Ataque

¡Hackea Windows Vista!

Cristián David Argentero

Grado de dificultad

Queda, absolutamente, demostrado que en El Mundo de las TIC´s (Tecnologías de la Información y Comunicaciones): El 1° puede llegar a ser el último y el último puede lograr alcanzar la punta. MS Windows Vista... ¿Un Candidato que se coronó Rey antes de conseguir „Su Legado”?

Cuenta una bella historia (ya conocida en el Mundo de las TICs) que Microsoft Corp. invitaba a los asistentes expertos y especialistas en Seguridad Informática -véase, Hackers - a la prestigiosa conferencia de Black Hat (Famoso y reconocido MegaEvento sobre Seguridad en las TICs (Tecnologías de la Información y Comunicaciones) realizado cada año en Las Vegas. Es un acontecimiento que reúne a los gurús más idóneos en la materia a lo largo y lo ancho de todo El Mundo. Es una cita obligada y majestuosa para quienes quieren *degustar la otra cara de la moneda*. Más Información: *http://www.blackhat.com/*) 2006, con el motivo de intentar *reventar* los sistemas de seguridad de Windows Vista (su nuevo Sistema Operativo en el mercado, aparentemente, el más seguro hasta la fecha jamás creado, según *El Tío* Bill Gates). Entonces, como era de suponerse, pasó lo inesperado (para Microsoft Corp.) y lo esperado (para el público investigador y/o seguidor de estos productos, acostumbrados a ello). Pero… ¿Qué ocurrió? Una programadora polaca que trabaja para la Compañía Coseinc Inc (Empresa dedicada al estudio avanzado de Software *Neurálgico* relacionado a Códigos Maliciosos. Más Información: *http://www.coseinc.com/*) consiguió lograr el osado desafío propuesto. Joanna Rutkowska, ha sido la primera persona (hasta el momento) en demostrar que es posible saltarse las medidas de seguridad de Windows Vista. Ella explicó que: *es posible utilizar tecnología virtual para volver indetectables los códigos maliciosos (Malware) e insertarlos en el Kernel, de la misma forma*

En este artículo aprenderás...

- Explotar las vulnerabilidades del nuevo y „jugoso” SO de MS (Windows Vista);
- Conceptos del trabajo con técnicas/estrategias sobre las profundidades del Kernel del SO;
- Tecnologías de virtualización por Hardware y Software dedicado a la emulación del mismo;
- Programación de Mini-Exploit personalizado para lograr los objetivos propuestos;
- Posibles (y futuras) soluciones a tales inconvenientes.

Lo que deberías saber...

- Las nociones preliminares del funcionamiento „interno” de un SO;
- Fundamentos básicos de la Programación Orientada a Objetos (OMT);

que actuaría un RootKit (Conjunto de utilidades (herramientas) para explotar las vulnerabilidades de un sistema informático y hacerse de él sin los permisos otorgados legítimamente por el administrador a cargo. Más información: *http://www.rootkit.com/.) para obtener su objetivo; inclusive, pueden usarse Drivers de dispositivos sin necesidad de que estén firmados digitalmente (algo que se intenta impedir en tal Software Base).* También, dijo que: *el hecho de que los sistemas de seguridad de Windows Vista hayan sido violados no significa que éste sea un sistema operativo inseguro; simplemente, no es tan seguro como se ha dicho.* Por eso, a continuación develaremos al *monstruo de dos cabezas* que sorprendió e hizo temblar al mismísimo Microsoft y a su más reciente *pequeña criatura* gestada.

Entrando en Clima

Para poder hablar de las técnicas y estrategias abocadas a tales temas -ideados por Joanna Rutkowska (Foto ID)- es fundamental precisar los conceptos que determinan sus instauraciones.

Por ello, expondremos (sencilla y brevemente) los principios que hicieron permisible la siguiente nota:

Listado 1. *Detector VMM basado en la Instrucción „SIDT Trick" del Procesador*

```
/* Detector VMM: Basado en "SIDT Trick"
  * Escrito por: Joanna Rutkowska
  * Adaptado por: ^[(CR@M3R)]^
  * Puede ser compilado con DevC++
      (para Windows) y ejecutado en su SO
  */

#include <stdio.h>
int main () {
  unsigned char m[2+4], rpill[] =
      "\x0f\x01\x0d\x00\x00\x00\x00\xc3";
  *((unsigned*)&rpill[3]) = (unsigned)m;
  ((void(*)())&rpill)();

  printf ("idt base: %#x\n", *((unsigned*)&m[2]));
  if (m[5]>0xd0) printf ("Inside Matrix!\n", m[5]);
    else printf ("Not in Matrix.\n");
```

