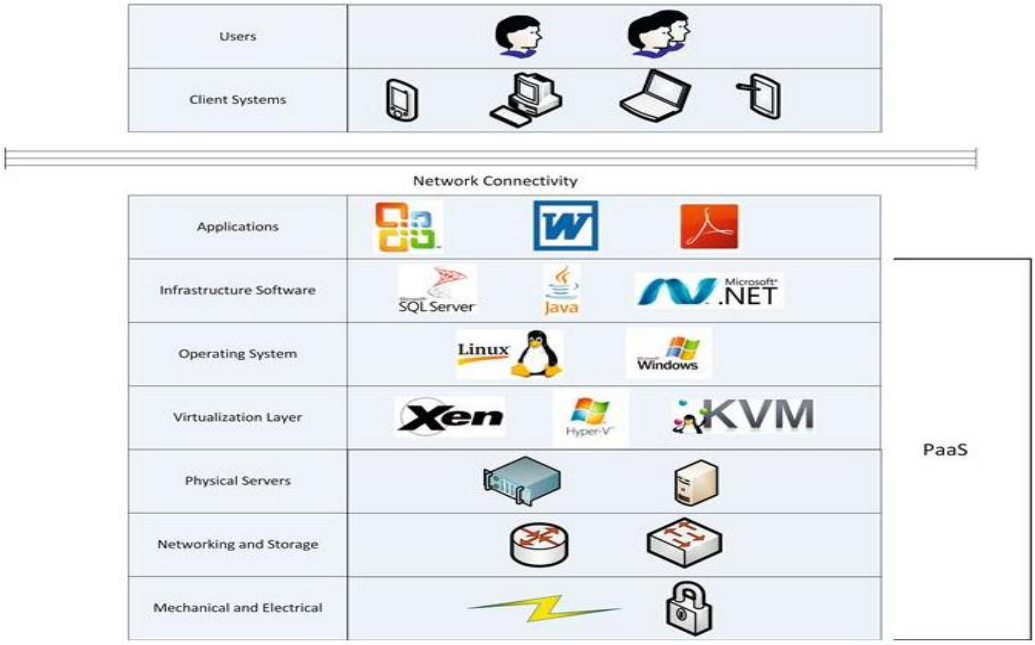
Las implementaciones PaaS permiten a las organizaciones crear y desplegar aplicaciones web sin tener que construir su propia infraestructura. Las ofertas PaaS generalmente facilitan servicios de desarrollo, integración y pruebas.

Siempre hay que tener en cuenta que en un entorno PaaS cuando una organización decide implementar una aplicación o servicio dentro de la plataforma el proveedor del servicio generalmente no tiene ningún control sobre el desarrollo de la aplicación o sobre la calidad del desarrollo. En muchos desarrollos el proveedor puede ofrecer servicios adicionales como por ejemplo servicios de medición del equilibrio de las cargas que ayudan a tener niveles más equilibrados de carga de trabajo en la aplicación distribuyendo la carga a través de múltiples servidores y recursos.

-**Personalización:** Con PaaS, hay un control completo sobre la aplicación, por lo que se puede personalizar de la manera que más se considere oportuna .No existen la posibilidad de hacer muchos cambios en el desarrollo de la plataforma. En la mayoría de los casos la plataforma es estrictamente controlada por el proveedor aunque es probable que haya diferentes opciones de configuración que se pueden establecer, pero la verdadera personalización de la plataforma será limitada.

-**Analítica:** puesto que el cliente es el encargado de crear la aplicación, tiene la posibilidad de monitorizar el uso de la aplicación y determinar las tendencias. Se puede observar qué componentes tienen más uso que otros además de visualizar el uso de la plataforma y determinar la necesidad de añadir nuevos sistemas para manejar la carga correctamente. A menudo los proveedores dan la capacidad al cliente de poner en marcha nuevos sistemas automáticamente cuando los actuales sufren algún exceso de carga.

-**Integración:** en un entorno PaaS, los datos serán almacenados en los servidores del proveedor pero el cliente tendrá un acceso directo a los mismos. En ámbitos donde se necesita el acceso directo a los datos, por ejemplo los informes de inteligencia de negocio, no debería ser un problema en el punto de vista del acceso, pero sí que podrían surgir una serie de problemas relacionados con el ancho de banda cuando se necesita mover grandes cantidades de datos entre el medio interno y el entorno del proveedor, lo que provoca problemas de rendimiento pero no de acceso o funcionalidad.



Diferentes servicios que ofrece un PaaS

2.5.2.3 Infraestructura como Servicio(IaaS)

Los proveedores de IaaS ofrecen servicios "básicos" tales como procesamiento, almacenamiento, redes, y sistemas operativos para que se pueda construir un entorno por encima de estos recursos.

Dentro de los recursos de hardware que ofrecen los proveedores están los servidores que están alojados en los centros de datos del propio proveedor, a los que se puede tener acceso directo a ellos e instalar lo que se necesita. Esto podría ser costoso porque el proveedor no tendría la posibilidad de aplicar tenencias múltiples o economía de escala, por lo tanto los clientes se encargarían de los costes que se pueden generar.
Un modelo más común que utilizan los proveedores es ofrecer el servicio a través de máquinas virtuales en los que el cliente puede interactuar con la infraestructura sin la necesidad de aumentar los costes ya que los proveedores podrían tomar ventaja de las tenencias múltiples. Este tipo de modelo permite alojar diferentes clientes en un mismo conjunto de hardware físico.

2.5.3 Modelos de Implementación

Para atender a las diferentes necesidades de las empresas existen varios tipos de nubes, dependiendo de donde se encuentren instaladas las aplicaciones y qué clientes pueden usarlas tendremos nubes públicas, privadas o híbridas, cada una de ellas con sus ventajas e inconvenientes.

2.5.3.1 Nubes Públicas

Los servicios se encuentran en servidores externos al cliente, sea un particular, empresa o grupo de industria pudiendo tener acceso a los datos y aplicaciones vía aplicaciones web de forma gratuita o pagando una determinada cuota.

Ventajas:

- **Disponibilidad:** las nubes públicas pueden ofrecer mayor disponibilidad sobre lo que puede lograr internamente una empresa. Cada organización tiene un cociente de disponibilidad que le gustaría alcanzar. El problema de la disponibilidad es que tiene diferentes costes como el coste de Hardware, de Software, de formación, o coste del

personal que muchas empresas no se lo puede permitir y por tanto no podrán alcanzar el nivel de disponibilidad que les gustaría. Sin embargo los proveedores de nube pública ya poseen el Hardware, software y el personal para ofrecer sus servicios, aunque exista la posibilidad de tener un coste adicional pero siempre será más rentable que hacerlo internamente.

- **Escalabilidad:** Las implementaciones de las nubes públicas ofrecen una arquitectura altamente escalable, lo que les diferencia de las nubes privadas es la posibilidad de poder escalar la capacidad de una organización sin tener que ampliar su propia infraestructura. Los recursos de la nube están disponibles bajo demanda desde los grandes conjuntos de recursos de las nubes públicas para que las aplicaciones que se ejecutan en ellos pueden responder perfectamente a las fluctuaciones de la actividad.

- **Sistema de costes:** los servicios Cloud públicos utilizan frecuentemente el modelo de pago por lo que se utiliza, a través del cual el cliente puede acceder al recurso que necesite, cuando lo necesite, y sólo paga por lo que usa. Así se evita pagar por una capacidad que no se utiliza.

- **Fiabilidad**: el gran número de redes y servidores involucrados en la creación de una nube pública y las configuraciones redundantes significa que si uno de los componentes físicos falla, el servicio Cloud puede seguir funcionando sin afectar al resto de componentes. En algunos casos, donde las nubes utilizan recursos de varios centros de datos, un centro de datos completo podría estar fuera de línea y los servicios Cloud individuales no estar afectados. En otras palabras, no existe un solo punto de fallo que haga vulnerable un servicio de nube pública.

Inconvenientes:

- **Límites de integración:** en las nubes SaaS públicas los sistemas son externos esto significa que los datos también. Esto podría causar dilemas en ámbitos donde se requiera entre otras cosas ejecutar informes o hacer amplia analítica contra los datos almacenados en la nube. Una solución podría ser transmitir los datos a través de Internet lo que provocaría problemas de rendimiento y seguridad.

- **Tiempo de inactividad forzado:** Cuando se utiliza un proveedor de nube pública, el proveedor controla cuando los sistemas se deberían desconectar para un proceso de mantenimiento. El mantenimiento puede ser realizado en un momento en que no es conveniente para la organización. Es posible que se pueda posponer el mantenimiento de un corto período de tiempo y ponerse de acuerdo en un momento que sea

conveniente tanto para la organización como para el proveedor. Sin embargo, es muy poco probable que el mantenimiento puede ser pospuesto durante un largo periodo de tiempo.

2.5.3.2 Nubes Privadas

Las nubes privadas están completamente gestionadas y mantenidas por el usuario. En general, toda la infraestructura del entorno se encuentra en un centro de datos que el usuario controla. Por lo tanto, el usuario es responsable de la compra, mantenimiento y soporte.

Ventajas:

- **Soporte y solución de problemas:** los entornos en las nubes privadas son más fáciles de solucionar ante posibles problemas que las nubes públicas. En las nubes privadas se puede tener un acceso directo a todos los sistemas, y realizar los métodos convenientes para poder trazar el problema y buscar una solución.

- **Mantenimiento:** En las nubes privadas se puede controlar el ciclo de las actualizaciones. No hay obligación de actualizar cuando no se desea

- **Monitorización:** El hecho de tener un acceso directo a los sistemas del entorno del Cloud privado, se puede realizar cualquier tipo de monitorización desde la aplicación hasta el sistema hardware. Una gran ventaja de esta capacidad es que se pueden tomar medidas preventivas para evitar un fallo en el sistema, por lo que son capaces de ser más proactivo en el servicio a sus clientes.

Inconvenientes

- **Coste:** Implementar una nube privada requiere costes considerables. Hay que implementar una infraestructura en el que no solo debe soportar las necesidades actuales del sistema sino que también las necesidades futuras por lo que es inevitable estimar los requisitos de todas las unidades de negocio que formarán parte del entorno. Dentro de la implementación de la infraestructura hay que tener en cuenta los momentos de pico. No es necesario que los que todos los sistemas utilizados para soportar los momentos de pico estén en ejecución constante sino que haya mecanismo de ejecución automatizada cuando sea necesario

- **Compatibilidad Hardware y Software:** Hay que asegurarse que el Software que se implementa es compatible con el Hardware del entorno.

- **Requerimiento de especialidad:** En las nubes privadas hay que tener una habilidad en todas las aplicaciones y sistemas que se desea implementar. La necesidad de tener esa habilidad puede conducir a una costosa formación y enseñanza. Se tiene que asumir la responsabilidad de la instalación, mantenimiento y soporte para todos los sistemas por lo que hay que asegurarse de que haya conocimientos suficientes dentro de la empresa como para llevar a cabo este tipo de acciones o la posibilidad de contratar consultores externos para realizarlos.

2.5.3.3 Nubes híbridas

A medida que la computación en nube madura, las nubes híbridas serán con mayor probabilidad la implementación de la nube más común. Existen una ligera idea errónea acerca de la nube híbrida, muchas personas piensan que una nube híbrida es un entorno de nube en el que algunos componentes son públicos y otros son privados , pero no es así , un entorno de nube híbrida está formado por múltiples entornos aislados y conectados entre sí.

Al permitir que las cargas de trabajo se puedan mover entre nubes privadas y públicas cuando las necesidades de computación y los costes lo pidan, las nubes híbridas ofrecen a las empresas una mayor flexibilidad y más opciones de despliegue de datos.

Una nube híbrida tiene la ventaja de una inversión inicial más moderada y a la vez contar con SaaS, PaaS o IaaS bajo demanda. En el momento necesario, utilizando las APIs de las distintas plataformas públicas existentes, se tiene la posibilidad de escalar la plataforma todo lo que se quiera sin invertir en infraestructura con la idea de tomar uno de los siguientes caminos:

• Si dicha necesidad llegara a ser de carácter estable, sería recomendable incrementar la capacidad de la nube privada e incorporar los servicios adoptados en la pública pasándolos a la nube propia.

• Si dicha necesidad es puntual o intermitente se mantendría el servicio en los Clouds públicos, lo que permite no aumentar la infraestructura innecesariamente.

Parece que este tipo de nubes está teniendo buena aceptación en las empresas de cara a un futuro próximo, ya que se están desarrollando software de gestión de nubes para

poder gestionar la nube privada y a su vez adquirir recursos en los grandes proveedores públicos.

2.5.3.4 Nubes Comunitarias

Son aquellas en las que la infraestructura tecnológica se comparte entre diversas organizaciones que mantienen objetivos similares, por ejemplo, en materia de requisitos de seguridad.

2.6 CLOUD EN "CLOUDBOTNET"

Después de haber mencionado y detallado los principales tipos de Cloud así como los diferentes tipos de implementación, en este apartado vamos a explicar el uso del Cloud Computing en nuestro proyecto. Nuestra Botnet, contiene una funcionalidad que interactúa con la nube, precisamente con la parte Storage (Almacenamiento) de la plataforma Google Cloud. En el Storage del Cloud almacenamos un repertorio de herramientas Hacking para explotar las máquinas víctimas que forman parte de la Botnet.

Cuando el usuario interactúa con la Botnet puede acceder a la sección de herramientas, y realizar las siguientes funciones:

- Listar las herramientas existentes en la nube.
- Seleccionar una herramienta y descargarla en uno de los Zombies de la Botnet.

2.6.1 Google Cloud Platform

Google Cloud Platform está construido sobre la misma infraestructura de clase mundial que Google ha diseñado, ensamblado y utilizado para ofrecer productos corporativos como el buscador de Google, que ofrece miles de millones de resultados de búsqueda en milisegundos. Google posee también una de las mayores y más extendidas y avanzadas redes informáticas en el mundo. Su red está formada por miles de kilómetros de cables de fibra óptica, redes avanzadas por software (**SDN Software defined networking**) y todo ello para ofrecer un rendimiento rápido, consistente y escalable. Google es también una de las pocas empresas de poseer un cable de fibra óptica privada bajo el Océano Pacífico.

Cloud Platform consiste en un conjunto de recursos físicos, tales como computadoras y unidades de disco duro y recursos virtuales, tales como máquinas virtuales (VM), que están

contenidos en los centros de datos de Google en todo el mundo. Cada ubicación del centro de datos se encuentra en una región del mundo. Regiones que incluyen América Central, Europa Occidental y Asia Oriental.

Google Cloud Platform permite a los desarrolladores de aplicaciones de software construir, probar, implementar y monitorizar las aplicaciones utilizando la infraestructura altamente escalable y fiable de Google. Además, permite que los administradores de sistemas centrarse en el conjunto de soluciones lo que les permite subcontratar el desafiante trabajo de ensamblaje Hardware, mantenimiento y actualización de tecnología para expertos en Google.

2.6.1.1 Comparación con AWS (Amazon Web Services)

- Google Compute Engine , el producto IaaS de Google Cloud Platform , adopta un modelo de cobro por minuto , excepto por el mínimo inicial de 10 minutos . Por otro lado, AWS cobra a base de horas.

- Google Compute Engine es más adecuado para manejar los picos de tráfico. Esto es porque los equilibradores de carga de Compute Engine a diferencia de los de AWS no requieren pre-calentamiento. Además, los equilibradores de carga de AWS requieren que sus usuarios hagan una suscripción a soporte. Los equilibradores de Google tienen la capacidad de escalar su potencial automáticamente cuando se dan cuenta de un repentino aumento de tráfico.

- Los discos persistentes de Compute Engine soportan un tamaño de disco más grande (actualmente 10 TB) en comparación con AWS. Además Google incluye los gastos de E/S dentro del coste de los discos persistentes dando a los clientes un presupuesto. En el caso de AWS el coste de E/S es separado del coste del disco. Otra característica interesante que incluye Google es la capacidad de montar un disco persistente en múltiples máquinas virtuales en modo lectura o en una sola máquina virtual en modo lectura-escritura.

