ROOTKITS
ESCONDIÉNDOSE DEL ADMINISTRADOR

-BEATRIZ SÁNCHEZ

- RODRIGO FRANCISCO

¿QUÉ ES UN ROOTKIT?

Un **rootkit** es un programa o un conjunto de programas que adquieren y mantienen acceso privilegiado al sistema operativo mientras que activamente están ocultando su presencia. El término *rootkit* es la unión de los palabras "root" y "kit" ya que originalmente era una colección de herramientas que permitían el acceso de nivel administrador a una computadora o red. Por su parte *root* hace referencia al usuario administrador en los sistemas Unix y *kit* se refiere a los componentes de software que implementan la herramienta. Hoy en día los rootkits simplemente se asocian con malwares (para cualquier sistema operativo) que encubren su existencia y sus acciones de los usuarios y de otros procesos el sistema.
En la definición de "rootkit", la palabra clave es "indetectable". La mayor parte de la tecnología y trucos empleados por un rootkit está diseñado para ocultar código y datos en un sistema. Por ejemplo, muchos rootkits pueden ocultar archivos y directorios. Otras características en un rootkit son generalmente para acceso remoto y escuchas ilegales, por ejemplo, para oler paquetes de la red. Cuando se combinan, estas características brindan un golpe de gracia a la seguridad.La amenaza surge cuando los programas malignos hacen uso de esta técnica y logran ocultar su presencia en el sistema durante un período de tiempo prolongado, generalmente mayor que cualquier programa maligno de los conocidos hasta el presente.

MALWARE Y ROOTKIT NO SON LO MISMO

MALWARE:
También llamado “software de actividades ilegales es una categoría de código malicioso que incluye virus, gusanos y caballos de Troya” , además se refiere a todo aquel software cuyo objetivo está en corromper la estructura del sistema operativo , así como recolectar información personal de usuarios de manera ilegítima, hasta el empleo de recursos de forma remota. Es importante precisar que si bien un virus informático es un software malintencionado que se asocia a otros programas computacionales o archivos informáticos para poder ejecutarse, por lo general sin el conocimiento o permiso del usuario, cuando nos referimos al software malicioso, aludimos a un cúmulo de amenazas que afectan a los sistemas de cómputo, es decir, puede tratarse de un virus, un caballo de Troya, una puerta trasera (backdoor), un programa espía (spyware), hasta un devastador gusano que puede echar abajo toda una infraestructura de red.
Es importante remarcar que un Rootkit no es un Malware en sí mismo pero, debido a que es utilizado ampliamente para ocultar los mismos, muchas veces se lo considera incorrectamente como programa dañino.
El enfoque de muchos rootkits recientes ha sido cooperar con malware con el fin de ocultar la funcionalidad de control y comando remoto del malware. El malware requiere acceso remoto a

las estaciones de trabajo infectadas, y los rootkits proporcionan el sigilo para permitir que el malware se ejecute sin ser detectado, es esto el cómo se relaciona un rootkit con el malware.

A diferencia de otros programas como exploits o malware, los rootkits generalmente continuarán funcionando incluso si el sistema tiene ha sido reiniciado. ¿Por qué es importante esta definición? Esta definición establece varias diferencias clave entre los rootkits y otros tipos de software como troyanos, virus o aplicaciones. Por ejemplo, eliminar la palabra no detectada de la definición cambiaría la definición a la de un paquete de software de administración del sistema o software de administración remota. Sin embargo, el hecho de que el software no se detecte y proporcione una conexión constante al sistema implica que el software proporciona una puerta trasera para facilitar el acceso en el futuro. Los rootkits también se escriben a propósito para que no se detecten a través de métodos tradicionales o aceptados dentro de la industria de la seguridad.