- Blue Pill (Pastilla Azul): utiliza la tecnología de virtualización por hardware en arquitecturas de x64b, SVM/Pacífica de AMD o Bit VT de Intel, para tomar el control del SO (Sistema Operativo) donde se aloja, instalándose on-the-fly (*al vuelo*), sin necesidad de reiniciar el equipo y sin hacer modificaciones en la BIOS (Sistema Básico de I/O) ni en el sector de arranque del disco.

Éste, se fundó con la intención de ser un RootKit indetectable para las mecanismos de seguridad de Windows Vista, aunque se conociera su algoritmo o su código. Para ello, virtualiza el SO donde se instala e inserta el conjunto de códigos maliciosos (malware) en el mismo, sin ser descubierto.

Puede funcionar en los SO MS Windows Vista distribuidos como *Plataforma x64 Bits*.

- Red Pill (Pastilla Roja): utiliza un método genérico (basado en un bug de diseño mal implementado) que permite insertar código arbitrario en las entrañas del Kernel (núcleo del SO) en MS Windows Vista -*Beta 2 Plataforma x64 Bits*- sin percatase de saber si éste se encuentra firmado digitalmente o no. Tal procedimiento puede ser usado, por ejemplo, para sortear el sistema de certificados genuinos requeridos en la instalación de drivers (necesarios para la funcionalidad adecuada de dispositivos en el SO).

Vale aclarar que, ambas nociones, son independientes unas de otras. Pero, pueden combinarse para dar un resultado de inseguridad aún mayor que si se probaran por sí solas.

Por lo tanto, su única relación estrecha es que las 2 (dos) pueden ser funcionales en una cuenta de administrador y en un SO MS Windows Vista en Versionas Betas que operen

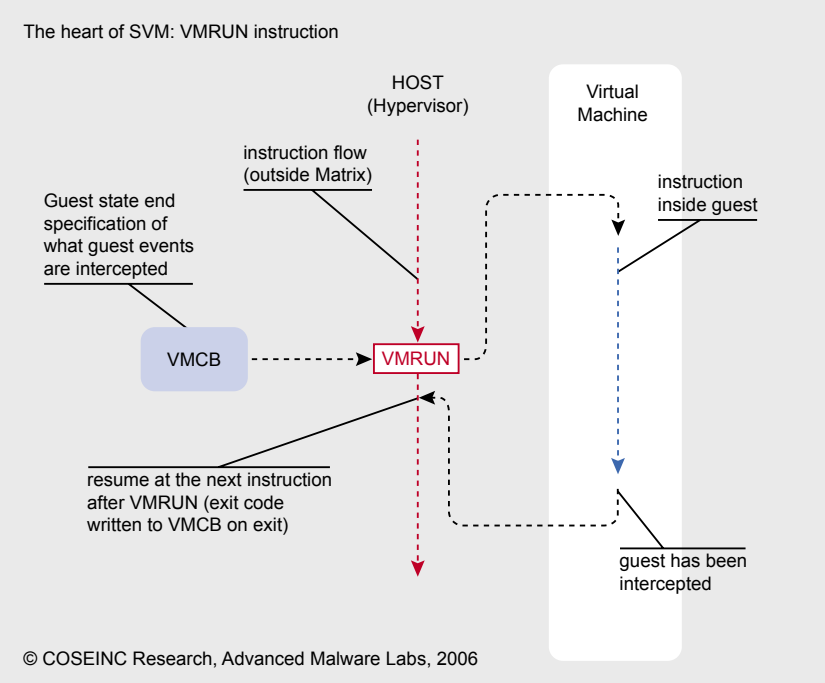

Figura 1. *El „corazón" de SVM/Pacífica (AMD): Instrucción „VMRUN"*

sobre Plataformas de x64 Bits (otorgada mediante el Software Base y los Microprocesadores que soporten tales arquitecturas).

Además, se hace énfasis en indicar que se trata del SO MS Windows Vista Beta 2 x64 Bits (versión publica). Lo cual, establece que lo más probable y normal sea corregir tales fallas cuando salga una distribución final (definitiva) del mismo.