- Las instancias están alojadas como máquinas virtuales en IaaS . Periódicamente, el proveedor de servicios IaaS tiene que hacer mantenimiento en la plataforma. El Hardware también puede fallar de vez en cuando. En tales casos, es deseable tener una migración automática de las máquinas virtuales a otro host físico. Compute Engine es capaz de hacer una migración en directo.

- BigQuery, el programa de análisis de Bigdata, es una plataforma integrada y totalmente alojada que escala a miles de nodos y cobra sólo por espacio y tiempo de cálculo. Sin embargo el programa RedShift de Amazon requiere que los usuarios configuren el sistema y tiene un sistema de cobro por hora en lugar de cobro por uso.

- Los centros de datos de Google se extienden a nivel mundial y están interconectados por una red privada de fibra. Esto significa que el tráfico de red entre dos plataformas de Cloud pasa a través de una red privada y no a través de Internet (público). Hasta hoy, AWS no tiene este tipo de red privada.

2.6.2 Google Cloud Storage

Es un servicio para almacenar y acceder a los datos en la nube. Contiene las características principales que potencian a Google Cloud Platform tales como alta disponibilidad, escalabilidad y rendimiento. Tiene una interfaz de usuario interactiva lo cual es fácil de usar y tiene la posibilidad de subir y eliminar contenido rápidamente.

Google Cloud Storage ofrece un acceso directo al almacenamiento escalable de Google y a la infraestructura de red así como poderosos mecanismos de autenticación e intercambio de datos. Se puede almacenar archivos de cualquier tamaño y gestionar que el acceso a los datos sea de forma individual o en grupo. Los datos almacenados pueden designarse como públicos o privados. Los datos se organizan en Buckets que contienen objetos. Un Bucket es similar a un directorio y un objeto es similar a un fichero, a diferencia de los sistemas de archivos normales, los Buckets no se pueden anidar unos dentro de otros.

Se puede cargar y descargar archivos en Google Cloud Storage de las siguientes maneras:

- Interactuado directamente con el navegador web.
- Desde líneas de comando usando la herramienta Gsutil.
- A través de programación utilizando el API REST de Google Cloud Storage.

2.6.2.1 Características

- **Edge-caching:** los centros de datos de Google y otros nodos se distribuyen en todo el mundo y los datos se entregan a los usuarios desde el lugar más cercano de la tierra
- **Ubicaciones específicas de Buckets** : pueden ayudar a reducir la latencia y por lo tanto aumenta el rendimiento percibido por todo el sistema.
- **Fuerte consistencia de archivos**: los datos están disponibles en el momento en el que se han subido correctamente al Cloud Storage
- **Cifrado en el lado del servidor**: los datos se cifran de forma transparente antes de que se escriban en el disco
- **Subidas de archivos reanudable:** esto te permite seguir con el proceso de reanudación sin tener que consumir más recursos en caso de error.

- **Listas de control de acceso (ACL):** se pueden definir permisos de lectura y escritura a usuarios o grupos. Las reglas se aplican a los buckets y objetos conjuntamente o independientemente.

2.6.2.2 Gsutil

Es una aplicación Python que permite el acceso a Cloud Storage a través de líneas de comando. Se pueden gestionar a través de Gsutil varias tareas de Buckets y objetos:

- Crear y borrar Buckets
- Descargar, Subir y eliminar objetos
- Listar Buckets y objetos
- Mover , copiar y renombrar objetos
- Editar los ACL de objetos y Buckets

2.6.2.3 Usando Google API- OAuth 2.0

Prácticamente todos los productos de Google están construidos de acuerdo con la filosofía de API-First. El acceso al API está sujeto al control de acceso. El control de acceso comprende la autenticación y autorización y se conoce colectivamente como AUTH. Con el fin de usar una API una aplicación debe ser adecuadamente autenticada y autorizada. El nivel de control de acceso depende de si la aplicación está solicitando acceso sólo para una API pública (por ejemplo API Translate) o para una API que tiene acceso a información protegida (Por ejemplo, Cloud Storage). En el primer caso, la aplicación necesita ser autenticada; en el segundo caso, la aplicación necesita ser autenticada y autorizada a acceder a los datos del usuario.

Cada aplicación que intenta acceder al API de Google necesita demostrar su identidad. El nivel de identificación depende del alcance del acceso solicitado por la aplicación. Por ejemplo, para las API de Google Translate en la que no se accede a aplicaciones o datos privados de los usuarios, el nivel de identificación es una clave de API sencilla. Una aplicación que necesita acceso a información protegida (Cloud Storage) debe utilizar un proceso de identificación basado en OAuth2.0

El protocolo OAuth2.0 fue creado con la intención de proporcionar el acceso a contenido protegido en una manera estandarizada y abierta. Este protocolo fue adoptado por Google para permitir el acceso a sus API, ofreciendo una manera de autenticación y autorización a los agentes externos interesados en el intercambio de información con las API de Google.

Los siguientes pasos describen el proceso completo de solicitar el acceso a contenidos específicos:

1. El cliente solicita la autorización del propietario del recurso.
2. El propietario del recurso envía una concesión de autorización.
3. El cliente utiliza esta concesión de autorización para solicitar un token de acceso al servidor de autorización.
4. Si el proceso es exitoso, el servidor de autorización proporciona al cliente el token de acceso.
5. El cliente tiene acceso a contenido protegido, se autoriza a las peticiones con el token de acceso que acaba de adquirir.
6. Si el token es válido, el cliente recibe la información solicitada.

3. TECNOLOGÍAS Y ARQUITECTURA DE DESARROLLO

En este apartado se mencionan las principales tecnologías que se han usado para desarrollar el proyecto.

3.1 PYTHON

Python es un lenguaje de programación multiplataforma. Permite varios estilos: programación orientada a objetos, programación funcional y programación imperativa.
Python usa tipado dinámico y conteo de referencias para la administración de memoria.
Python tiene una resolución dinámica de nombres, enlaza un método y un nombre de variable durante la ejecución del programa, también conocido como ligadura dinámica de métodos.

Con Python se puede escribir extensiones con facilidad en otros lenguajes de programación como C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

Características

- Sintaxis limpia y legible

- Lenguaje multiplataforma
- Capacidad de Introspección
- Orientación a Objetos bastante intuitiva
- Soporte de paquetes en jerarquía
- Manejo de errores basado en excepciones
- Tipos de datos dinámicos de alto nivel
- Extensa librería y módulos third-party
- Se pueden escribir nuevos módulos fácilmente en C, C++ (en Java -Jython- y .NET -IronPython-)
- Python puede incluirse en aplicaciones que necesitan una interfaz programable.

3.2 FABRIC

3.2.1 Definición

Fabric es una librería de Python que se utiliza para interactuar con SSH y sistemas informáticos para automatizar una amplia gama de tareas que varía desde el despliegue de una aplicación hasta la administración general del sistema.
Proporciona un conjunto básico de operaciones para ejecutar localmente o remotamente comandos Shell (de forma normal, o través de Sudo) y descargar/subir archivos así como otras funcionalidades auxiliares.
El uso típico implica la creación de un módulo de Python que contiene una o más funciones y a continuación, su ejecución a través de la herramienta de línea de comandos fab.

3.2.2 Funcionalidades

Administración Sistema/Servidor

Una de las áreas clave para el uso de Fabric es la automatización de tareas de administración cotidianas del sistema (y servidor). Estos trabajos incluyen casi todo lo que se relaciona con:

- Creación de un servidor.
- Mantenimiento del servidor.
- Monitorización.

Despliegue de aplicaciones

El despliegue de una aplicación (Independientemente de que sea en una página web, una API, o un servidor) por lo general significa la creación de un sistema desde cero, su preparación mediante la actualización de todo, la descarga de las dependencias, la configuración del archivo estructural y los permisos y por último subir el código base o descargarlo usando un SMC como GIT.

Tener la posibilidad de crear un Script (tanto en local como en remoto), de forma organizada y lo más importante de forma programada, que realice las tareas mencionadas demuestra ser una librería muy valiosa.

3.2.3 Integración con SSH

Cualquier comando Python(o procedimiento) y módulo se puede utilizar a través de Fabric dado el hecho de que Fabric es una librería de Python.

Lo que realmente Fabric tiene de ventaja es su amplia y excelente integración y coordinación con SSH lo que le permite su uso a través de script simples (fabfile.py)

En esta sección, se van a mencionar una selección de operaciones (funciones) que vienen con Fabric y que se pueden utilizar para interactuar con SSH.

1. **run (fabric.operations.run)**

El procedimiento *run* se usa para ejecutar un comando Shell en una o más maquinas remotas.

El resultado de *run* se puede capturar usando una variable. Para saber si el comando falla o termina con éxito se puede averiguar usando *.failed* y *.succeed*

Ejemplo:

```
# Crear un Nuevo directorio
run("mkdir /tmp/trunk/")

# Uptime
```

```
run("uptime")

# Hostname
run("hostname")

# Guardar la salida del comando "ls"
result = run("ls -l /var/www")

# Chequear si el commando ha fallado
result.failed
```

2. sudo (fabric.operations.sudo)

Junto con *run*, probablemente es el comando con más uso. Permite la ejecución de una serie de comandos y argumentos con privilegios Sudo(superusuario) en la máquina de destino.

Si el comando Sudo se usa especificando explícitamente un usuario en concreto, la ejecución no se efectuará con root si no como otro usuario.

Ejemplo:

```
# Crear un directorio
sudo("mkdir /var/www")

# Crear un directorio con otro usuario
sudo("mkdir /var/www/web-app-one", user="web-admin")

# Salida del comando
result = sudo("ls -l /var/www")
```

3. local (fabric.operations.local)

Como se ha mencionado anteriormente, un script de Fabric(fabfile) se puede usar para realizar acciones tanto en local como en sistemas remotos. Para este propósito Fabric proporciona el comando Local para ejecutar comandos de forma local.
A diferencia de los comandos run y sudo, la interacción con la salida del comando local no es posible.

4. get (fabric.operations.get)

El comando *get* es para descargar ficheros del sistema remoto a la máquina donde la librería Fabric está siendo usada. Su comportamiento es similar al comando *SCP*(secure copy) a la hora de descargar backups , acceder al registro de datos o a otra información relacionada con el servidor.
Se puede especificar la ruta de la máquina remota con el argumento *remote_path* y también la ruta de la descarga (máquina local) con el argumento *local_path*

Ejemplo:

```
# Descarga de logs
get(remote_path="/tmp/log_extracts.tar.gz", local_path="/logs/new_log.tar.gz")


# Descargar un fichero backup
get("/backup/db.gz", "./db.gz")
```

5. put (fabric.operations.put)

Cuando se necesita subir archivos , el comand put se puede usar similarmente que el comando get. Se puede acceder al resultado de la ejecución de los comandos con `.failed` o `.succeeded`

- `local_path` – especifica la ruta local.
- `remote_path`- especifica la ruta remota.
- `use_sudo`- subir el archive a cualquier ruta de la máquina remote
- `mode` – especifica el modo del archive(flag)
- `mirror_local`- set the file flags (i.e. make executable) automatically by reading the local file's mode.

Ejemplo:

```
# Subir un archivo tar de una aplicacion
put("/local/path/to/app.tar.gz", "/tmp/trunk/app.tar.gz")
```

```
# Usa el gestor de context ‘cd’ en vez del argument “remote path
# Esto sube app.tar.gz a /tmp/trunk/
with cd("/tmp"):
    put("local/path/to/app.tar.gz", "trunk")

# Sube un archivo y especifica el modo deseado
upload = put("requirements.txt", "requirements.txt", mode=664)

# Verifica la subida
upload.succeeded
```

6. prompt (fabric.operations.prompt)

El comando *prompt* da una cierta flexibilidad a la hora de usar la librería Fabric. Este comando se usa para sugerir y preguntar al usuario (el que está ejecutando el script por ejemplo) que inserte una serie de datos para que sean usados en caso de una ejecución exitosa.

Cuando se está usando por ejemplo un único fichero con múltiples aplicaciones, se puede usar el comando Prompt para indicar las acciones que se puedan tomar como datos de entrada.

Ejemplo:

```
# aviso al usuario
port_number = prompt("Which port would you like to use?")

# sugiere al usuario con los datos de por defecto y el tipo valido
port_number = prompt("Which port?", default=42, validate=int)
```

7. reboot (fabric.operations.reboot)

El comando reboot como su propio nombre indica se usa para reiniciar el sistema remoto. Por defecto espera dos minutos (i.e. 120 seconds -> wait=120) antes de realizar su trabajo

Ejemplo:

```
# Reiniciar el Sistema remoto
```

```
reboot()


# Reiniciar en 30 segundos
reboot(wait=30)
```

Gestores de contexto en FABRIC

Los gestores de contexto se usan con la declaración *with* de Python

1. **cd (fabric.context_managers.cd)**

El gestor de contexto *cd* permite mantener el estado del directorio (Es decir, cuando el siguiente bloque de comentarios se deben ejecutar). Es similar a ejecutar el comando cd durante una sesión SSH y ejecutar varios comandos diferentes.

Ejemplo:

```
# The *cd* context manager makes enwrapped command's
# execution relative to the stated path (i.e. "/tmp/trunk")
with cd("/tmp/trunk"):
    items = sudo("ls -l")


# It is possible to "chain" context managers
# The run commands gets executed, therefore at "/tmp/trunk"
with cd("/tmp"):
    with cd("/trunk"):
        run("ls")
```

2. **lcd (fabric.context_managers.lcd)**

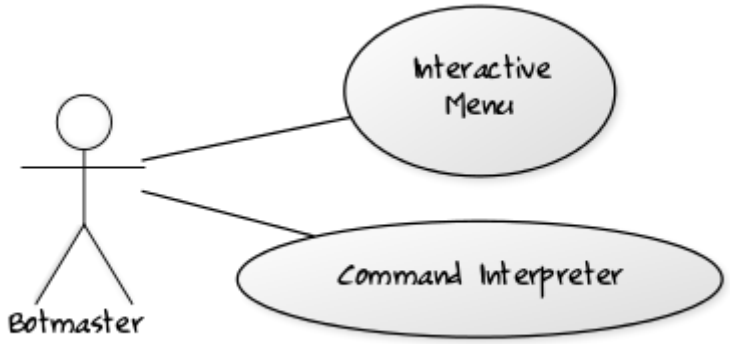
Le gestor de context lcd funciona de manera similar al anterior gestor(cd); sin embargo este gestor solo afecta al sistema local

```
# Change the local working directory to project's
# and upload a tar archive
with lcd("~/projects/my_project")
```

3.3 ARQUITECTURA DE DESARROLLO CLOUDBOTNET

Modos de ejecución

CLOUDBOTNET ha sido diseñada para poder funcionar sobre dos modos de trabajo diferentes. Por un lado, cuenta con un menú interactivo, el cual permite al botmaster ejecutar acciones/órdenes navegando por el menú. El segundo modo de trabajo es un intérprete de comandos, el cual permite ejecutar órdenes ejecutando comandos de consola propios del CLOUDBOTNET.