FUNCIONES DE UN ROOTKIT

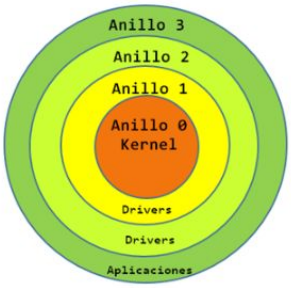
El rootkit tiene 3 objetivos principales

- **Ejecutarse**. Un rootkit desea poder ejecutarse sin restricciones en un equipo de destino. La mayoría de los sistemas informáticos (incluido Windows) tienen mecanismos tales como listas de control de acceso (ACL) para evitar que una aplicación tenga acceso a recursos protegidos. Los rootkits aprovechan las vulnerabilidades de estos mecanismos o utilizan los ataques de ingeniería social para instalarse, de modo que no tengan restricciones sobre lo que pueden hacer.
- **Esconderse.** El rootkit debe permanecer invisible a otras aplicaciones para evitar ser desinstalado por software de seguridad.
- **Actuar.** un autor de rootkit desea obtener algo de la computadora comprometida, como robar contraseñas o ancho de banda de la red, o instalar otro software malicioso.

TIPOS DE ROOTKITS

Espacio de usuario y espacio de kernel

En general un sistema operativo cuenta con dos áreas de memoria bien diferenciada: el espacio de usuario y el espacio de kernel. En una arquitectura típica, estos espacios se ubican en "anillos de seguridad" con nivel de privilegios distintos.



En el espacio de usuario correspondiente al anillo menos privilegiado (anillo3) se ejecutan las aplicaciones en un entorno controlado, que no tiene acceso directo a los recursos (memoria, disco, dispositivos, etc.) si no que debe solicitarlo a través de llamadas al sistema. En el espacio de kernel (anillo 0, de máximo privilegio) se gestiona y se tiene acceso total a cualquier recurso, siendo donde se ejecuta el núcleo del sistema operativo. Los anillos intermedios no son de implementación frecuente y en la mayoría de sistemas solo tendremos que contar con el espacio de usuario y el espacio de kernel.

Teniendo en cuenta este punto los rootkits pueden contener fácilmente componentes de modo usuario y modo kernel. La parte de modo de usuario se ocupa de la mayoría de las funciones, como la conexión en red y el control remoto, y la parte de modo de kernel se ocupa de la ocultación y el acceso al hardware.

Rootkits en espacio de usuario

Una definición precisa para un rootkit en espacio de usuario podría ser: un conjunto de colección de programas y código capaz de no ser detectado y que reside un un espacio de aplicación no perteneciente al kernel, permitiendo la presencia constante en un ordenador o sistema de información. Todas las aplicaciones de modo de usuario se ejecutan en el nivel de privilegio de la cuenta del usuario dentro del sistema y no como parte del sistema operativo.
Este tipo de rootkit habitualmente sustituye ejecutables legítimos del sistema por otros modificados, de modo que la información que proporcionan esté manipulada según interese.

Entre los principales binarios objetivo de un rootkit para conseguir su ocultación y operación se encuentran los siguientes:

- Ficheros: ls, df, stat, du, find, lsof, lsattr, chattr, sync…
- Conexiones: ip, route, neststat, lsof, nc, iptables, arp…
- Procesos: ps, top, pidof, kill, lsof…
- Tareas: crontab, at…
- Logs: syslogd, rsyslogd…
- Accesos: sshd, login, telnetd, inetd, passwd, last, lastlog,su, sudo, who, w, runlevel…

Hoy en día los rootkits de modo usuario no son muy efectivos y se detectan relativamente fácil con la mayoría de los productos antivirus. Incluso se llega a argumentar que los rootkits de este tipo son inútiles, sin embargo, muchos de los malwares todavía emplean técnicas de rootkit de modo usuario por lo cual analizarlos es importante para su posterior detección.

El primer paso para cualquier rootkit del modo usuario es inyectar su código en el proceso donde quiere instalar su “hooking” o gancho, el cual es el mecanismo utilizado por los rootkits para llevar a cabo su tarea. Aunque hay varios métodos y técnicas para enganchar procesos, los rootkits utilizan dos en especial: la tabla de direcciones y la función en línea.

Rootkits en espacio de kernel

Los rootkits de modo kernel son simplemente binarios maliciosos que se ejecutan en el nivel de privilegio más alto disponible en la CPU que es implementado por el sistema operativo (es decir, Ring 0).