Allí, teóricamente, ocurrirían 2 (dos) cosas que evitarían tales inconvenientes:

- Por defecto, el SO trabajaría en cooperación con un virtualizador propio, de *capa fina*, que redirigiría cualquier llamada al hardware. Éste detectaría que otra VM (máquina virtual) intenta meterse y apoderarse de su control, por lo que podría interactuar con el Sistema Operativo y alertar al usuario de esas intenciones quién, a la vez, frenaría la/s acción/es que quiere/n llevarse a cabo. Pues, así, se solucionaría el problema Blue Pill (Pastilla Azul);
- Estrictamente, sólo se podrían instalar drivers de dispositivos autorizados por su firma digital, con Certificación WHQL (Laboratorio de Control de Calidad de Hardware de Windows).

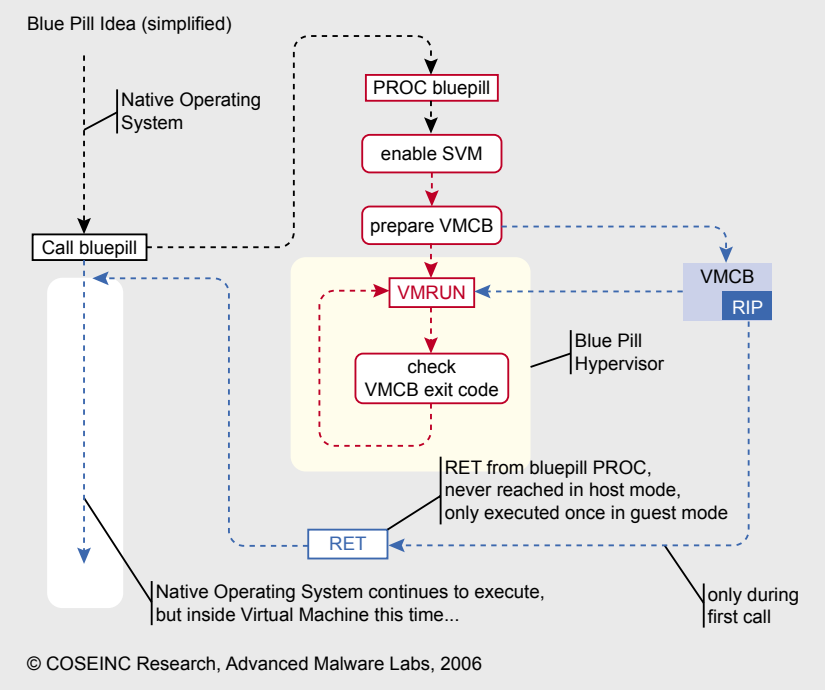

Figura 2. *La idea simplificada de Blue Pill (Píldora Azul)*

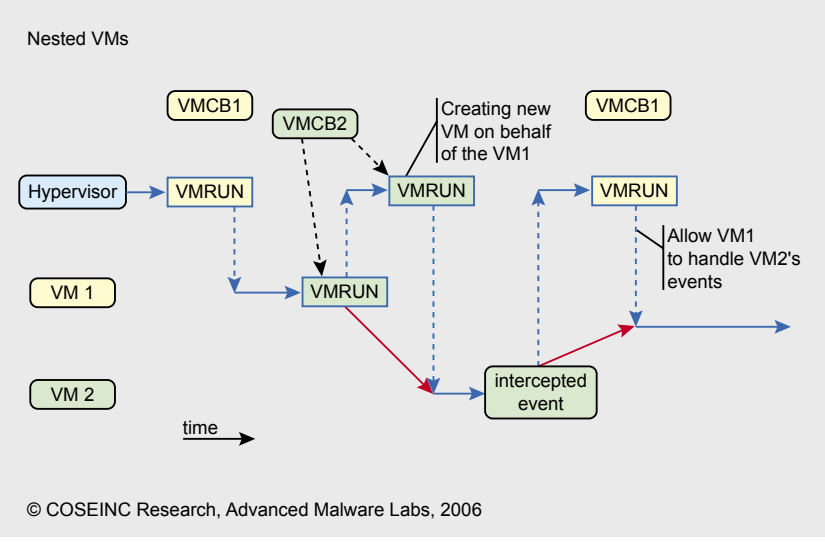

Figura 3 *Anidamiento de Máquinas Virtuales (VMs)*

Por lo que, el otro inconveniente (Blue Red -Pastilla Roja-) también sería resuelto.