Espacios de trabajo

Ejecutar la ayuda para el parseo de argumentos:
python CloudBotNet.py -h

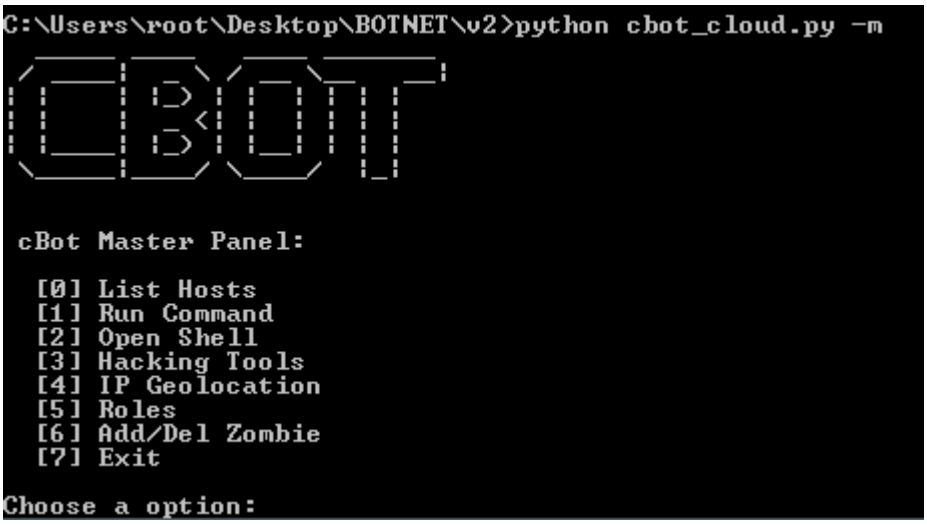
```
C:\Users\root\Desktop\BOTNET\v2>python cbot_cloud.py
execute with argument --help

C:\Users\root\Desktop\BOTNET\v2>python cbot_cloud.py --help
Usage: cbot_cloud.py [options]

Options:
  -h, --help         show this help message and exit
  -m, --menu         Botnet based on menu
  -i, --interpreter  Botnet based on command interpreter

C:\Users\root\Desktop\BOTNET\v2>
```

Ejecutar modo menú interactivo (-m):
python CloudBotNet.py -m



Ejecutar modo intérprete de comandos (-i):
python CloudBotNet.py -i

Programación modular

En CLOUDBOTNET tenemos una parte de programación modular, esto consiste en una seríe de funciones que subdividen el desarrollo de la botnet en partes. Se usa programación modular para el menú interactivo de CLOUDBOTNET. El menú interactivo está compuesto por diferentes funciones que unidas hacen la funcionalidad total del sistema.

El menú invoca a las funciones haciendo uso de una estructura if-elif

```
try:
    choice = int(raw_input("Choose a option: "))

    if choice == 0:
        status()

    elif choice == 1:
        runCommand()

    elif choice == 2:
        list_hosts()
        openShell()

    elif choice == 3:
        cloud()
```

Estructura if-elif menú interactivo

Funciones en el desarrollo de CLOUDBOTNET:

def execCommand(command)

Ejecuta el comando en el bot y devuelve el resultado.

def list_hosts

Muestra los bots que hay en la variable entorno. Es decir, los bots que tenemos en nuestra botnet.

def get_hosts

Funcion que solicita al usuario que seleccione uno o varios hosts. Devuelve esta selección.

def geoIP(tgt)

Carga la IP a geocalizar en la base de datos del archivo 'GeoIP/GeoLiteCity.dat' y muestra el continente, el país, la ciudad, la longitud y la latitud.

def createKML(ip)

Crea una mapa .kml visualizable con Google Maps con la información obtenida de la función GeoIP(tgt).

def cloud

Muestra las herramientas hospedadas en el cloud computing. Deja seleccionar una de ellas y los hosts en los que será descargada.

def uploadFile

Sube la herramienta al bot.

def status

Ejecuta el comando 'uptime' en todas las máquinas de la variable de entorno 'env.hosts', es decir en todos los bots.

def runCommand

Solicita un comando por pantalla para ser ejecutado en uno o varios hosts.

def openShell

Solicita un host para abrir una sesión de consola de comandos en ese host.

def role

Función que permite crear un rol asignándoles hosts y ejecutar un comando sobre un rol.

def adddelHost

Función que permite añadir un bot solicitando sus datos o eliminar un bot existente.

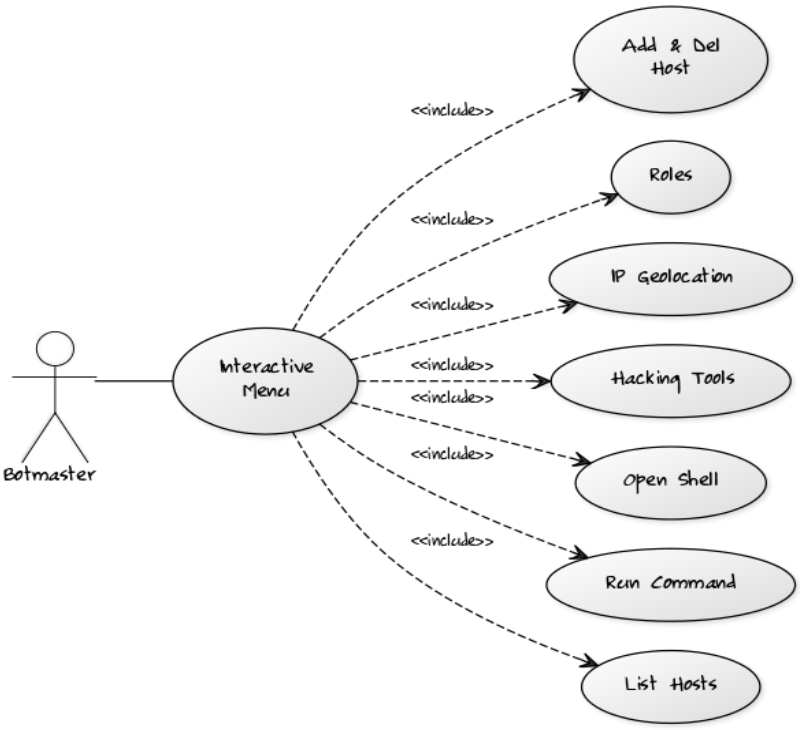


Diagrama caso de uso del menú interactivo

Programación orientada a objetos

La parte desarrollada mediante programación a objetos es el intérprete de comandos. En la cual tenemos una clase por cada comando que hereda de la clase '**Command**'. Cada clase de comando tiene una función de inicialización, una función de ejecución y una función de parseo.

```
class Command(object):

    def __init__(self):
        pass
    def executeCommand(self):
        pass
    def parse(self,s):
        pass
```

Cuando el botmaster introduce un comando en el intérprete de comandos, se crea un objeto de dicho comando si el comando introducido existe como clase. De esta comprobación de existencia se encarga la función **parse(self,s)** de cada clase. Una vez creado el objeto e inicializado, se procede a ejecutar la función **executeCommand(self)**.

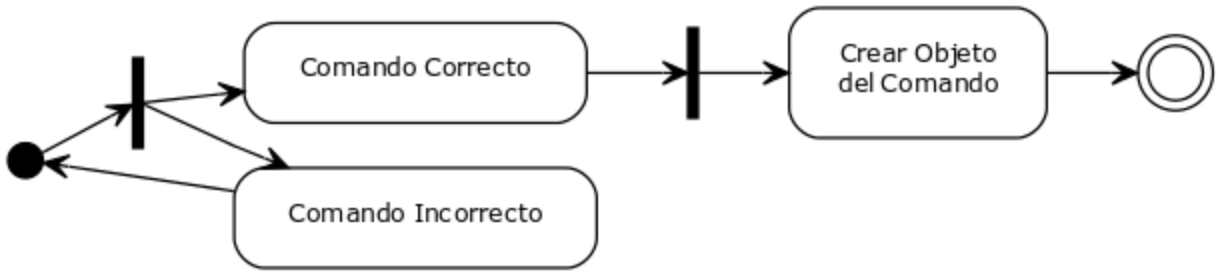


Diagrama de actividad

Clases en el desarrollo de CLOUDBOTNET:

class Command(object)

Contiene la incialización, el parseo, y la ejecución

class OpenShell(Command)

Contiene la incialización, el parseo, y la ejecución para el comando 'openshell'

class Status(Command)

Contiene la incialización, el parseo, y la ejecución para el comando 'status'

class RunCommand(Command)

Contiene la incialización, el parseo, y la ejecución para el comando 'run command c'

class Help(Command)

Contiene la incialización, el parseo, y la ejecución para el comando 'help'

class DownloadTool(Command)

Contiene la incialización, el parseo, y la ejecución para el comando 'download'

class ListTools(Command)

Contiene la incialización, el parseo, y la ejecución para el comando 'listools'

class Role(Command)

Contiene la incialización, el parseo, y la ejecución para el comando 'role'

class Addel(Command)

Contiene la incialización, el parseo, y la ejecución para el comando 'addel'

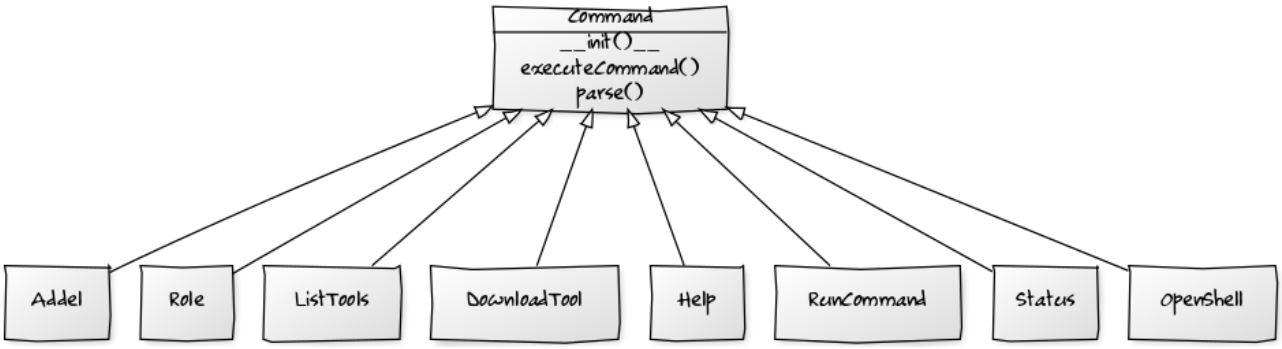


Diagrama de clases

Diagrama caso de uso intérprete de comandos:

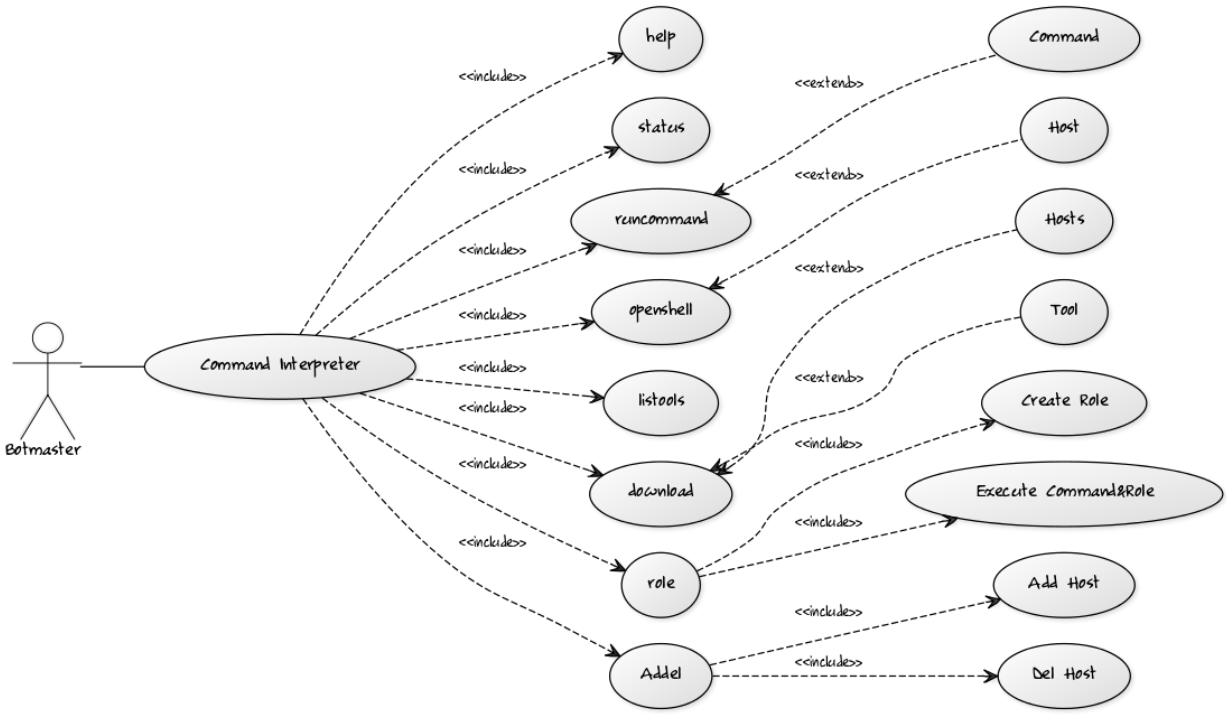


Diagrama caso de uso del intérprete de comandos extendido

Directorio de trabajo, ficheros y bases de datos

CLOUDBOTNET es un único fichero de python .py. Debemos alojarlo en un directorio de trabajo donde python pueda ser ejecutado. CLOUDBOTNET se compone de 3 ficheros principalmente.

- CLOUDBOTNET script
- zombies.db
- GeoLiteCity.dat

CLOUDBOTNET script es el código completo programado en python de CLOUDBOTNET
zombies.db es el fichero de base de datos que usa CLOUDBOTNET para la gestión de bots/zombies.
GeoLiteCity.dat es el fichero de base de datos usado por CLOUDBOTNET para la funcionalidad de geocalización. Este fichero debe estar contenido en una carpeta llamada 'GeoIP'.

El directorio de trabajo debe de quedar así:

/CloudBotNet.py
/zombies.db
/GeoIP/GeoLiteCity.dat

Infraestructura CLOUDBOTNET

En CLOUDBOTNET se hace uso de una arquitectura centralizada para la construcción de la red sobre la que se basa la botnet, con un único C&C de comando y control. De este modo el único C&C es el servidor donde el botmaster ejecuta las acciones y recibe los resultados. Hemos decidido hacer uso de una infraestructura centralizada debido a que CLOUDBOTNET está basada en el protocolo SSH, y realizamos una conexión directa cliente-servidor.

4. MANUAL DE USUARIO Y PREPARACIÓN DEL ENTORNO

En esta sección vamos a ver como se prepara el entorno para poder utilizar correctamente la herramienta CLOUDBOTNET en nuestro equipo. Recordemos que como CloudBotNet está escrito en un lenguaje multiplataforma, el sistema operativo que corra en nuestra máquina no será ningún problema. Recordemos que para instalar muchos paquetes y librerías en python siempre nos podemos apoyar en la herramienta 'pip install paquete/libería' que trae Python para ahorrarnos trabajo en instalaciones manuales.

4.1 PREPARACIÓN DEL ENTORNO

Requisitos:

- Python 2.7.x
- Cuenta con Google Cloud Platform
- Gsutil
- Google Cloud SDK

Para descargar Python 2.7.x debemos dirigirnos a la página oficial:
https://www.python.org/downloads/

Para crear una cuenta con Google Cloud Platform nos dirigimos a:
https://cloud.google.com/

Crear una cuenta con Google Cloud cuesta dinero, no obstante podremos solicitar una cuenta gratuita de prueba.

Gsutil es una aplicacion en python que permite acceder a la plataforma de Google (Google Cloud Storage) através de líneas de comando:
https://cloud.google.com/storage/docs/gsutil

Google Cloud SDK es un conjunto de herramientas que se pueden utilizar para administrar los recursos y aplicaciones alojadas en Google Cloud Platform. Estas herramientas incluyen la herramienta Gcloud de línea de comandos, así como Gsutil y bq.
https://cloud.google.com/sdk/

Requisitos (Librerías):

- Fabric
- Pygeoip
- gcs_oauth2_boto_plugin
- Optparse

- Sqlite 3

Fabric es la librería principal con la cual creamos la red de Bots. Para descargarla y poder trabajar con ella podemos encontrar toda la información en su página oficial:
http://www.fabfile.org/

Pygeoip es la librería usada por la Botnet para calcular la geolocalización de los bots. Hace uso de una base de datos GeoLiteCity.dat, que debe ser añadida a una carpeta en el mismo directorio de trabajo en el que se encuentra la CloudBotNet 'GeoIP/GeoLiteCity.dat':

https://pypi.python.org/pypi/pygeoip
http://dev.maxmind.com/geoip/legacy/geolite/

gcs_oauth2_boto_plugin es un plugin de autenticación para el Framework de boto.Proporciona credenciales de OAuth 2.0 que se pueden utilizar con Google Cloud Storage:
https://github.com/GoogleCloudPlatform/gcs-oauth2-boto-plugin

Optparse es la librería usada para manejar los argumentos de entrada mediante línea de comandos que precisa nuestra CloudBotNet para invocar a nuestro menú interactivo o al intérprete de comandos:
https://docs.python.org/2/library/optparse.html

sqlite 3 es la librería para trabajar con bases de datos SQL en Python. CloudBotNet hace uso de ella para almacenar persistentemente los Bots con su información.
https://docs.python.org/2/library/sqlite3.html

Una vez que tenemos nuestra cuenta de Google Cloud Platform creada, todas las librerías/paquetes instalados, el código CloudBotNet en nuestro directorio de trabajo. Podemos empezar a conectar nuestro código con el Google Storage siguiendo estas instrucciones:

a. Use an existing service account or create a new one, and download the associated private key.