El objetivo más interesante para un atacante tras acceder a un sistema y conseguir privilegios de superusuario es instalar un rootkit en modo kernel y, de este modo asegurarse un control total del sistema a la par que una mejor ocultación. Una vez integrado en el sistema, su detección es mucho más complicada pues se mueven en un nivel de privilegio superior que les brinda permisos para poder modificar, ya no sólo los binarios en sí, sino las propias funciones y llamadas del sistema operativo.

En este caso el diseño del rootkit debe ser mucho más elaborado, puesto que al trabajar en espacio de kernel e interaccionar con funciones y llamadas del sistema básicas, cualquier defecto en su funcionamiento puede provocar un fallo en el kernel con consecuencias fatales.

La mayoría de los rootkits del modo kernel tienen algunos atributos definitorios que tienden a dificultar su captura y eliminación.

• Sigilo: dado que obtener acceso al modo kernel puede ser difícil, normalmente el autor es lo suficientemente inteligente como para hacerlo con sigilo. Además, dado que muchos antivirus, detección de intrusión de host (HIDS), sistemas de prevención de intrusión de host (HIPS) y productos de firewall observan de cerca el modo kernel, el rootkit debe tener cuidado de no activar alarmas o dejar huellas obvias.
• Persistencia Uno de los objetivos generales de escribir un rootkit es obtener una presencia persistente en el sistema. De lo contrario, no hay necesidad de pasar por la molestia de escribir un controlador de kernel. Por lo tanto, los rootkits de modo kernel generalmente están bien pensados e incluyen alguna característica o conjunto de características que aseguran que el rootkit sobreviva al reinicio e incluso al descubrimiento y limpieza mediante la replicación de su punto de apoyo mediante múltiples técnicas.
• Los rootkits en modo Kernel de gravedad utilizan técnicas avanzadas para violar la integridad de la computadora de un usuario en el nivel del sistema operativo. Esto no solo es perjudicial para la estabilidad del sistema (el usuario puede experimentar choques frecuentes o impactos en el rendimiento), sino que también es mucho más difícil eliminar la infección y restaurar el funcionamiento normal del sistema.

Los rootkits en el espacio del kernel también utilizan el mecanismo de gancho o “hooking”, en los últimos años se han documentado varias técnicas en la comunidad de rootkit. Algunos de los métodos generales que son comúnmente los más utilizados son:

Enganche a SSDT (Tabla de despacho de servicio del sistema)

La tabla de distribución del servicio del sistema es una tabla que contiene punteros a las funciones de servicio (API).

La creación de un gancho en esta tabla consiste en reemplazar el valor del puntero original de una entrada (tomemos NtOpenProcess para el ejemplo) por la dirección de una función con el mismo prototipo en cualquier módulo cargado en modo kernel. Por lo general, el desvío de una API solo se realiza para filtrar los parámetros de entrada (y denegar el acceso si es necesario) y devolver el valor del puntero original al final del procesamiento , para llamar a la función original.

Enganche a IDT (Tabla de envío de interrupciones)

Las interrupciones son un concepto fundamental para las transacciones de E / S en los sistemas operativos. La mayoría del hardware está controlado por interrupciones, lo que significa que envía una señal al procesador denominada solicitud de interrupción cuando necesita reparación. El objetivo de conectar el IDT es enganchar cualquier función que ya esté registrada para una interrupción determinada. Un ejemplo es un keylogger de bajo nivel. Al

reemplazar la rutina de servicio de interrupción que se almacena en el IDT para el teclado, un rootkit podría detectar y grabar pulsaciones de teclas.