Entonces, a continuación, dejo a su criterio la evaluación técnica del desarrollo de ambas metodologías…

Poniéndonos a Punto

El posterior análisis (estructural, sistemático, metódico y conciso) *desplegará* los fundamentos y las características esenciales de los artilugios anteriormente descritos. Por ende, estudiaremos y/o examinaremos el siguiente contenido:

Parte I - Blue Pill (Pastilla Azul): Creando Código Malicioso -Malware- Indetectable

Todos los RootKit´s y Backdoors actuales, de los cuales estoy enterado, se basan en una concepción (teoría). Por ejemplo: FU fue asentado en una idea de desatar bloques de EPROCESS de la lista de procesos activos en el Kernel del SO, Shadow Walker fue cimentado en un concepto de engañar al visitante de una Página Web (preparada para el caso) y de marcar algunas partes del cuerpo del sitio (Body Site) como *inválida* y así poder ejecutar código arbitrario, Deepdoor hizo lo suyo cambiando algunos campos en la estructura de datos NDIS; tomando control remoto del sistema, Etc…

Por lo tanto, una vez que sepas *de qué concepto* se trata, podrás, al menos hipotéticamente, detectar el Código Malicioso que se encuentra en tu máquina.

Ahora, imagina un Malware (desconocido, aún, en su tipo) con capacidades de ser imperceptible. Del cual no se pueda confiar en la oscuridad de su concepto (teoría). Éste, no podría detectarse (en la práctica) aún cuando su algoritmo (conjunto ordenado y finito de operaciones que

permiten el cálculo de su notación) se diera a conocer en público.

Vayamos más lejos e imaginémonos que, incluso, su código de compilación se encuentre In The Wild (*en la calle*), pero que todavía no se consiguiera manera alguna para detectar que esta criatura está funcionando en vuestras máquinas… ¡Sería un potencial desastre libre por doquier!

Sin embargo, aunque pueda sonar un tanto absurdo, se ha estado trabajando en una tecnología código-nombrada Blue Pill (Pastilla Azul), que es justamente sobre ésa presunción (imperceptible al 100%, según las condiciones contemporáneas) en la que uno se apoya para verter sus postulados arriba dispuestos sin ninguna duda. (Figura 1).

La idea detrás de tal postura es mera: tu SO *traga* La Píldora Azul (Blue Pill) y se despierta dentro de una matriz controlada por el Hypervisor Azul Ultrafino de la Píldora, es decir, un prototipo subyacente de ejecución que bloquea, encripta y oculta procesos y/o procedimientos totalmente manipulables para el control de eventos en el SO, pero indetectables. Todo esto sucede *en marcha* (on-the-fly), o sea, sin el reinicio del sistema, y no hay patrones visibles de comportamientos extraños en el funcionamiento de los recursos (lógicos y físicos) de tu ordenador.

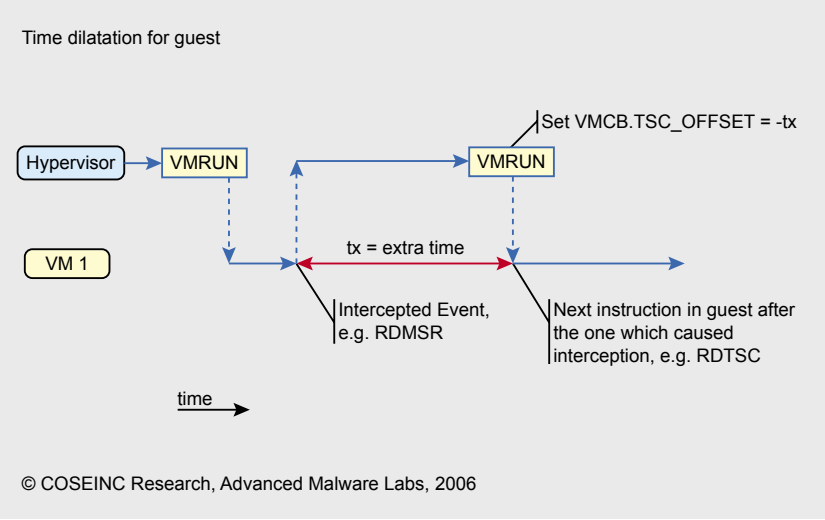

Figura 4. *Incapacidad de ejecución y dilatación en el tiempo a nivel de Usuario Invitado*