Creating a service account

i. Open the list of credentials in the Google Cloud Platform Console.

OPEN THE LIST OF CREDENTIALS

ii. Click **Create credentials**.

iii. Select **Service account key**.

A *Create service account key* window opens.

iv. Click the drop-down box below *Service account*, then click **New service account**.

v. Enter a name for the service account in **Name**.

vi. Use the default **Service account ID** or generate a different one.

vii. Select the **Key type**: **JSON** or **P12**.

viii. Click **Create**.

A **Service account created** window is displayed and the private key for the **Key type** you selected is downloaded automatically. If you selected a P12 key, the private key's password ("notasecret") is displayed.

ix. Click **Close**.

You need to get the private key in PKCS12 format. By default, when you create a new key, the JSON format of the private key is downloaded.

b. Configure the `.boto` file with the service account. You can do this with `gsutil` :

```
gsutil config -e
```

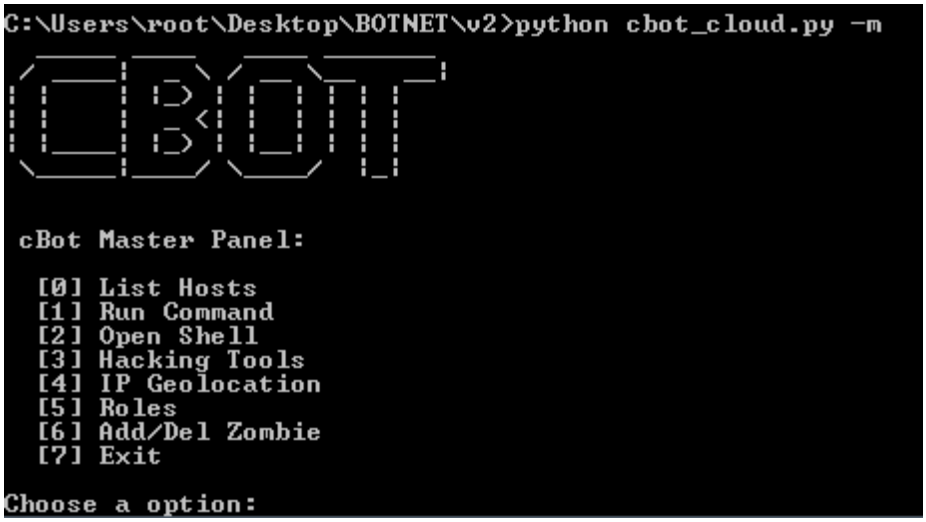
The command will prompt you for the service account email address and the location of the service account private key (.p12). Be sure to have the private key on the computer where you are running the `gsutil` command.

4.2 MANUAL DE USUARIO

En esta sección vamos a ver pantallazos de la herramienta CloudBotNet ejecutándose. Serán dos partes: el modo menú interactivo y el modo interprete de comandos.

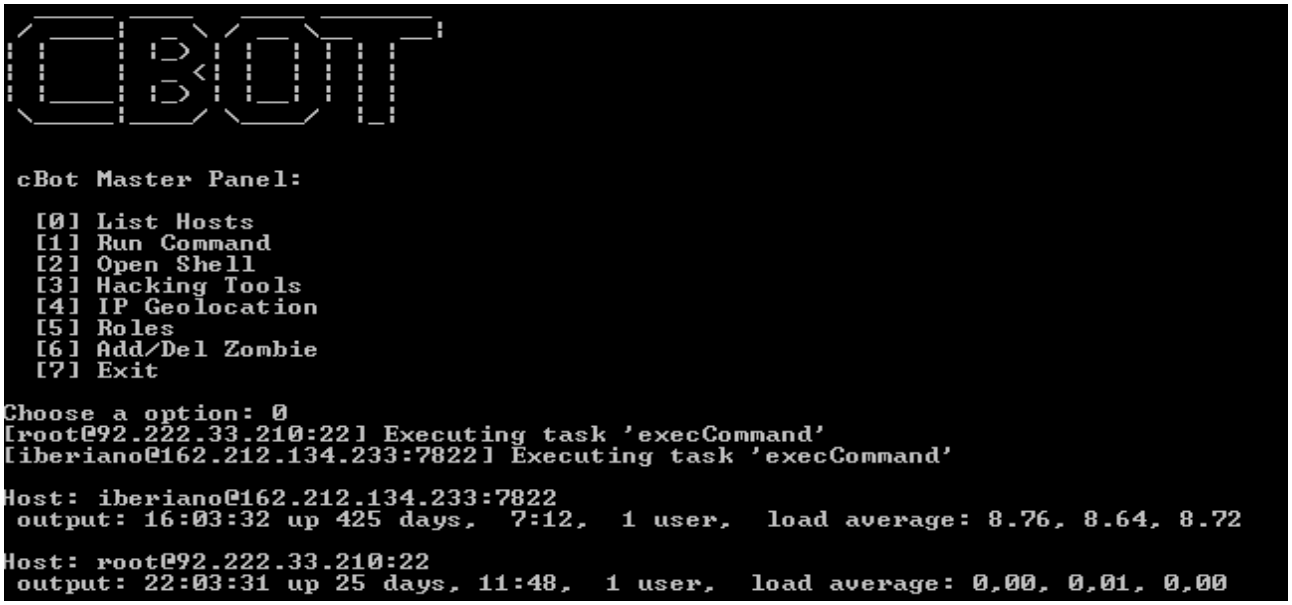
4.2.1 Menú Interactivo

Menú



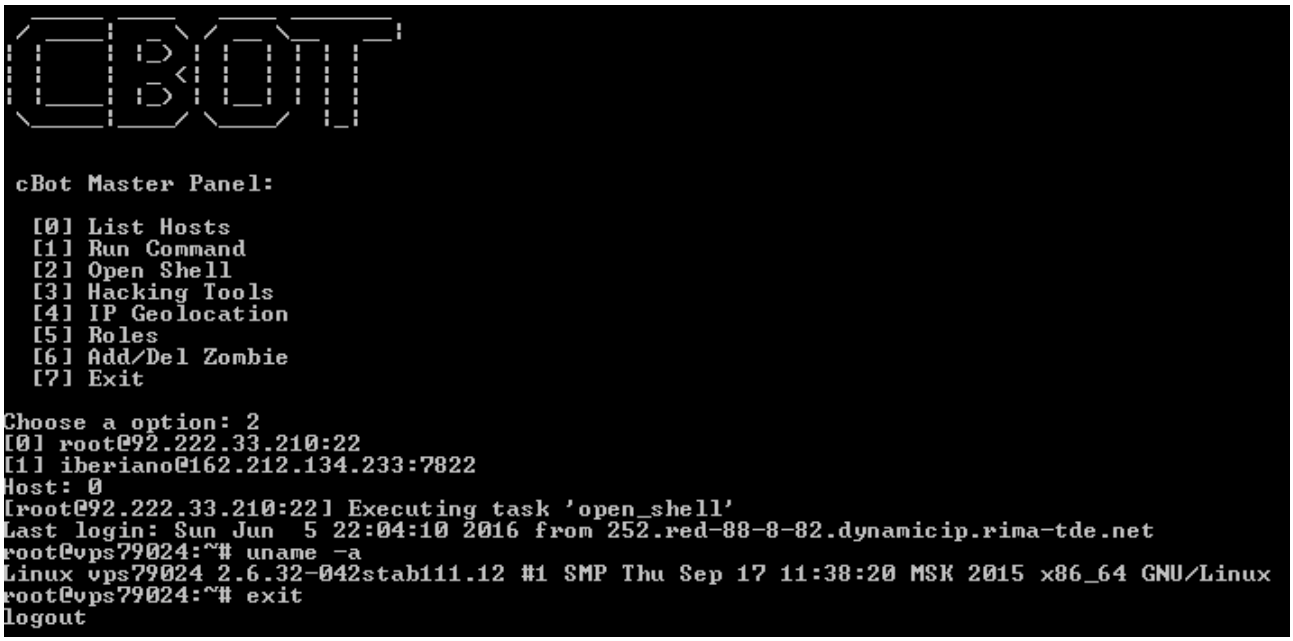
Aquí tendremos la posibilidad de navegar por las diferentes opciones que ofrece el menú.

List Hosts



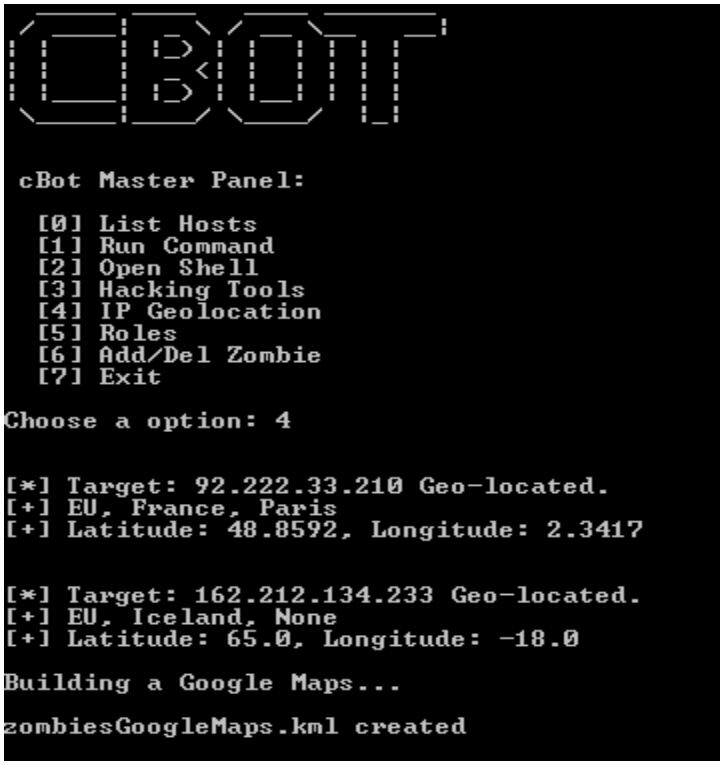
En la opción 'List Hosts' se nos listarán las máquinas de nuestra red con el comando 'uptime'

Open Shell



La opción 'OpenShell' nos ofrece la posibilidad de abrir una consola de comandos un uno de los hosts de nuestra red.

IP Geolocation



La opción 'IP Geolocation' nos creará un archivo .kml visualizable con Google Maps.

Rol

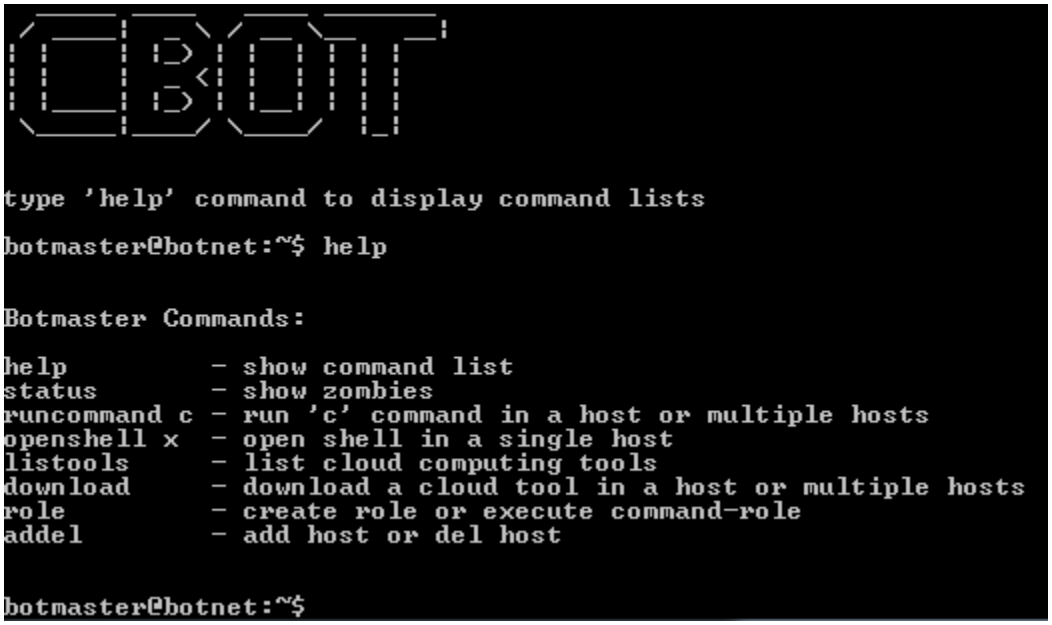
Desde 'Roles' podremos hacer la gestión y la ejecición de comandos por roles.

Add/Del Host

Desde 'Add/Del' tenemos la opción de añadir o eliminar host de nuestra botnet.

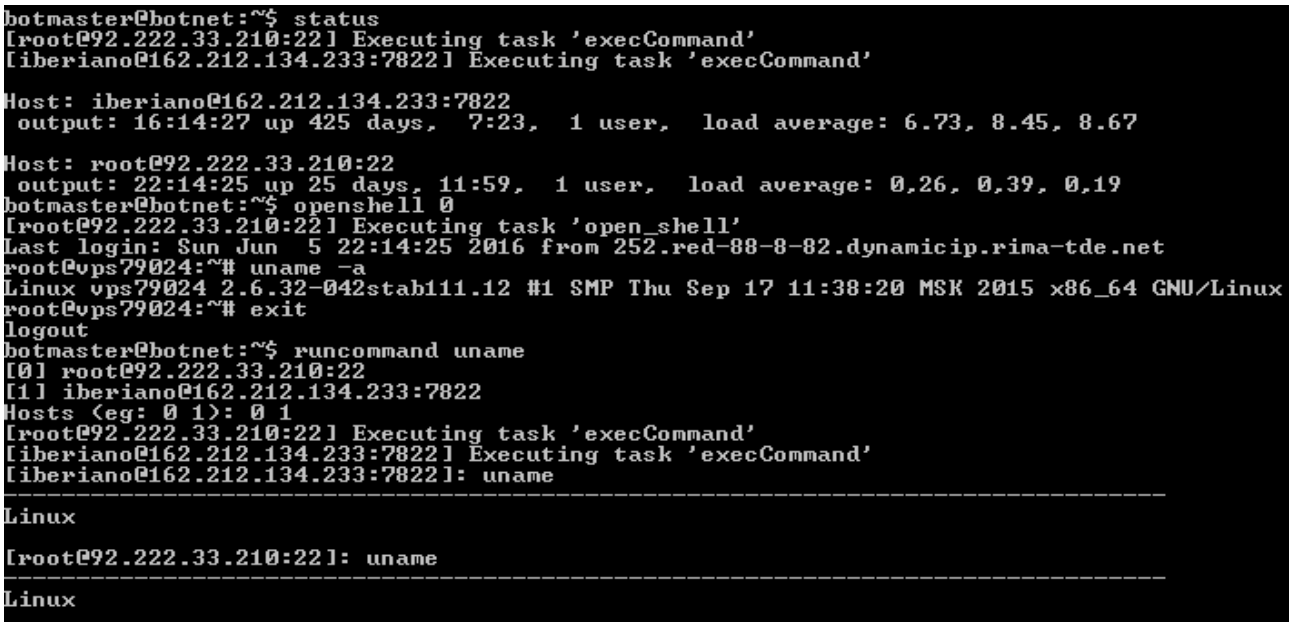
4.2.2 Intérprete de Comandos

Comando 'help'