Enganche a IAT (Tabla de direcciones importantes)

Esta técnica es bastante simple y se usa ampliamente en programación, tanto infame como benigna. Cuando se carga un ejecutable, Windows lee la estructura de Portable Executable (PE) ubicada dentro del archivo y carga el ejecutable en la memoria. El formato PE, un formato de archivo Unix modificado, es el nombre del formato de archivo de todos los archivos EXE, DLL, SYS y OBJ dentro de Windows y es fundamental para la arquitectura de Windows. El ejecutable enumerará todas las funciones que requiere de cada DLL. Esta técnica permite que el rendimiento en tiempo de ejecución sea rápido, mientras que la carga inicial de un ejecutable puede ser más lenta. Todo lo que debe hacer ahora una DLL de rootkit es cambiar la dirección de una función específica en el IAT, por lo que cuando la aplicación llama a la función específica, se llama a la función del rootkit.

Otras variaciones de un rootkit.

Bootkits

Los kits de arranque van un paso más allá, al agregar funciones de arranque a los rootkits y afectar el firmware de los sistemas y los sectores de arranque de los discos. Usualmente este tipo de rootkits buscan afectar al Master Boot Record, un pequeño programa que se ejecuta cuando una computadora arranca. Normalmente, el MBR reside en el primer sector del disco duro. El programa comienza el proceso de arranque buscando la tabla de particiones para determinar qué partición usar para el arranque. Luego transfiere el control del programa al sector de inicio de esa partición, que continúa el proceso de inicio.
De esta manera, la dificultad intrínseca de detectarlos se ve aumentada por su resistencia a la eliminación, de hecho este tipo de rootkits puede persistir pese al reemplazo del sistema operativo de la computadora e inclusive después del reemplazo del disco duro.

Virtual Kits

Un rootkit virtual es un rootkit que está codificado y diseñado específicamente para entornos de virtualización. Su objetivo es el mismo que los rootkits tradicionales, y los componentes son en gran parte los mismos, pero la técnica es completamente diferente. La principal diferencia es que el objetivo del rootkit ha pasado de modificar directamente el sistema operativo a subvertirlo de forma transparente dentro de un entorno virtual. En resumen, el rootkit virtual contiene una funcionalidad para detectar y, opcionalmente, escapar del entorno virtual (si está

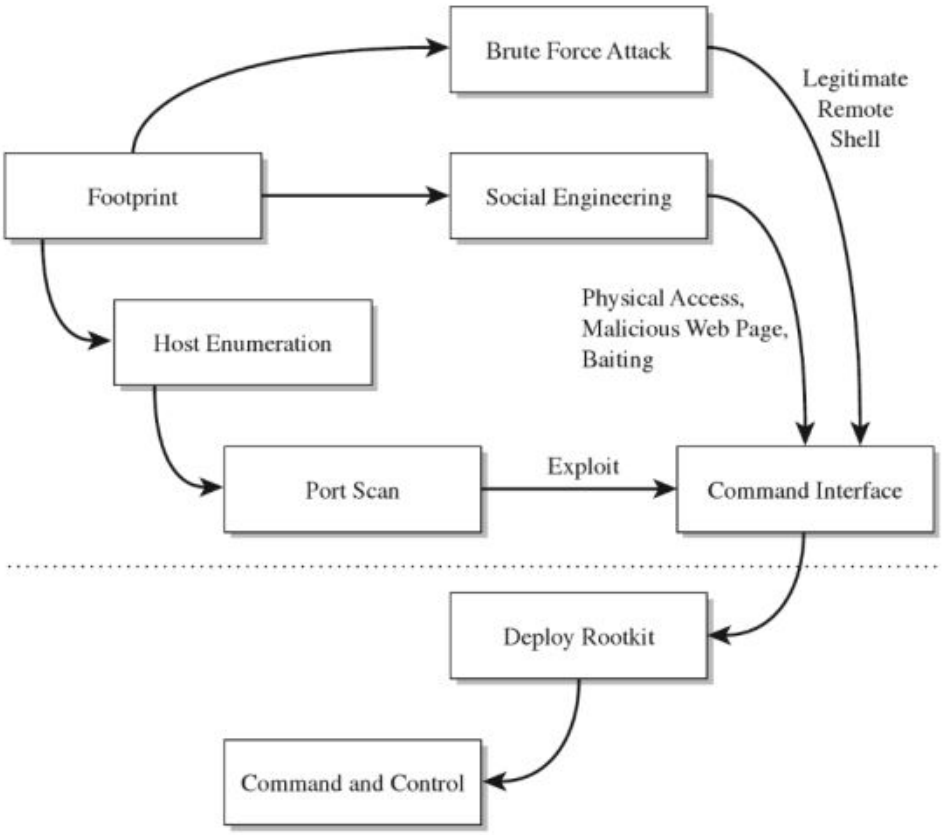
implementado dentro de una de las máquinas virtuales invitadas), así como también secuestrar completamente el sistema operativo nativo (host) instalando un hipervisor malicioso debajo. El hipervisor es el componente de la máquina virtual de hardware que maneja la virtualización a nivel del sistema para todas las máquinas virtuales que se ejecutan en el sistema host. Administra la asignación de recursos y las ejecuciones entre el hardware físico y virtual, permitiendo que dos o más sistemas operativos compartan los recursos del sistema.

