Haciendo un *bypass* a lista blanca de aplicaciones de McAfee en sistemas de infraestructura crítica

Traducción al español de artículo público

Titulo Original: *Bypassing McAfee's Application Whitelisting for Critical Infrastructure Systems*

Autor: Responsible: René Freingruber - SEC Consult Vulnerability Lab, Vienna

Confidentialy Class: Public

Traductor: Francisco Alva Segovia

Contenido

1. Resumen

El presente trabajo describe los resultados de una investigación realizada por la *SEC Consult Vulnerability Lab* para la seguridad del producto Control de aplicación de McAfee (*McAfee Application Control*). El producto es un ejemplo de las soluciones basadas en listas blancas de aplicaciones que se puede utilizar para fortalecer aún más los sistemas críticos, tales como los sistemas de servidores en ambientes SCADA o en sistemas de cliente con altos requerimientos de seguridad, tales como estaciones de trabajo de administradores.

Las listas blancas de aplicaciones es un concepto que trabaja mediante listas de control que permiten la ejecución de aplicaciones instaladas en el sistema y la denegación en aquellas que no estén incluidas. Esto debería denegar la ejecución de malware y, por lo tanto, proteger contra ataques de amenaza persistente avanzada (*APT- advanced persistent threat*). *McAfee Application Control* es un ejemplo de este tipo de software, se puede instalar en cualquier sistema, sin embargo, el principal campo de aplicación es la protección de infraestructura altamente crítica. Aunque la característica principal del producto es la lista blanca, adicionalmente soporta características de seguridad adicionales, incluyendo protección de lectura y escritura, así como protección de corrupción de memoria.

Durante la investigación de la versión para Windows del producto (versión 6.1.3.353) se comprobaron las debilidades y fallas en el diseño e implementación. Se identificaron varios métodos que pueden ser utilizados para eludir la característica principal de *McAfee Application Control,* iniciar la ejecución de programas que no estaban definidos en la lista blanca y por lo tanto que no estaban autorizados. Durante la auditoría, diferentes métodos fueron desarrollados utilizando los vectores de ataque más comunes hoy en día. En la mayoría de los casos el ataque inicial fue impedido por la aplicación, sin embargo, solamente con pequeños cambios se logró eludir la protección e infectar el sistema.

Estos escenarios consistieron en diferentes ataques de ingeniería social y explotación mediante corrupción de la memoria. *McAfee Application Control* pretende implementar protección contra ataques de corrupción de memoria (Por ejemplo *buffer overflow*). De hecho estas protecciones solo corresponden a las protecciones típicas de sistema operativo como ASLR y DEP [1]. Por lo tanto los exploits desarrollados para nuevos sistemas funcionan sin modificaciones porque ya incluyen bypass de ASLR y DEP.

Fueron identificadas fallas de diseño y debilidades adicionales que se pueden utilizar para eludir la protección de lectura y escritura, además existen varias vulnerabilidades en el controlador del kernel, el cual puede ser usado para bloquear el sistema.

[1] ASLR.- Address space layout randomization; DEP.- Data Execution Prevention.- Son tecnologías orientadas a la prevención de ejecución de *shellcodes* y de explotación de vulnerabilidades de *buffer overflow* (nota del traductor)

Teniendo en cuenta que el principal campo de la aplicación es la seguridad de infraestructuras críticas (por ejemplo servidores que regularmente inspeccionan las temperaturas de reactores de centrales eléctricas) estos ataques pueden causar serios problemas. Como nota final, distribuciones de *McAfee Application Control* con componentes muy desactualizados que datan de 1990 también pueden ser explotados.

2. Introducción

"El software *McAfee Application Control* proporciona una forma eficaz para bloquear aplicaciones y código no autorizados en los servidores, equipos de escritorio corporativos y dispositivos de función fija. Esto solución de administración centralizada basada en lista blanca utiliza un modelo de confianza dinámica con características innovadoras que detienen las amenazas persistentes avanzadas – sin la necesidad de actualizaciones de firmas o de gestión de listas de trabajo intensivo[i]

McAfee Application Control Es un software el cual puede ser usado para fortalecer los sistemas operativos mediante el uso de listas blancas de aplicaciones. Esto es especialmente utilizado para proteger infraestructura crítica. La infraestructura necesita actualizaciones que en ocasiones no pueden ser instaladas debido a ciertos requisitos de confiabilidad y disponibilidad que son diferentes a otros campos de aplicación. Ejemplos de estos requisitos pueden ser encontrados en ambientes SCADA donde las actualizaciones no son aplicadas para prevenir el riesgo de un defecto en un paquete de actualización. Teóricamente la aplicación debe bloquear los ejecutables no incluidos en la lista blanca y por lo tanto evitar la ejecución de un código administrado por un atacante. La siguiente cita se encuentra en la página del producto:

"*Minimizar la aplicación de parches mientras se protege la memoria, le permite retrasar la aplicación de revisiones hasta que se normalice el ciclo regular de parches. Además. Previene que las aplicaciones en la lista blanca puedan ser explotadas vía ataques de desbordamiento de memoria (Buffer Overflow) en sistemas basados en Windows de 32 y 64 bits."* [i]

El objetivo de este trabajo es describir los resultados de la investigación llevada a cabo para verificar si las protecciones previstas por *McAfee Application Control* pueden prevenir o detener los ataques y que tan difícil es para un atacante eludirlas.

La sección 2 describe varias formas de eludir la protección de listas blancas para lograr la ejecución de código arbitrario. Está dividida en tres partes, la primera describe técnicas para recuperar el llamado "ejecución de código básico", que significa una forma básica de ejecución de código sin tener la capacidad de ejecutar código arbitrario. La segunda parte describe como una ejecución básica de código puede convertirse en una ejecución completa de código para alcanzar la meta de eludir la protección de la lista blanca. Esto además incluye una discusión en la seguridad de la protección contra corrupción de memoria, proporcionadas por *McAfee Application Control* y como esto puede ser eludido por atacantes. La tercera y última parte describe conceptos para eludir el

Control de cuentas de usuario (UAC- *User Account Control*) de Microsoft y como puede ser eludido en sistemas ejecutando *McAfee Application Control*.

La sección 3 aborda el concepto de protección de lectura y escritura y como esa protección puede fácilmente ser eludida tan pronto como la ejecución de código es lograda. La sección 4 describe las vulnerabilidades del controlador del kernel identificadas, así como el impacto de los mismos. Los últimos capítulos dan una conclusión de la investigación.

3. Elución de la protección de ejecución de código

La principal característica de McAfee Application Control es la prevención de ejecución de código no autorizado. Por lo tanto, el primer paso es demostrar cómo esta característica es eludida. En la siguiente discusión el objetivo general es conseguir la ejecución completa de código que está compuesta por tres pasos:

El primer paso es lograr la ejecución de código básico, lo cual quiere decir que es la habilidad de comenzar una aplicación incluida en la lista blanca con argumentos específicos.

El segundo paso es convertir la ejecución de código básico en ejecución de código completo, que significa que un código arbitrario (shellcode) puede ser ejecutado. En esta etapa, las protecciones contra ejecución de código de *McAfee Application Control* son completamente eludibles.

Un último paso no necesario es eludir el control de cuentas de usuario (UAC), para lograr la ejecución de código en el contexto de un usuario administrador. Esto es solo posible si la cuenta atacada posee privilegios de administrador; sin embargo, los otros dos pasos pueden hacerse con una cuenta estándar.

3.1 Ejecución de código básico

3.1.1Abuso de tipos de archivo sin revisar: HTA y JS

McAfee Application Control previene la ejecución de archivos de ejecución por lotes que no están registrados en la lista blanca, tales como .bat, .com, .vbs y así sucesivamente, sin embargo, se le da un enfoque de lista negra y es muy común que en las listas negras, estos tipos de archivo se olvidan a registrar y por lo tanto no son verificados. Exactamente esto fue observado en *McAfee Application Control*.

En este caso especial, las aplicaciones HTML (Archivos HTA) no son comprobadas y pueden ser iniciadas sin restricciones. Utilizando el método de ejecución de Wscript. Un objeto Shell puede iniciar otras aplicaciones de la lista blanca en el sistema que serán utilizadas en una etapa posterior del ataque.

El archivo HTA malicioso puede ser entregado a la víctima por ejemplo vía correo electrónico o en una memoria USB (simple ataque de ingeniería social). Si la victima abre el archivo el código comienza a ejecutarse y puede infectar aún más el sistema.

Otro tipo de archivo sin registrar fue identificado durante el análisis. La ejecución de archivos J-scripts tampoco es verificada y puede ser atacada. Para iniciar otra aplicación el mismo código puede ser utilizado como ya se había mencionado. En el escenario más común de ataque, un objeto del sistema de archivos de script perteneciente a ActiveX es utilizado en lugar de escribir un ejecutable (el virus) en el disco duro y después de eso el Wscript.shell es utilizado para iniciar el ejecutable. Sin embargo, éste ejecutable no estaría autorizado en la lista blanca para ser ejecutado y por lo tanto sería bloqueado. La segunda parte en este capítulo trata acerca de las técnicas que pueden utilizarse para tener una ejecución completa de código con argumentos específicos. Exactamente esto se puede hacer utilizando archivos JS o HTA. Por otra parte, es posible implementar el malware completo dentro del archivo JS/HTA.