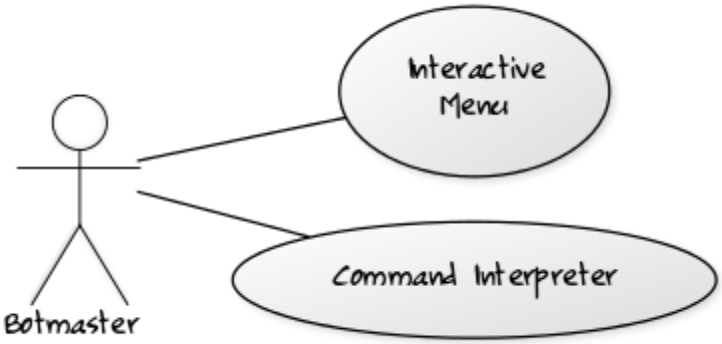
Muestra la ayuda de los comandos disponibles en CLOUDBOTNET.

comandos 'status','openshell 0','runcommand uname'



En este pantallaza vemos el resultado de los tres comandos ejecutado

4.3 DIAGRAMAS DE CASOS DE USO



Espacios de trabajos disponibles para el botmaster

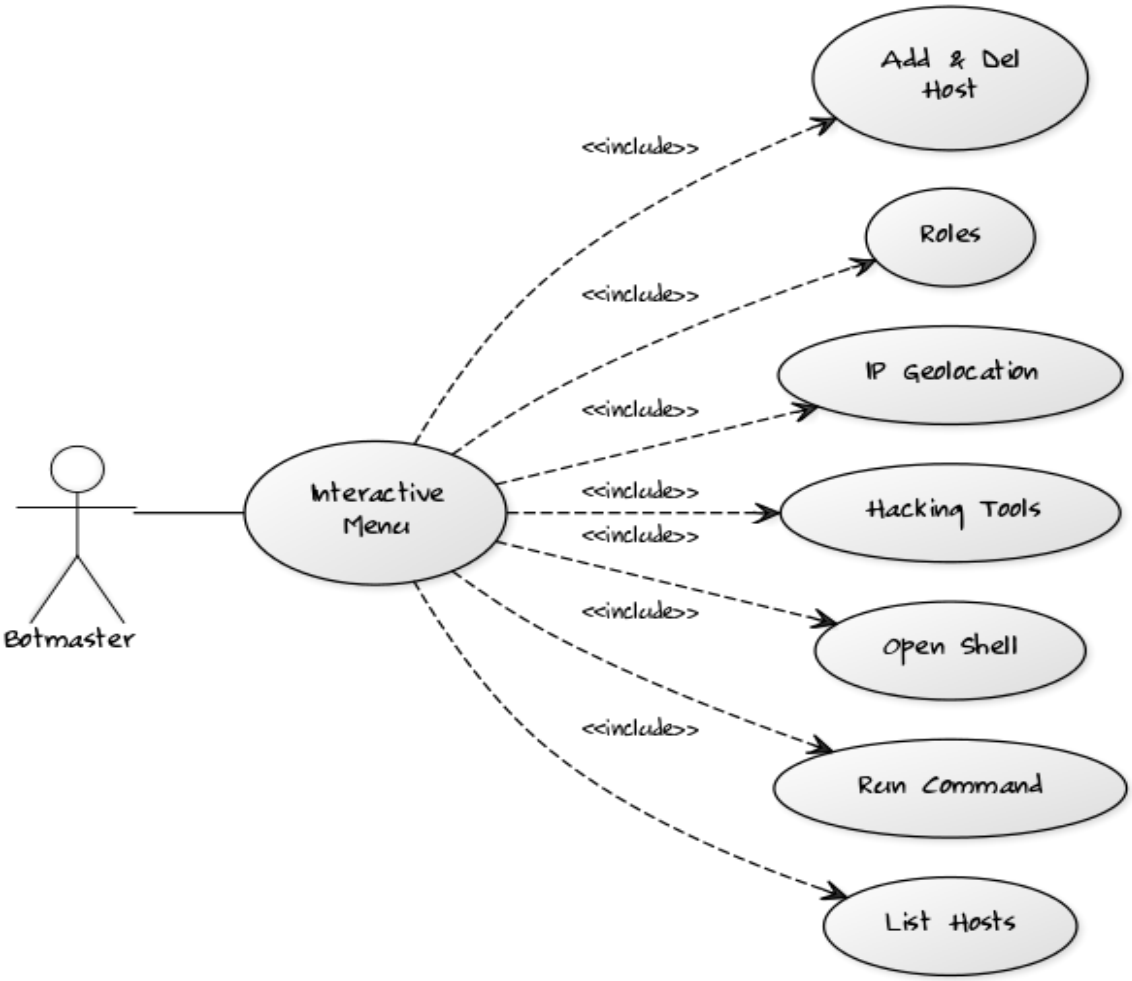


Diagrama caso de uso del menú interactivo

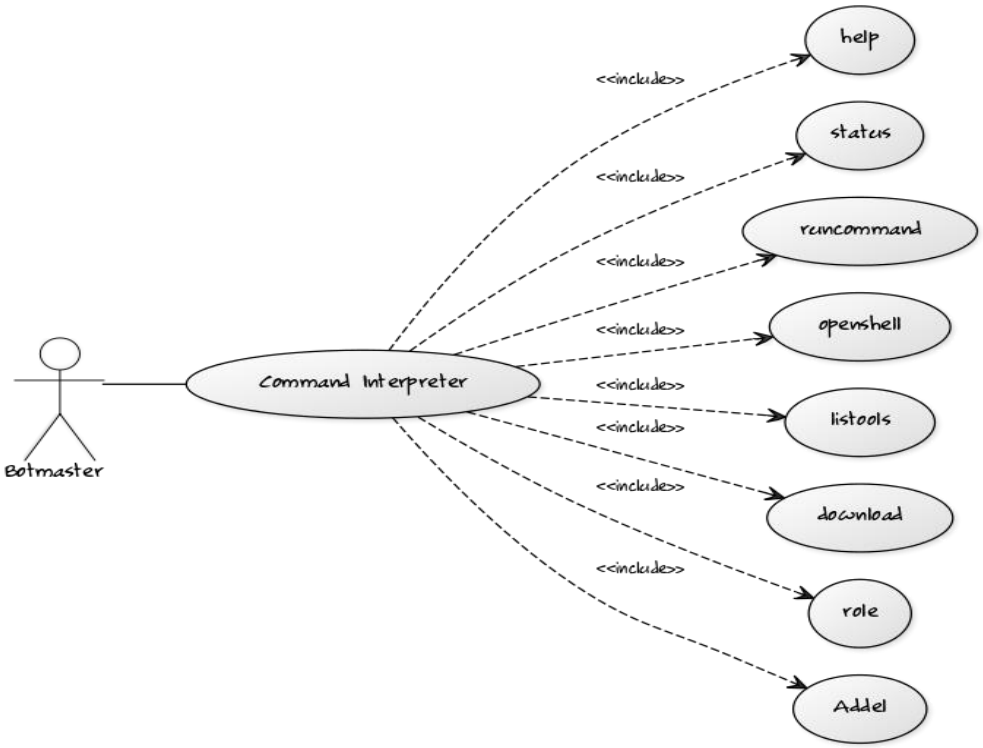


Diagrama caso de uso del intérprete de comandos

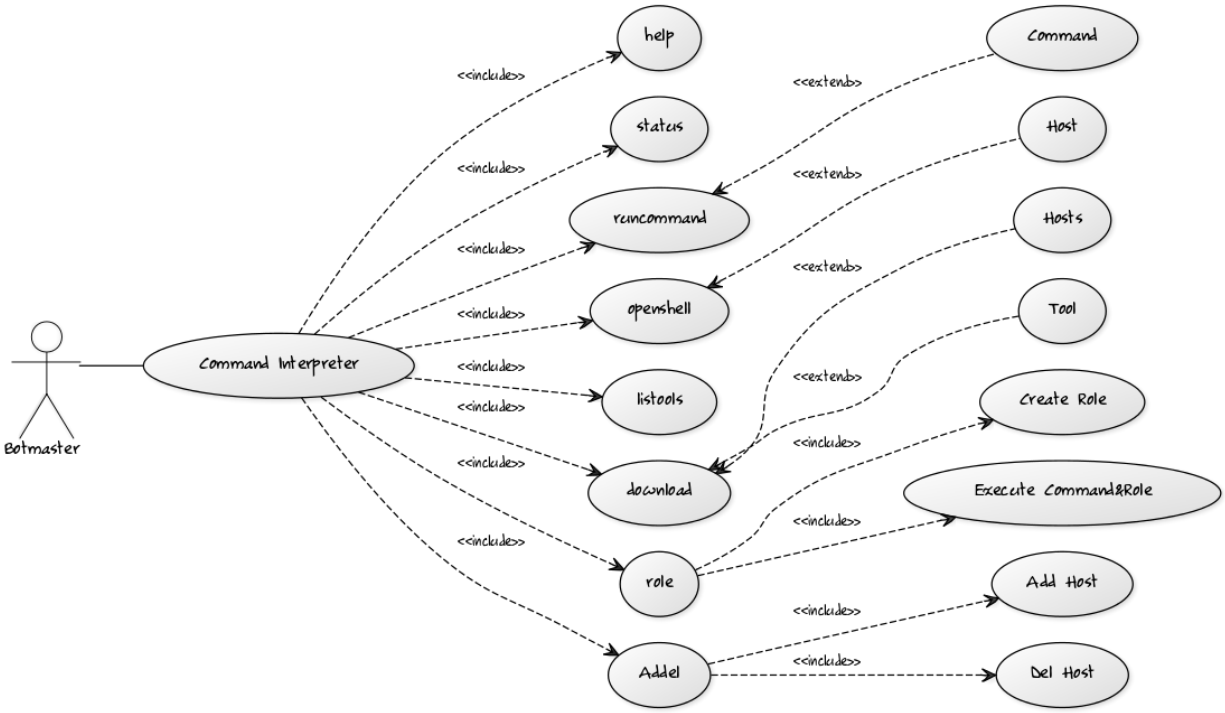


Diagrama caso de uso del intérprete de comandos extendido

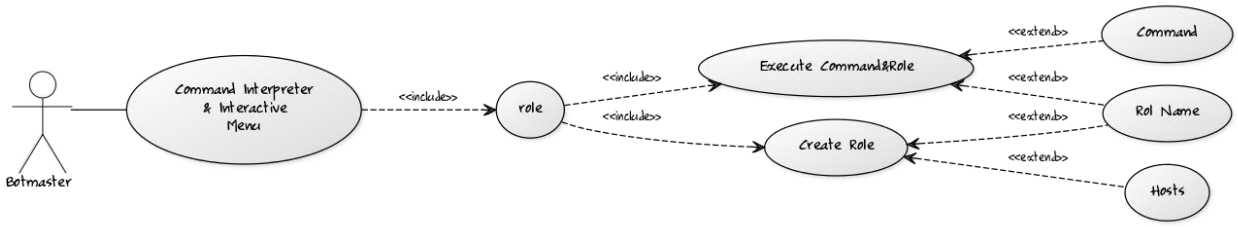


Diagrama caso de uso de la funcionalidad roles

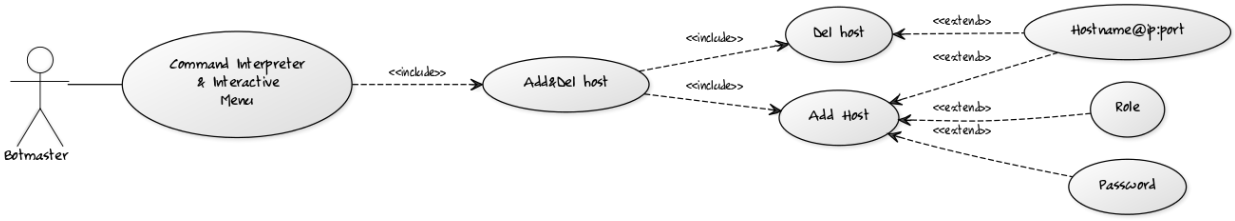


Diagrama caso de uso de la funcionalidad Añadir/eliminar Host

5. CONCLUSIONES

5.1 Valoración Personal

El objetivo de nuestro proyecto ha sido demostrar que mediante el conocimiento y la investigación de unos requisitos mínimos en redes, programación y Cloud Computing, hemos sido capaces de desarrollar una herramienta versátil y robusta, que bien puede ser empleada para la administración de sistemas o la creación de una Botnet. Hemos sumado tres ingredientes que en su justa medida ha dado lugar a una herramienta capaz de facilitar la administración de varios sistemas simultáneamente mediante un entorno multi-sesión basado en el protocolo SSH.

El uso de computación en la nube como soporte de almacenamiento externo le otorga a la herramienta un nivel de dimensión mayor, proporcionándole un medio para el almacenamiento de herramientas y/o archivos que el Botmaster puede necesitar durante sus tareas de administración. De esta manera mejoramos la versatilidad de la herramienta, siendo capaces de tener un gran repertorio de utilidades en la nube capaces de ser usadas desde nuestra herramienta desarrollada.

Concluimos que al ser un proyecto envuelto en un entorno educativo y universitario, entendemos que los usuarios finales de la herramienta desarrolla tiene únicamente fines destinamos a la administración de sistemas, y por con siguiente la herramienta no tiene cabida para ningún fin ilícito o uso irresponsable que se pueda hacer mediante el uso de la misma.

5.2 Trabajo Futuro

CLOUDBOTNET ha sido desarrollado en un lenguaje multiplataforma, además de seguir un diseño/arquitectura de código modular y programación orientada a objetos. CLOUDBOTNET está estructurado de forma sencilla y modular, lo cual facilita enormemente el desarrollo de nuevos módulos para CLOUDBOTNET, ofreciendo la oportunidad a investigadores y desarrolladores de modificar/añadir módulos a la herramienta basados en sus propios intereses. Por consiguiente, CLOUDBOTNET es un buen punto de partida para los administradores de sistemas que usen el protocolo SSH para la administración y configuración de los mismos.

CLOUDBOTNET puede ser fácilmente redimensionado en el futuro. La arquitectura de desarrollo empleada hace muy viable la posibilidad de ir añadiendo nuevas funcionalidades a la herramienta. Sólo se necesitan unos pocos conocimientos de programación y redes para ajustar CLOUDBOTNET a sus propios intereses. Por eso, desde el equipo oficial de desarrollo animamos a investigadores y desarrolladores a realizar su propia versión de CLOUDBOTNET.

En un futuro, para facilitar aún más esta posibilidad de aumentar las funcionalidades de CLOUDBOTNET, sería conveniente el desarrollo de un sistema de funciones/módulos separados del núcleo del código. De esta manera no sería necesaria la reprogramación del núcleo para la adición de nuevas funcionalidades.

6. CONCLUSIONS

6.1 Personal evaluation

The aim of our project was to demonstrate that through knowledge and research of minimum requirements in networking, programming and Cloud Computing, we were able to develop a versatile and robust tool that it can be used for system administration or creation of a Botnet. With these three subjects has led to a tool able to facilitate the management of multiple systems through a multi-session environment based on SSH protocol.

The use of Cloud Computing as a support of external storage awards a big level to the tool, providing a medium for storing tools and / or files that Botmaster may need during his administration tasks. Thus the versatility of the tool is improved, being able to have a large repertory of utilities in the cloud ready to being used from our developed tool.

We conclude that being a project involved in an educational and university environment, we understand that the end users of the developed tool has only purposes spend on system administration, this tool has no place for any illegal purpose or irresponsible use it can be done by using it.

6.2 Future Work

CLOUDBOTNET has been developed in a multiplatform language, following a modular design/architecture and object-oriented programming. CLOUDBOTNET is structured in a simple and modular way, which greatly facilitates the development of new modules for CLOUDBOTNET, offering the opportunity for researchers and developers to modify / add modules to the tool based on their own interests. Therefore, CLOUDBOTNET is a good starting point for system administrators using SSH protocol for configuring and managing systems.