FUNCIONAMIENTO DE UN ROOTKIT

Para que un rootkit pueda ser puesto en funcionamiento el atacante primero tiene que identificar la computadora a atacar y todos su servicios que están activos para poder encontrar una manera de ganar acceso a una shell de dicha computadora.

Una vez que el atacante tiene acceso al sistema puede tirar comandos de manera arbitraria y tal vez escalar sus privilegios para convertirse en usuario *root*. Por ejemplo, si bajo ataque es un servidor web, el atacantes podría lanzar ataque de inyección SQL y eso podría comprometer la seguridad asociada con el servidor de base de datos. Luego, puede aprovechar sus acceso al servidor de base de datos y adquirir privilegios de administrador y con suerte la contraseña del servidor web será la misma que del sistema operativo, en fin, existen miles de libros que hablan de las distintas maneras de vulnerar un sistema, por el momento asumamos que todo salió bien y el atacante tiene ahora privilegios de superusuario o como comúnmente les conocemos en los sistemas *nix, privilegios de usuario root.

A continuación se muestra un esquema que muestra alguna de las diferentes maneras que tenemos para atacar un sistema, algunas de ellas pueden funcionar sin necesidad de que las otras funcionen pero lo más común es que las técnicas de ataque se combinen.

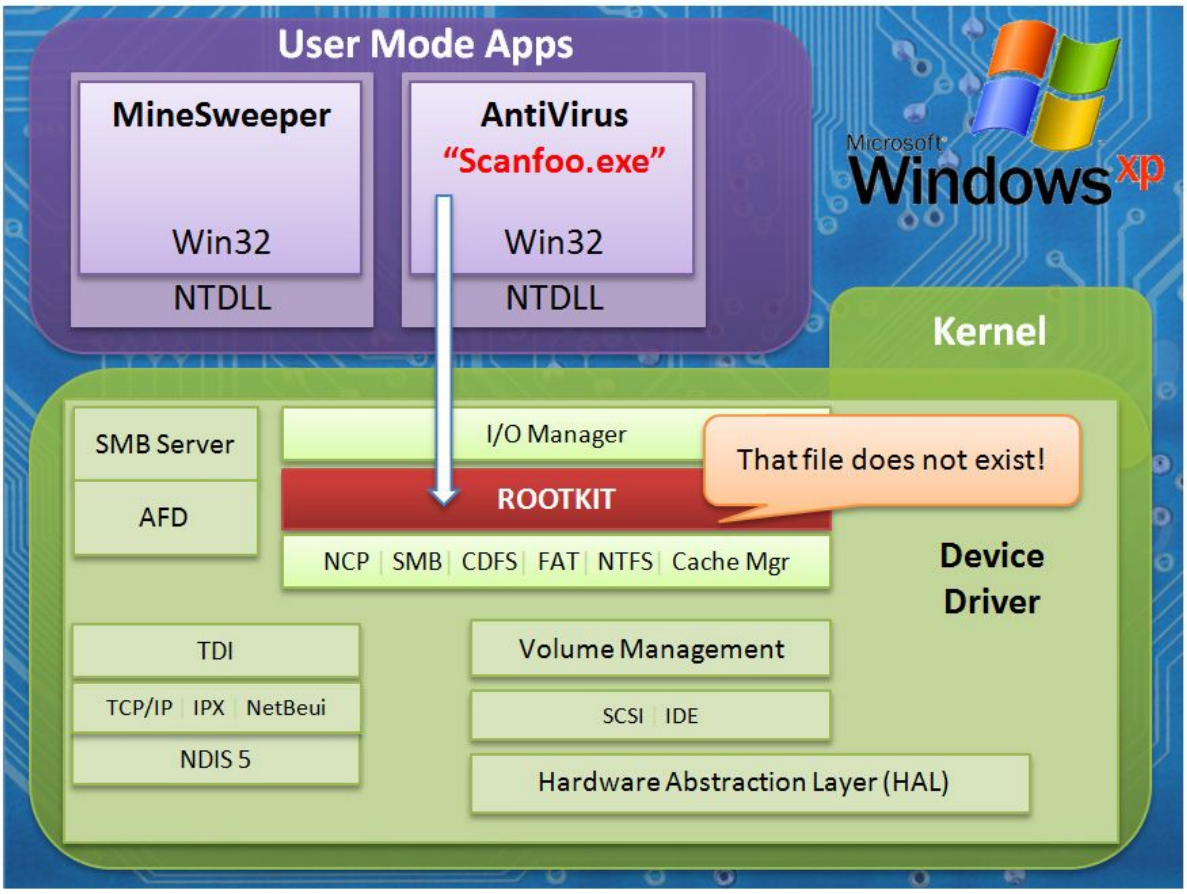


En algunos aspectos el funcionamiento de un rootkit será igual tanto en modo usuario como en modo kernel y lo único que cambiará será la dificultad para prevenirlo, implementarlo, para encontrarlo o para eliminarlo.