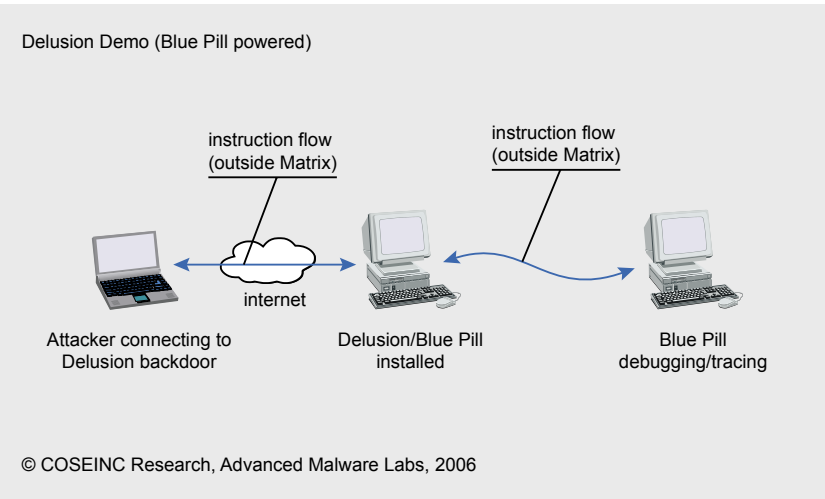

Figura 5. *Esquema radical de interoperabilidad (demostración de engaño con Blue Pill)*

Empero, aunque no lo notes, ahora todos los dispositivos son completamente accesibles al SO, que está ejecutando una máquina virtual (VM) interior. Esto ocurre gracias a las tecnologías de última generación que consienten la virtualización del hardware con respecto al software (sincronizados). Los paradigmas más reconocidos que hacen posible tal obrar, en nuestros días, son: SVM/Pacífica de AMD o Bit VT de Intel. (Figura 2).

Dada la infografía de arriba, podemos entender como el SO sigue creyendo que se está ejecutando en la PC, cuando en realidad se ejecuta dentro de una máquina virtual (VM) controlada por el programa que lo *infectó*.

De este modo, la aplicación se vuelve *virtualmente invisible*, dado que Windows Vista sólo puede ver la máquina virtual (*el mundo que ha sido puesto ante sus ojos para ocultarle la verdad*). (Figura 3).

Asimismo, un programa de esta magnitud *corriendo* dentro de una máquina virtual (VM) puede ser sospechoso, pero un SO dentro de una máquina virtual (VM) que a su vez está dentro de otra máquina virtual (VM) ejecutando la misma aplicación, muestra desconcierto de razonamiento lógico. Sin embargo, su punto en contra (defecto) es la degradación en su rendimiento y performance (debido al consumo extremo de requisitos para mantenerse estable en un sistema con tantas peticiones de este tipo), lo que puede ser *el talón de Aquiles* para producir una vacuna que lo evite y prevenga su intrusión desautorizada a posteriori de haberse entregado a la intervención dañina. Incluso, contamos con las opciones para desactivar estas VMs por Hardware desde la BIOS y el SO (indicadas por leyendas que hacen referencia a sus nombres y funciones), las cuales no resultan convenientes recomendar, salvo para una *salida de emergencia* al apuro oportuno.

Otro mecanismo para divisarlo podría ser llenando toda la memoria RAM del equipo, de esta manera, el programa sólo podría ir a la