CLOUDBOTNET can be easily resized in the future. Development architecture let the possibility of adding new features to the tool very viable. Only a few programming and network skills can be needed to adjust the tool to you own interests. Therefore, from the official development team we encourage researchers and developers to make their own version of CLOUDBOTNET.

In the future it would be convenient to develop a module system separates from core code. This way would not to be necessary reprograming the core in order to add new features to the tool.

7. BIBLIOGRAFÍA

Cloud Computing
http://0-proquest.safaribooksonline.com.cisne.sim.ucm.es/book/information-technology-and-software-development/9780124059320/index/idx001_html

http://cs.ecust.edu.cn/~yhq/course_files/cloud/Cloud%20Computing%20Bible.pdf

Fabric
https://www.digitalocean.com/community/tutorials/how-to-use-fabric-to-automate-administration-tasks-and-deployments
http://www.fabfile.org/

Google Cloud Platform
https://cloud.google.com/storage/docs/xml-api/gspythonlibrary

Google Cloud Platfom vs AWS
http://cloudacademy.com/blog/google-cloud-vs-aws-a-comparison/

Gsutil
https://cloud.google.com/storage/docs/gsutil

Python
https://www.python.org/doc/
http://www.cristalab.com/tutoriales/tutorial-basico-de-python-c90356l/

Botnets: Detection, Measurement, Disinfection & Defence
http://www.enisa.europa.eu/activities/Resilience-and-CIIP/critical-applications/botnets/botnets-measurement-detection-disinfection-and-defence/at_download/fullReport

http://www.kaspersky.es/
http://www.trendmicro.es/

Botnet detection techniques: review, future trends, and issues
http://link.springer.com/article/10.1631%2Fjzus.C1300242

Spamalytics: An Empirical Analysis of Spam Marketing Conversion
http://www.icsi.berkeley.edu/pubs/networking/2008-ccs-spamalytics.pdf

BotGraph: Large Scale Spamming Botnet Detection
http://research.microsoft.com/pubs/79413/botgraph.pdf

Spamcraft: An Inside Look At Spam Campaign Orchestration
http://cseweb.ucsd.edu/~savage/papers/LEET09.pdf

Detection of Spam Hosts and Spam Bots Using Network Flow Traffic Modeling
https://www.usenix.org/legacy/event/leet10/tech/full_papers/Ehrlich.pdf

NetFlow gives network managers a detailed view of application flows on the network
http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_case_study0900aecd80311fc2.pdf

Cisco IOS Flexible NetFlow
http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/flexible-netflow/product_data_sheet0900aecd804b590b.html

Using Machine Learning Techniques to Identify Botnet Traffic
http://www.ir.bbn.com/documents/articles/lcn-wns-06.pdf

Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis
http://www.eurecom.fr/en/publication/3886/download/rs-publi-3886.pdf

Detección y bloqueo de botnets mediante la combinación de técnicas basadas en el tráfico de red
http://www.issi.uned.es/Master_ISSI/WebMISSI/RepositorioTFM/2015/15S_MemoriaTFdM_ISW_TipoB_Juan_Enrique_Ripoll_Cervera.pdf

The Role of DNS in Botnet Command & Control
http://info.opendns.com/rs/opendns/images/OpenDNS_SecurityWhitepaper-DNSRoleInBotnets.pdf

Winning with DNS Failures: Strategies for Faster Botnet Detection
http://cesg.tamu.edu/wp-content/uploads/2012/04/reddy_papers/securecomm11.pdf

Inspecting DNS Flow Traffic for Purposes of Botnet Detection
http://geant3.archive.geant.net/Media_Centre/Media_Library/Media%20Library/gn3_jra2_t4_M4_deliverable.pdf

ANEXO I

En este anexo se van a detallar las diferentes contribuciones de cada participante.

Contribución Abdalah Fallaha Sabhan

Investigación:

- Botnet : Realizamos una investigación conjunta sobre los diferentes tipos de infraestructuras de Botnet que existen, leyendo varios artículos sobre ello para que posteriormente se decida optar por una infraestructura centralizada C&C (Comando y Control)de Botnet.

- Fabric : Después de que mi compañero tuvo la idea de hacer una investigación sobre la utilidad que podría tener la librería Fabric empezamos a investigar conjuntamente sobre las distintas forma que se utiliza la librería Fabric para la administración de sistemas y desligue de aplicaciones a través de protocolo SSH. De allí decidimos usar dicha librería de Python para poder dar lugar al comienzo de implementación de nuestra Botnet.

- Hosts de Botnet: Una vez decidida la infraestructura y la librería Fabric me puse a investigar la posibilidad de obtener hosts para añadirlos a nuestra Botnet. Encontré una página web que ofrece un servidor SSH gratuito así que cree una cuenta y se me asignó una dirección IP que nos sirvió como un Host para la Botnet.

- Cloud : Para la parte del Cloud investigué sobre los diferentes proveedores de servicio Cloud tanto AWS como Google Cloud Platform y al final se optó y se decidió conjuntamente por Google Cloud Platform ya que se encontró buena documentación para el uso de API de Google con Python , así como las funciones necesarias para poder interactuar con la parte de Google Storage a través de nuestra herramienta. Investigué sobre el protocolo que usa Google para la autorización de uso de las API, denominado OAuth.

Desarrollo e Infraestructuras:

Una vez elegido el lenguaje de programación con el que se va a desarrollar, la librería y la infraestructura de la Botnet empezamos a enfocarnos sobre el diseño y el desarrollo de la

aplicación. A continuación se detalla la contribución en las principales partes de desarrollo de la herramienta

- Interactive Menu: El menú interactivo está dividido por varias funciones según las opciones que tiene. La estructura principal del menú se implementó de manera conjunta.

- Command Interpreter: En este apartado investigué la posibilidad de realizar un intérprete de comandos en Python. Nuestra herramienta consta de varios comandos por lo que implementé los comandos de Download ,Role , add/del Host cada uno con sus diferentes métodos y clases

- Cloud: Me informé de las diferentes funciones del API de Google para desarrollar la parte interacción con Google Storage.

- Bases de datos: Investigué junto a mi compañero la manera de poder almacenar las IP's de los diferentes hosts de la Botnet en una base de datos.

- Roles: Dado que en el diseño principal decidimos incorporar una parte de roles a nuestra herramienta, desarrollé la parte de poder añadir y eliminar roles a los diferentes hosts

- Experimentación: la fase de experimentación se ha hecho de manera totalmente coordinada entre los dos integrantes del equipo. Cada vez que se ha ido implementando una funcionalidad nueva a nuestra herramienta se han ido haciendo los experimentos oportunos.

Memoria:

La memoria fue redactada de manera conjunta y equitativa.

Contribución José Luís Góngora Fernández

Investigación:

- Botnet: Después de la investigación que se realizó junto a mi compañero decidimos la infraestructura de nuestra Botnet. La infraestructura decidida fue una centralizada de comando y control.

- Fabric: Una vez definida la infraestructura me puse a investigar sobre los módulos de desarrollo necesarios para su implementación, por lo que después del estudio que realicé propuse la idea de utilizar la librería Fabric de Python. Una librería Python para la gestión de sistemas de forma paralela a través del protocolo SSH.
  A partir de este punto empezamos mi compañero y yo a informarnos de las diferentes maneras de implementación de la librería para su posterior integración en nuestro desarrollo.

- Hosts de Botnet: Participé en la investigación de conseguir hosts para que formen parte de nuestra red, ya que nuestra Botnet sería inútil sin que varios hosts estén conectados a ella.

- Cloud: Dado que el proveedor de Cloud escogido fue Google Cloud Platform. Investigué la manera de usar el API de Google para poder interactuar con la parte Storage de la plataforma de Google así como la forma de encajar el almacenamiento externo en nuestra Botnet. Al final de este estudio se decidió que en la parte Cloud se almacenaría las diferentes herramientas usadas por la Botnet para explotar los hosts.

Desarrollo e Infraestructuras:

Una vez decididos los requisitos de nuestro sistema, procedemos a las siguientes fases, el diseño y el desarrollo. A continuación se detalla la contribución en las principales partes de desarrollo de la herramienta

- Interactive Menu: El menú interactivo está dividido por varias funciones según las opciones que tiene. Se desarrolló de forma coordinada y conjunta siguiendo un diseño basando en programación modular.

- Command Interpreter: En este apartado investigué la integración de la programación orientada a objetos para desarrollar un intérprete de comandos. Así como implementé los comandos de Status , runCommand , OpenShell , cada uno de ellos con sus diferentes clases y métodos

- Cloud: Preparé el entorno Cloud con las diferentes herramientas y desarrollé la parte de Gsutil.

- Bases de datos: Desarrollé la parte de alimentar la Botnet con los diferentes Hosts. La información de los Hosts se almacenan en una base de datos SQlite

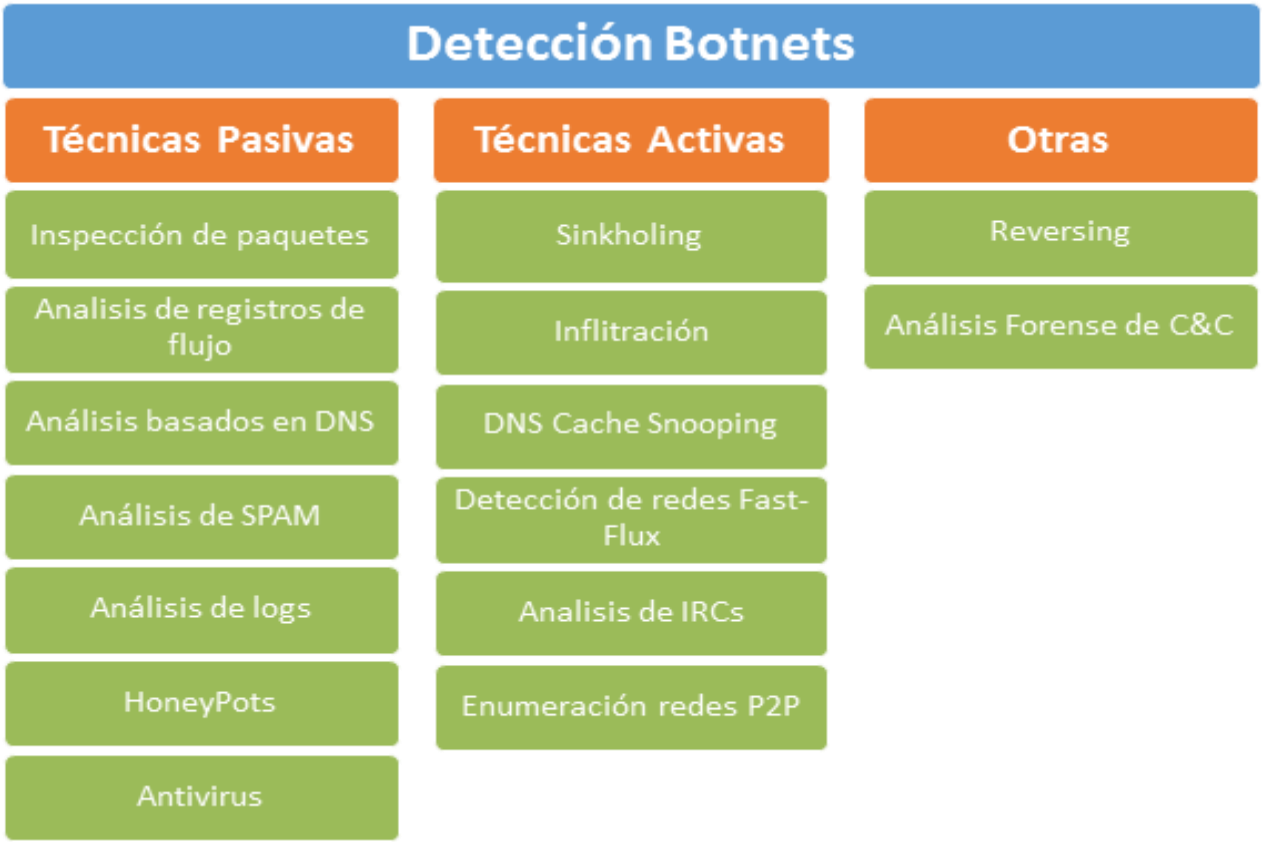
- Roles: Investigué la manera de usar la función Roles de la librería Fabric y participé en su desarrollo en nuestra herramienta.

- Experimentación: la fase de experimentación se ha hecho de manera totalmente coordinada entre los dos integrantes del equipo. Cada vez que se ha ido implementando una funcionalidad nueva a nuestra herramienta se han ido haciendo los experimentos oportunos.

Memoria:

La memoria fue redactada de manera conjunta y equitativa.

ANEXO II
MEDIDAS Y TÉCNICAS DE DETECCIÓN



TÉCNICAS PASIVAS

Las técnicas pasivas de mediación y detección se basan únicamente en la observación y análisis de datos como herramienta para detectar que nuestro sistema puede estar comprometido. A través de la supervisión, las actividades del sistema siguen su cauce normal puesto que no hay ninguna técnica invasiva para la medición o detección, esto también nos hace invisible al Botmaster, que no podrá saber que estamos realizando tareas de investigación. Hay que tener en cuenta que al usar solo técnicas pasivas los resultados de mediación y detección se verán limitados en cuanto al análisis final.

Inspección de paquetes

Esta técnica se basa en la inspección del tráfico de red. Es una técnica que proporciona mucha información, por lo que necesitaremos de experiencia y buenos filtros en nuestro analizador de tráfico favorito. Buscaremos protocolos que no tenemos constancia de que están activos, Payloads y Shellcodes en paquetes sospechosos, conexiones a internet

sospechosas con malware, etc. Toda esta información es de gran utilidad a la hora de medir y detectar si nuestro sistema está comprometido.

La dificultad de esta técnica viene dada por la dimensión y la cantidad de paquetes que la red puede tener. A mayor número de paquetes mayor complejidad y dificultad. Por eso es una técnica donde prime nuestra experiencia analizando tráfico y usando nuestro analizador de red.

Existen dos tipos de análisis de paquetes:

Stateful Packet Inspection o SPI: se centra exclusivamente en el los datos de cabecera (IPs, puertos, protocolos,…) pero no entra a analizar el contenido del paquete. Este tipo de análisis SPI es el que realizan los normalmente los firewalls.

Deep Packet Inspection o 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 basadas en firmas, análisis estadísticos y análisis de anomalías para detectar posibles amenazas en la red monitorizada. Con una parametrización adecuada, estos sistemas pueden detectar tanto amenazas conocidas e identificadas como amenazas nuevas al detectar patrones o tráfico anómalo.

Los tipos de sistemas o aplicaciones existentes enfocados en realizar análisis DPI pueden ser clasificados en IDS (Intrusion Detection Systems) e IPS (Intrusion Prevention Systems). Los IDS son sistemas pasivos que solo generan alertas, mientras que los IPS toman medidas activas como cortar las conexiones sospechosas.