En modo usuario y en modo kernel el rootkit funciona implementando un mecanismo llamado “hooking”. La mayoría del software se basa en el sistema operativo para proporcionar información sobre el entorno en el que se está ejecutando. Por ejemplo, una aplicación puede tener archivos que necesita ejecutar, o archivos de datos en los que necesita escribir, o claves de registro que se utilizan para configurar la aplicación. La aplicación pregunta al sistema operativo sobre esos archivos y claves de registro mediante la interfaz de programación de aplicaciones (API) proporcionada por el sistema operativo. (Los ejemplos incluyen FindFirstFile para enumerar archivos, o RegOpenKeyEx para obtener acceso al registro). El Sistema Operativo luego devuelve la información apropiada a la aplicación.

Para ocultarse, los rootkits secuestran estas API y están atentos a cualquier pregunta que una aplicación le pueda hacer que pueda ser incriminatoria. Así que imagine que una aplicación

pregunta: “Sistema operativo, ¿puede mostrarme el contenido del archivo en c: \ foo.exe?” El rootkit intercepta la pregunta antes de que llegue al sistema operativo y responde rápidamente (como si fuera el sistema operativo), "ese archivo no existe".



Usualmente los rootkits son traídos al juego al final del ciclo de ataque y por esta razón son conocidos como herramientas de post-exploit. Una vez que se han escalado los privilegios de un sistema y se tiene el control es casi natural querer retener ese acceso para poder seguir explotando la información o la infraestructura del sistema y es en este momento en donde los *rootkits* entran en acción.

TÉCNICAS DE DETECCIÓN DE ROOTKITS

Una vez que el instalador de rootkit ha podido hacer su trabajo, las cosas se complican y los métodos de detección son más complejos. Lo interesante de los rootkits es que, por naturaleza, son paradójicos. El autor del rootkit tiene dos requisitos básicos para cada rootkit que escribe:

• El rootkit debe permanecer oculto.

• El rootkit debe ejecutarse en los mismos recursos físicos que el host que ha infectado; en otras palabras, el host debe ejecutar el rootkit.