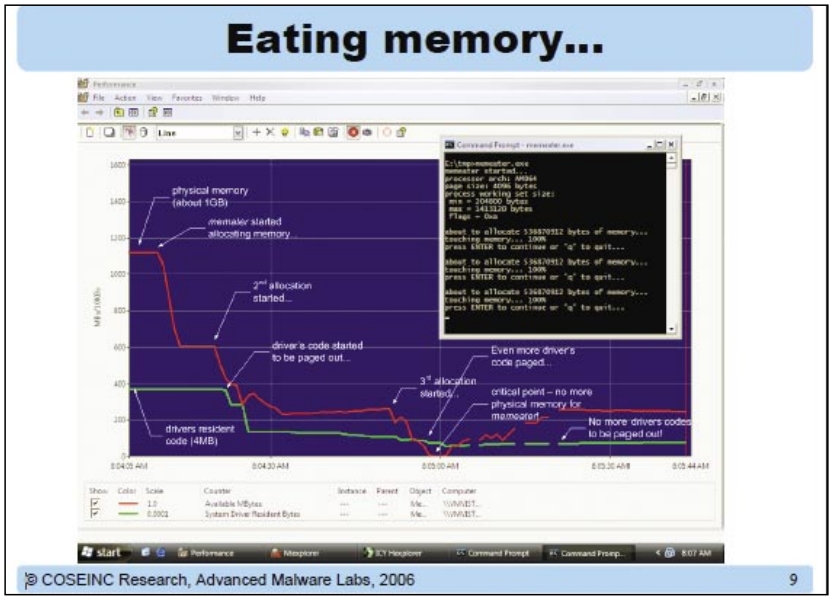

Figura 6. *Comiéndose a la memoria del sistema*

memoria de intercambio virtual del SO (Swap), donde ya no tendría la capacidad de seguir ejecutándose y, por lo tanto, tampoco podrá mutar ni moverse allí.(Figura 4). En definitiva, el esquema radical de su interoperabilidad sería el que a continuación se muestra en Figura 5.

Parte II - Red Pill (Pastilla Roja): Sometiendo a engaños el Núcleo del Sistema -Kernel-

¿Quién no conoce la importancia y relevancia del Kernel? ¿Quién no se ha visto *know out* (KO) por *la caída* del mismo? ¿Por qué los mayores (y mejores) ataques a nivel de Seguridad Informática se suceden sobre este? ¿A qué se debe tan renombrada autoridad y dogma? ¿Y…?

Para sintetizarlo en unas pocas líneas, el Kernel es el Núcleo de todo Sistema Operativo (SO); el sostén del mismo, la *columna vertebral* que permite la sincronización entre los Recursos Físicos (Hardware) con los Recursos Lógicos (Software) de una máquina (llámese PC). El encargado de *mantener en pie* (armazón) a cualquier plataforma de Software Base en posesión de un equipo…

Pero, tanto mérito no le concede *ser perfecto* y, lejos de estar de ello, tiene sus falencias.

En tal caso particular, MS Windows Vista sufre en su célebre incursión por las VMs de una patología un tanto grave (clasificación crítica) que permite insertar código arbitrario en la médula del Kernel (núcleo del SO) sin percatase de conocer si éste se encuentra firmado digitalmente o no. Lo que sería capaz de eludir el sistema de certificados genuinos requeridos para la instalación de drivers (precisos para el funcionamiento idóneo de dispositivos en el SO).

Se trata de una instrucción que no es privilegiada en los rangos del Microprocesador, la SIDT. Es decir, no causa una excepción cuando se la aplica. Pero, si es sensitiva. O sea, da un resultado distinto en la VM (Level 1) que en la RM (Level 0). Lo cual, no debería ser anunciado de ninguna manera. Tal artimaña es conocida con el nombre de Exclusion Trap (*Trampa de Exclusión*).

En conclusión, ejecutar una instrucción SIDT da un resultado distinto dependiendo de donde se encuentro operando el Kernel del SO con respecto al Microprocesador.

Algo que no tendría que ser así, por el simple hecho de que puede delatar su ubicación real a otras aplicaciones en memoria que pueden intentar *persuadirlo* para aprovecharse de tal vulnerabilidad y, por ende, esgrimir con sus actividades perjudiciales.

El siguiente Código de Programación (El programa puede fallar en los sistemas con protección PAX/X^W/grsecurity. Porque la variable rpill no es marcada como ejecutable. Para hacerlo, debe usarse `mprotect()` en la asignación rpill con el atributo de PROT_EXEC.

Otra solución sería usar, simplemente, `asm()` como *palabra clave* en lugar de shellcode (como buffer). Sin embargo, este código de programa debe ser considerado (más bien) como un *esqueleto* para construir el suyo. Mi meta era hacerlo tan simple y portátil como sea posible. El resto, se lo dejo al desafío de su imaginación…) efectuado en C++ descubre (fehacientemente) lo que intento confesarles:

Como antes lo dijimos, lo volvemos a reiterar… El corazón de este código, ciertamente, es la instrucción SIDT (aquí como 0F010D[addr]) que tiende a los volúmenes de interrupción del descriptor de la tabla del registro (IDTR) en el destino local de memoria que se esté operando.