Logotipos de los IDS SNORT, SURICATA y BRO

Análisis de flujos de red

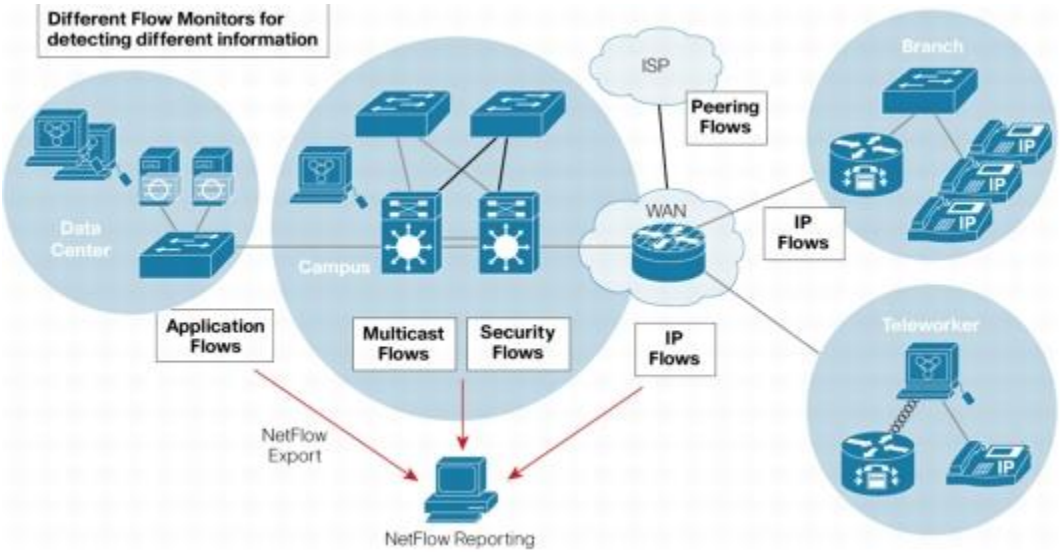
Análisis de flujo es una técnica para tracear tráfico red con un nivel de abstracción mayor al de inspección de paquetes. En esta técnica no nos centramos en analizar paquete por paquete. La técnica se centra en analizar un flujo de paquetes que componen un flujo

completo, también conocido como Stream. Típicos atributos son: direcciones origen y destino, puerto, protocolo, duración de la sesión, número de paquetes que componen el stream. 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.
El objetivo del análisis de flujo es identificar patrones de tráfico que pueden ser usados para separar tráfico malicioso de tráfico normal, con lo que se crea un esquema para detectar tráfico potencialmente malicioso.

Por ejemplo, imaginemos que queremos detectar una Botnet en nuestro sistema que trabaja sobre el protocolo IRC. La primera etapa consistiría en separar el flujo de paquetes perteneciente al tráfico mediante protocolo IRC. La siguiente etapa sería separar de todos los flujos de IRC cuáles de ellos son potencialmente maliciosos, mediante el análisis del flujo completo.

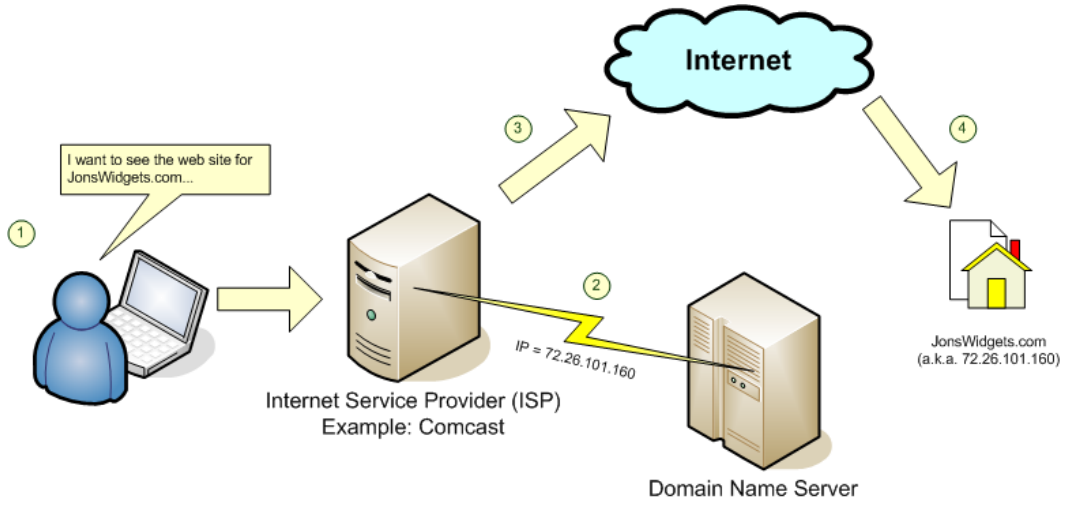
Routers y switches pueden ser configurados para agregar tráfico viajando a través de NetFlow. Los routers y switches que soportan este protocolo, además de distribuir el tráfico por la red, se encargan de enviar los metadatos de dicho tráfico (IPs, puertos,…) a un servidor especializado de recolección, donde se realizan los análisis correspondientes con el fin de identificar patrones de tráfico anómalos, que indiquen algún comportamiento extraño.

Modelo de datos NetFlow 5



Arquitectura NetFlow

Enfoques basados en DNS



Funcionamiento de un servidor DNS

Primero vamos a explicar porque algunas Botnets suelen hacer uso de nombres de dominio (DNS). Para establecer la conexión Bot-c&c, el desarrollador de la Botnet tiene principalmente dos opciones: el uso de IP fijas donde se encuentra el C&C, o el uso de nombres de dominio.

El uso de nombres de dominio tiene una mayor robustez que las IP fijas, debido a que un nombre dominio puede asociar distintas IP's, dando lugar a una mayor dificultad para la

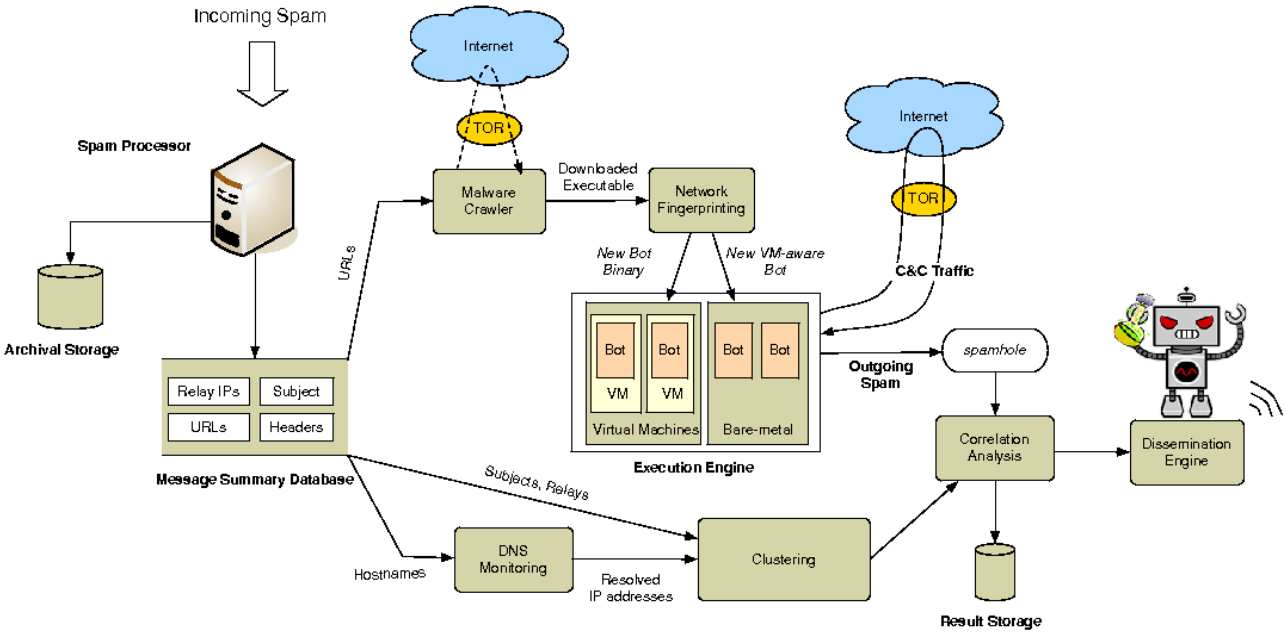
detección de la Botnet y el Botmaster.

La técnica Fast-Flux, a través de la cual un dominio resuelve distinta IP dependiendo del momento en el que se realice la petición, lo que consigue que se descentralizan los servidores C&C y dificulta la labor para la investigación de la propia Botnet y la captura del Botmaster. Al cambiar la IP de destino de las conexiones de los Bots se dificulta la detección de comportamientos anómalos por parte de sistemas de análisis de flujos de red.
Por otro lado, el uso de IPs fijas por parte del malware puede evitar su detección mediante el uso de técnicas basadas en DNS. Sin embargo, al usar siempre las mismas IPs en sus comunicaciones facilitan su detección por técnicas como el análisis de flujo, que posteriormente pueden ser bloqueadas fácilmente mediante reglas de red.

Algunas técnicas pasivas para la detección de una Botnet mediante el uso de DNS son:

Análisis estadístico de las peticiones fallidas de resolución de DNS
Monitorización de dominios maliciosos
Dominios con bajos TTLs
Detección de trafico DNS anormal

Análisis de SPAM logs



Sistema de detección Botnet spam

Una de las principales opciones que suelen tener una Botnet es el envió masivo de emails no solicitados (spam) desde las máquinas comprometidas. Obviamente esta técnica solo

sirve para detectar una Botnet que hagan spamming.

Una característica del spam que realizan las Botnets es que los mensajes suelen seguir un patrón o incluso ser muy parecidos. Debido a estas semejanzas, el análisis estadístico de los diferentes datos del correo, tanto la cabecera como el propio mensaje, en una comunicación SMTP puede ser un indicador para determinar una determinada red está infectada.

El uso de herramientas como los spamtraps hace que la detección de correos de spam sea más efectiva. El propósito de los spamtraps es la recepción de email no solicitado mediante la creación de buzones de correo electrónico. Para que sean efectivos estos emails deben ser publicitados y registrados en múltiples sitios como foros, listas de noticias, etc.

Análisis de ficheros Logs

Las aplicaciones modernas y sistemas informáticos suelen llevar un registro de toda la actividad que realizan. Esto supone la creación de unos ficheros denominados logs, los cuales sirven para archivar y guardar la información que se registra. Esto hace que estos ficheros logs son una buena partida para investigar y recaudar información acerca de lo que el sistema, controlados, firewalls, servicios de red, y dispositivos están haciendo. El estudio de estos ficheros puede dar lugar a descubrir comportamientos inadecuados, el cual puede derivar en que nuestra red está infectada.
Por ejemplo, si examinamos el fichero de logs de las direcciones webs a las que hacemos peticiones y vemos que una cierta IP está siendo muy solicitada, es posible que estimemos siendo participes de una ataque de denegación de servicios (Ddos).

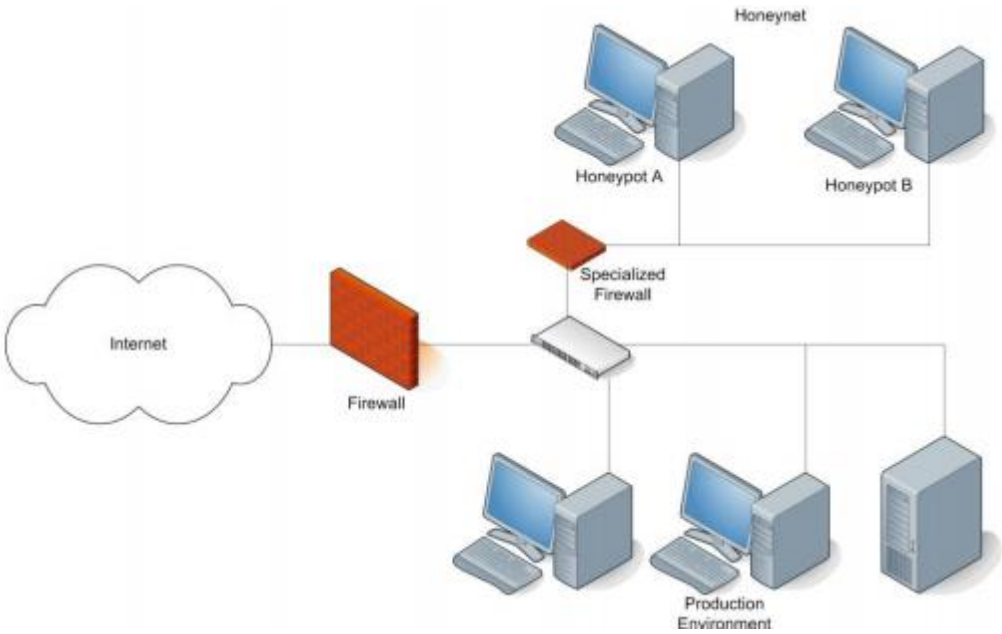
HoneyPots

Un hoynetpot es un entorno vulnerable preparado adrede, que se mantiene a la espera de ataques para registrarlos, y para que posteriormente los investigadores puedan analizarlos. El procedimiento consiste en mantener una debilidad o vulnerabilidad en un programa, en el sistema operativo, en el protocolo, o en cualquier otro elemento del equipo susceptible de ser atacado, que motive al atacante a usarlo.

Gracias al uso de honeypots y al análisis de la información que estos generan podemos ver la magnitud del problema al que nos enfrentamos, obteniendo datos valiosos acerca de:

Tendencias de ataques.
Vulnerabilidades explotadas.
Servicios que se pretenden vulnerar.
Países más activos en ciberataques
Muestras de malware no identificadas por motores antivirus.
Técnicas usadas por los atacantes.
Distribuidores de malware.

Equipos pertenecientes a Botnets.
Centros de comando y control (C&C).



Red de ejemplo con Honeypot

Honeypots pueden ser usados para medinar ataques provenientes de Botnets, que puede dar lugar a la detección de sistemas infectados.

El Honeypot puede estar diseñado con múltiples objetivos:

Honeypots de baja interacción, los cuales tienes un cierto número de servicios corriendo, para simplemente alertar de la existencia del ataque u obtener información sin interferir en el mismo, usados fundamentalmente como medida de seguridad.

Honeypots de alta interacción para tratar de ralentizar el ataque (sticky honeypots) y proteger así el resto del sistema, usados para reunir mucha más información y con fines como la investigación. La mayoría de las veces tienen componentes ocultos para la monitorización del sistema.

Si el sistema dispuesto para ser atacado forma toda una red de herramientas y computadoras dedicadas en exclusiva a la tarea de monitorización y análisis se le denomina honeynet.

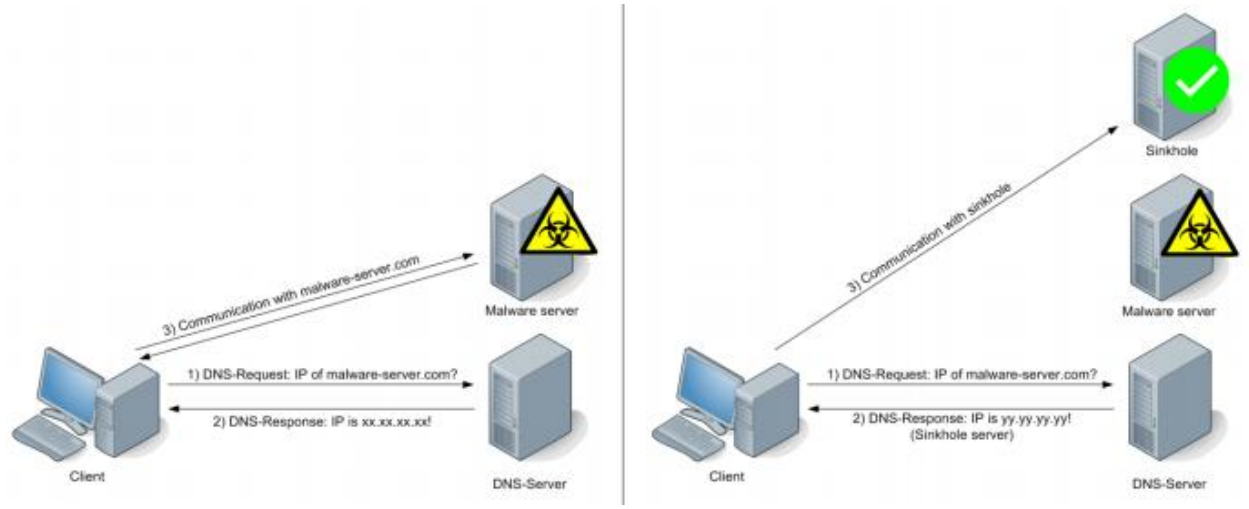
TECNICAS ACTIVAS

El grupo de técnicas activas contiene enfoques que implican la interacción con las fuentes de información que están siendo monitorizadas. Aunque este tipo de técnicas permite la realización de medidas más profundas, su aplicación puede dejar huellas que influyen en los resultados, o que incluyen actividades que pueden ser observados por el Botmaster. Esto puede causar reacciones, tales como un ataque DDoS contra el analista o la introducción de cambios en la estructura de la Botnet que complicarán las mediciones, incluso la migración del servicio para eludir la vigilancia.