Estos dos requisitos básicos crean una paradoja. Si el sistema operativo o, en el caso de un rootkit virtual, proceso / máquina debe conocer el rootkit para ejecutarlo, ¿cómo puede el rootkit permanecer oculto? La respuesta: la mayoría de las veces, el rootkit no puede permanecer oculto. Debe recordar que la detección de rootkits, como toda detección de malware, es una carrera de armamentos, y la carrera de armamentos avanza cada lado opuesto según sea necesario.. Muchas nuevas aplicaciones de antirootkit y técnicas de detección de rootkits están disponibles para el uso del público; sin embargo, todas las aplicaciones de detección de rootkits requieren una cantidad considerable de conocimientos técnicos para operar, y los proveedores comerciales, que normalmente hacen que el software sea fácil de usar, no han alcanzado realmente la última tecnología de detección de rootkits.

Existen varias formas de conectar un rootkit en el kernel o en el modo usuario, ya se había mencionado que esto lo hacen a través de un gancho o “hooking”, para cada tipo de enganches existen diferentes técnicas capaces de detectar si existe un rootkit, a continuación se describen brevemente.

ENGANCHES SSDT

Una de las técnicas más simples y usadas, la tabla de descriptores de servicio del sistema o el enganche SSDT, es bastante fácil de detectar, y casi todas las herramientas disponibles detectan los enganches SSDT. El kernel de Windows mantiene una tabla de todas las funciones que se exportan para que los utilicen los controladores. Un autor de rootkits simplemente necesita encontrar esta tabla, su versión en la sombra, que es utilizada por el subsistema GUI, y reemplazar el puntero en la tabla que apunta a la ubicación real de la función del kernel con la versión del rootkit de la función del kernel. Al reemplazar ese puntero en KiServiceTable, que almacena la dirección de todas las funciones del núcleo dentro del sistema operativo, el autor del rootkit cambia el flujo general de memoria dentro de la tabla.

ENGANCHES IDT

La tabla de descriptores de interrupción (IDT) se enlaza de la misma manera que el método de enlace SSDT . La tabla tiene un conjunto de punteros de función para cada interrupción. Para enganchar la interrupción, el rootkit reemplaza la interrupción con su propia función.

ENGANCHES IAT

Uno de los ganchos de usuario más prominentes es el gancho IAT. La detección de ganchos IAT es sencilla. Primero, los detectores de rootkits encuentran la lista de DLL que requiere un proceso. Para cada DLL, el detector carga esa DLL y analiza las funciones importadas y guarda las direcciones de importación para esas funciones DLL. El detector de rootkits luego compara esa lista de direcciones con las direcciones importadas utilizadas por todas las DLL dentro del proceso que se examina. Si el detector encuentra alguna discrepancia, esto indica que la función importada puede estar enganchada.

TÉCNICAS DE ELIMINACIÓN DE ROOTKITS

La eliminación de rootkits puede ser difícil, especialmente para los rootkits que se han incorporado a los kernels del sistema operativo, al firmware o en los sectores de arranque de dispositivos de almacenamiento. Si bien algunos programas anti rootkit pueden detectar y eliminar algunos rootkits, este tipo de malware puede ser difícil de eliminar por completo.

Un método para eliminar rootkits es reinstalar el sistema operativo, que en muchos casos eliminará la infección. La eliminación de rootkits del cargador de arranque puede requerir el uso de un sistema limpio que ejecute un sistema operativo seguro para acceder al dispositivo de almacenamiento infectado.

Al reiniciar un sistema infectado con un rootkit de memoria se eliminará la infección, pero es posible que se requiera más trabajo para eliminar la fuente de la infección, que puede estar vinculada a las redes de comando y control con presencia en la red local o en la Internet pública.

Fuentes de consulta:

Sean Bodmer, Aaron LeMasters, Michael A. Davis, Christopher C. Elisan. (2016). Hacking Exposed Malware & Rootkits: Security Secrets and Solutions, Second Edition. New York: McGraw-Hill Education

Reveren Bill Blunden. (2009). The Rootkit Arsenal: Escape and Evasion. Burlington: Jones&Barlett learning

https://www.adlice.com/kernelmode-rootkits-part-1-ssdt-hooks/

https://slideplayer.com/slide/6278728/