Debido a que hay sólo un IDTR registrante como verdadero (True), pero (por lo menos) dos SO que "corren" concurrentemente (el anfitrión -host- y el invitado -guest-), VMM necesita relocalizar el IDTR del invitado en un lugar seguro, para que no cause conflictos con el anfitrión y termine desbordando al sistema. Desgraciadamente, VMM no puede saber si (y *cuando*) el proceso que se ejecutó en el invitado desencadenó la instrucción SIDT en el anfitrión, porque no es una orden privilegiada (y, como inicialmente describimos, no genera una excepción). Así, el proceso consigue la dirección IDT sin mayores inconvenientes y, por supuesto, sin el consentimiento del usuario a cargo.

Por ejemplo, fue observado que en VMWare la dirección relocalizada IDT estaba en el sector 0xffXXXXXX, mientras que, en Virtual PC, la sección era 0xe8XXXXXX; respectivamente.

¿Extraño, no…? Sabiendo que una misma máquina (PC) comparte un cabal recurso de privilegios en instancias del Microprocesador que el SO (más específicamente el Kernel del mismo) impone. La captura de pantalla que veremos en Figura 6, mostrará su representación substancial de interoperabilidad.

Igualmente, se pretende que en un futuro, el SO (comandado por su Kernel) sólo pueda instalar drivers de dispositivos autorizados por firmas digitales con Certificación WHQL (Laboratorio de Control de Calidad de Hardware de Windows). Por lo que, tal contingencia sería resuelta sin mayores peligros e inconvenientes en su devenir.

Pues, entonces, como veredicto final… Reforzar la prevención, preferentemente, con lo *malo conocido*, sin arriesgarse en ir en búsqueda de lo *bueno por conocer*.

Esa sería mi principal recomendación; hasta que, por lo menos, las empresas en convenio, nos den una solución pertinente a la complicación existente. ¿Hacemos trato…?

Al fin llegamos a la meta: ¿¡Lo logramos!?

Podríamos decir, en un *juego inteligente de palabras*, que el apasionante y sorpresivo Mundo de la (IN)seguridad Informática nos sobrepasa y colapsa cualquier seguridad vigente en tiempos inconcebibles, pero reales. ¿Qué lo corrobora y/o cerciora?

Sin ir más allá, lo aquí dispuesto (mecanismos -cariñosamente llamados: Blue Pill / Red Pill- ideados y desarrollados por Joanna Rutkowska en técnicas y/o estrategias de *Hackeo a Windows Vista*) dan una *pequeña gran pauta* de lo lejos que estamos de una forma bastante efectista y eficiente de poseer un SO (promocionado por Microsoft Corp.) que sea robusto, confiable, versátil y flexible a las exigencias de los usuarios alrededor de todo el mundo y de *los tiempos que corren*.

Pero… Sin olvidarse de hacerlos ¡SEGUROS!

Muchos le encontraran y/o consideraran a esta nota un halo de fantasía y superstición. Sin embargo, las denominaciones y tendencias aquí desempeñadas y aprendidas se apoyan en el pedestal hegemónico (por analogía) a la tan admirada Película Matrix donde la gente cree vivir en un mundo real cuando en realidad viven en úteros artificiales controlados por máquinas. Del mismo modo, Windows Vista cree ejecutarse en una máquina real cuando en realidad se ejecuta en una máquina virtual y… Allí es dónde *comienzan los dolores de cabezas*. ●

En la Red

- *http://www.kriptopolis.org/node/2688*
- *http://invisiblethings.org/*
- *http://www.eweek.com/category2/0,1874,1237918,00.asp*
- *http://feeds.computerworld.com/Computerworld/TopNews*
- *http://www.internetnews.com/security/*
- *http://www.internetnews.com/7-days/*
- *http://news.zdnet.com/2038-1009_22-0-topic.html?id=6219&name=Windows+Vista*
- *http://www.networkworld.com/news/*
- *http://www.vmware.com/standards/index.html*
- *http://www.winehq.com*

Sobre el Autor

Argentero, Cristián David (alias ^[(CR@M3R)]^)... Es estudiante de Lic. en Informática y gran apasionado por las TIC´s (Tecnologías de la Información y Comunicaciones). Sus actuales condiciones en la materia lo han llevado al abordaje de la Información y el Conocimiento de manera: Universal, Libre y Gratuita; prestando servicios y soluciones en la profesión de manera idónea y acorde a las exigencias dispuestas. Contacto con el Autor: *cdaznet@hotmail.com*