Sinkholding

En general el término Sinkholding describe una contramedida técnica para cortar una fuente de control maliciosa del resto de la Botnet. Esta técnica se realiza en contra de una variedad de objetivos, los más destacados son contra servidores de comando y control de Botnets o troyanos Dropzone. Una de las variantes más comunes de esta técnica es cambiar constantemente el nombre de dominio del objetivo malicioso para que apunte a una máquina controlada por una parte de confianza como se muestra en la figura. Un efecto similar puede lograrse cambiando el encaminamiento del tráfico de una dirección IP estática de la misma manera.

El principio básico de esta medida se basa en el hecho de que el dominio es muy probable de ser contactado por las máquinas infectadas tratando de alcanzar su servidor -mando y control-. Como los nombres de dominio y direcciones IP utilizadas para tales fines a menudo sólo se utilizan con fines maliciosos, este enfoque tiende a tener una baja tasa de falsos positivos.



Técnica sinkholding

Un inconveniente de esta técnica es que la exactitud de medición depende en gran medida de la información disponible para el host de destino. Si los Bots de contacto con el servidor DNS proporcionan información que pueda conducir a una identificación única, la exactitud de la precisión debería ser bastante alta. Por otro lado, si la información proporcionada por los hosts es escasa, los resultados de medición se verán influenciados por una variación significativa. Por ejemplo, cuando todas las cargas útiles de paquetes entrantes están totalmente cifradas, sólo la dirección IP (y otros detalles de las cabeceras de paquetes) se pueden usar para la identificación.

La medición a través de direcciones IP únicas implica varios aspectos que pueden afectar a los resultados como se explica en el siguiente ejemplo. En una parte si tenemos varias maquinas infectadas que están conectadas a un único Router que tiene una IP estática y está fuera de la red LAN ,estas múltiples infecciones se puede tener en cuenta de una vez. Por otra parte si tenemos varias máquinas con direcciones IP dinámica esto nos llevaría a resultados estadísticos. Este caso es precisamente lo que sucede cuando acceden a Internet los dispositivos móviles y sus direcciones son traducidas a través de una puerta de enlace central o cuando un usuario se conecta y desconecta obteniendo una nueva dirección IP cada vez.

Infiltración

Las infiltración en las Botnets se puede dividir en técnicas basadas en Hardware y técnicas basadas en Software. En la primera se cubre la investigación sobre el Bot ejecutable y el resultado del tráfico monitorizado para lograr medidas de control y dirección. Este último se puede aplicar si el acceso al servidor de comando y control es posible. Esto incluye máquinas físicas, así como máquinas virtuales que podrían estar ejecutándose en un centro de datos.

Esta técnica requiere en su punta de partida el uso de ingeniería inversa en los mecanismos de comunicación utilizados por la Botnet. Un análisis tan preciso puede conducir a la identificación de las debilidades potenciales. Este procedimiento puede ser comparado con una auditoría de seguridad o un test de penetración de la Botnet y su infraestructura. El conocimiento obtenido en el proceso puede ser explotado en nuevas medidas para lograr una posición dominante dentro de la Botnet. Esto puede dar lugar a la posibilidad de realizar mediciones o revelar información sobre los hosts infectados, o incluso el Botmaster.

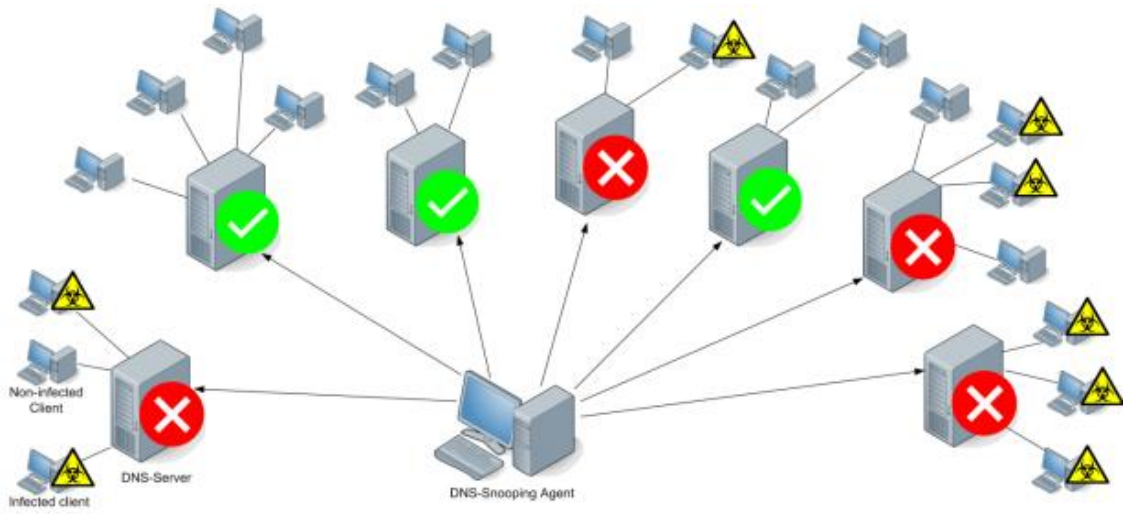
El otro enfoque, la infiltración basada en hardware, se puede aplicar si una dirección IP perteneciente a un servidor de comando y control ha sido identificada y se puede establecer una relación con un centro de procesamiento de datos o con la empresa de alojamiento. Con la obtención de una conexión a un puerto de los servidores sospechosos, la comunicación podría ser interceptada y analizada. Esto permite que todo el tráfico que

pasa por el servidor sea supervisado, además de obtener información relevante sobre las máquinas infectadas como el número de máquinas y su ubicación entre otros.

DNS Cache Snooping

La técnica de medición llamada DNS Cache Snooping se basa en la propiedad de almacenamiento en caché implementado y utilizado por muchos servidores DNS. Si un servidor DNS es consultado por un dominio del cual no tiene entrada emitirá una consulta hacia el responsable del servidor de nombres y almacena el dato resultante en una caché local. El almacenamiento en caché se utiliza principalmente para aumentar el rendimiento de un servidor de nombres y reducir su carga de tráfico.

Esta funcionalidad puede ser usada para fines de medición. La idea principal es comprobar si un dominio objetivo ha sido consultado a través de un servidor DNS específico. Esto se puede averiguar comprobando si existe alguna respuesta almacenada en la caché, como se muestra en la figura.



DNS Cache Snooping

Existe dos variantes de esta técnica, elegir una depende de la configuración del servidor DNS. En la primera variante, una consulta es enviada al servidor DNS con una variable especial, el flag no-recursivo activado, esto prohíbe al servidor reenviar la consulta a los servidores de nombres responsables. El comportamiento del servidor puede variar,

dependiendo si el servidor consultado originalmente ha almacenado en caché una respuesta para el nombre del dominio objetivo. El servidor enviará directamente o bien una respuesta a la consulta incluyendo la dirección de la IP que ha resuelto o si el servidor no puede responder enviará una respuesta indicando los servidores que contactará más tarde. Este método no funcionaría con todos los servidores DNS, ya que muchos servidores simplemente rechazan este tipo de consultas con el fin de protegerse contra los ataques de denegación de servicio.

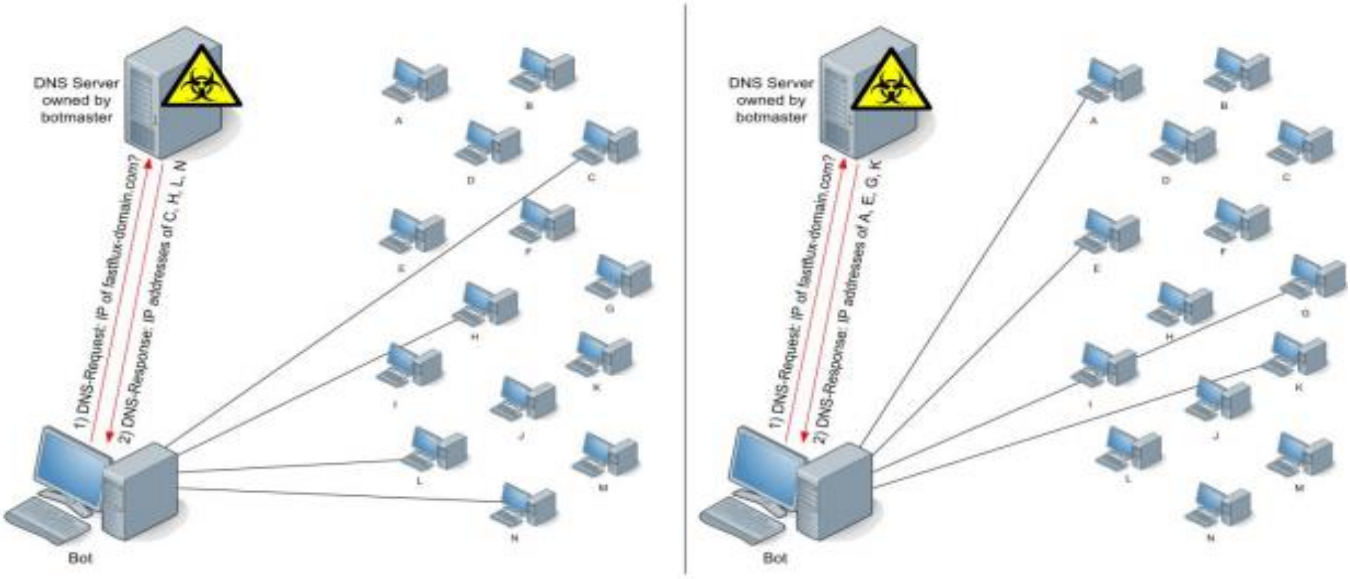
La segunda variante funciona con cualquier servidor DNS. En este caso la variable mencionada antes no se activa y el servidor puede reenviar la consulta a otros servidores de nombres. Esto hace que sea posible determinar si una respuesta ha sido almacenada o no en caché analizando cuándo ha sido consultado por última vez el dominio. Esto se puede lograr mediante la evaluación del valor de la TTL(Time to Live) incluida en la respuesta. Si tiene un valor por defecto del servidor entonces no se ha almacenado en la caché ninguna información antes que la consulta. Si es menor que el valor predeterminado entonces el dominio ya habría sido consultado anteriormente porque el contador TTL para dicho dominio ya habría empezado con una consulta antes que la consulta de espionaje. Esta técnica es posible porque el valor de TTL no se actualiza para un nombre de dominio guardado en la caché.

Rastreo de Redes Fast-Flux

Algunas Botnets usan redes Fast-Flux. Las redes Fast-Flux cambian los registros DNS rápidamente, apuntando a un gran número de máquinas, actuando como una capa adicional para ocultar los sistemas de distribución de contenidos reales.

Mediante la asociación de un gran volumen de direcciones IP con sólo unos pocos, o incluso un solo nombre de dominio, la red se vuelve mucho más robusta frente a las contramedidas. Sin embargo las características especiales que representan los registros DNS que sirven este tipo de redes les distingue de otras redes. Los registros de dominios que están conectados a redes Fast-Flux suelen tener un periodo de validez corto de tan sólo unos minutos. Esto se indica por el valor de tiempo de vida (TTL) que se incluye en la respuesta generada por el servidor DNS, que está bajo control del Botmaster.
Un servidor DNS malicioso es generalmente operado por el Botmaster o instalado como servicio en un servidor comprometido. Una vez transcurrido este período (TTL), una nueva consulta dará lugar a un conjunto diferente de direcciones IP asociadas con poca o ninguna similitud o relación topológica entre sí. Una característica que se puede observar es la variedad de direcciones IP devuelta para un dominio fast-flux. Estas direcciones IP suelen proceder de varias redes y ISP (Proveedores de servicio). Las redes que no están infectadas como un corto periodo de TTL, por ejemplo páginas web como google.com o facebook.com,

generalmente devuelven direcciones IP que tienen una gran similitud, lo que indica que nacen desde una misma red y están relacionadas entre sí.



Técnica Fast-Flux

Monitorizando dominios cuyas respuestas DNS cuentan con un TTL bajo no solo permite la identificación de dominios de flujo rápido si no que también mediante la emisión de consultas DNS repetidas, máquinas que forman parte de la red se pueden extraer y coleccionar de estos registros. Como se muestra en la figura.

Medición y detección de Botnet basadas en IRC

Hoy en día, Internet Relay Chat (IRC) todavía sirve como infraestructura comando y control(C & C) para la administración de Botnets. Según el informe anual del año 2010 de Symantec, 31% de todos los servidores C&C identificados en 2009 utilizaron IRC como su protocolo de comunicación. IRC es un protocolo de chat ligero y robusto, que ofrece una gran cantidad de funcionalidades adecuadas para controlar grandes Botnets. Un diseño de comando y control basado en IRC implica uno o más canales de IRC ya sea en redes IRC públicas o servidores donde los Bots informan su presencia y reciben órdenes.
Por lo general, los Bots recién lanzados, en su primera acción se conectan a un canal y esperan instrucciones.

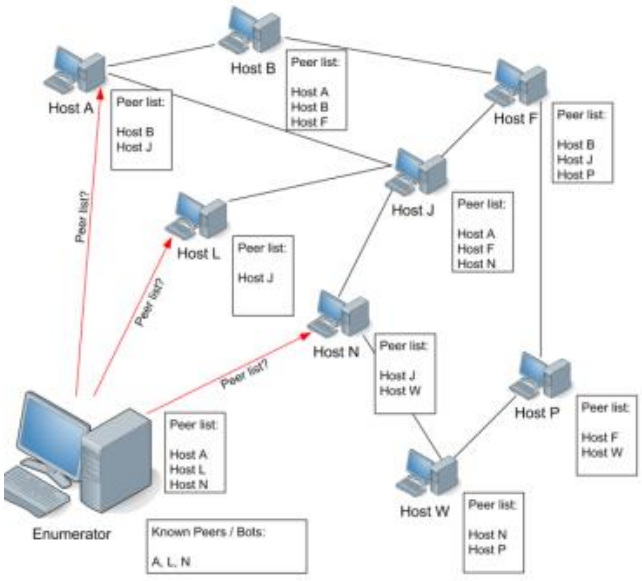
Con el fin de medir los Bots que utilizan el modelo de comunicación basado en IRC, primero es necesario obtener la información necesaria para unirse y participar en el canal de Botnet. Información básica de conexión, que implica la dirección IP y número de puerto del servidor IRC, así como el canal utilizado para el control la Botnet, toda esta información se puede extraer de muestras de malware de la Botnet que se trate. Dependiendo de la complejidad del mecanismo de comunicación de la Botnet, el nombre de usuario o los

credenciales de autenticación, y también la funcionalidad de cifrado, pueden ser extraídos y entendidos.

Como segundo paso, los canales IRC pueden ser unidos así como se pueden recoger datos. La calidad y la cantidad de información que se puede obtener dependen en gran medida de las precauciones tomadas por el Botmaster. Si el canal de control está alojado en un servidor IRC estándar (o incluso público), puede ser posible simplemente registrar y realizar un seguimiento de los nombres de usuario en el canal sin mucho esfuerzo.

Enumeración de redes Peer-to-Peer

Un enfoque común utilizado por las Botnets es emplear una infraestructura basada en redes P2P. La idea básica es crear una red superpuesta con su propio esquema de direccionamiento y de protocolo para encaminar mensajes entre los participantes solamente.
A pesar de que la información sobre la Botnet no está disponible en un punto central, la estructura de la red puede ser explotada para fines de medición. Haciendo consultas repetitivas por las listas de los nodos vecinos es posible enumerar la Botnet recursivamente. El resultado sería una lista exhaustiva de todos los nodos activos que forman parte de la red peer-to-peer.



Enumeración P2P

Esto se logra generalmente mediante ingeniería inversa, el protocolo de comunicación y creando una implementación del Bot con el fin de realizar la tarea de enumeración.