Es muy probable que existan otras extensiones de archivos sin revisar, lo que también permitirían la ejecución de código básico.

3.1.2 Accesos directos a archivos

McAfee Application Control no impide la ejecución de accesos directos no incluidos en la lista blanca. Por lo tanto un atacante puede crear un acceso directo a un ejecutable pre-instalado incluyendo los argumentos y eludir la lista blanca, para posteriormente enviar el acceso directo a la víctima. Si la victima abre el acceso directo, la aplicación incluida en la lista blanca inicia su ejecución y los argumentos especificados son pasados al ejecutable. Al especificar argumentos maliciosos es posible abusar de la aplicación autorizada en la lista blanca para lograr la ejecución del código.

3.1.3 Memorias USB maliciosas

Si utiliza una de las técnicas mencionadas anteriormente, el archivo malicioso debe ser enviado de alguna manera a la víctima. Esto puede hacerse a través de diferentes canales. Por ejemplo a través de correo electrónico, un recurso compartido de red o una memoria USB. En el caso de una memoria USB existe una mejor oportunidad. El protocolo USB también es compatible con otros dispositivos tales como teclados USB, ratones USB o concentradores USB. Una memoria USB maliciosa puede ser creada para simularse a sí misma como un concentrador USB a la que un teclado USB y memoria USB están conectados. Entonces puede utilizar el teclado para enviar pulsaciones de teclas específicas muy rápidas. Por ejemplo, las pulsaciones de teclado para el botón de Windows y la tecla R se pueden enviar para abrir el cuadro de diálogo de ejecución. Después de realizar esto, se puede iniciar un archivo ejecutable desde la memoria USB. Con este ataque la interacción con el usuario (a partir de un archivo) puede ser reducida y la víctima sólo tiene que conectar la memoria USB manipulada. La prevención de un ataque de este tipo no es la tarea de *McAfee Application Control* directamente, sin embargo, se describe aquí ya que puede ser utilizado como el primer paso para eludir la lista blanca de aplicaciones.

3.1.4 Paso de hash

Otro vector de ataque muy común es aquel donde el atacante ya ha comprometido los sistemas dentro de la red interna y luego utiliza un ataque de paso de hash para comprometer más otros componentes. Si por ejemplo todos los servidores comparten la misma contraseña de administrador también son lo mismo y por lo tanto el hash extraído de uno de los sistemas atacados puede ser usado para autenticarse en otro sistema. Después, estos comandos pueden ser ejecutados en el sistema remoto. Metasploit incluye un módulo para este ataque, sin embargo, no está trabajando contra los sistemas que tienen instalado *McAfee Application Control*. La razón detrás de esto es que el módulo escribe primero todos los comandos en un archivo bat y después lo ejecuta. Debido a que la ejecución de archivos .bat no incluidos en la lista blanca está prohibida, este ataque es bloqueado con éxito. Sin embargo, esto no protege contra el vector de ataque real. Con sólo la modificación de una línea de código del módulo puede ser fijado para que directamente invoque el comando suministrado. Ejemplos de comandos maliciosos que pueden utilizarse para lograr la ejecución de código completo serán presentados en el siguiente capítulo.

3.2 Ejecución de código completo

3.2.1 Abuso de aplicaciones en lista blanca: PowerShell

El típico caso de uso es para fortalecer el sistema y por lo tanto incluir en la lista blanca todas las aplicaciones existentes asegurando así la capacidad de funcionamiento correcto del sistema. El problema con este enfoque es que las aplicaciones que posiblemente pueden ser utilizadas indebidamente por los atacantes también están en la lista blanca.

Un ejemplo de tal aplicación es *PowerShell* y su ejecutable "powershell.exe" que se instala en todos los sistemas más nuevos. Tan pronto como el sistema se fortaleció esta aplicación también está en la lista blanca por defecto y por lo tanto puede ser ejecutado por un atacante. *McAfee Application Control* comprueba si un atacante intenta iniciar un script de PowerShell (un archivo con extensión .ps1) y sólo permite la ejecución de scripts de la lista blanca. Sin embargo, todavía es posible utilizar a powershell.exe especificando el guión completo dentro de los argumentos. Por ejemplo, es posible iniciar calc.exe utilizando el siguiente comando:

```
- powershell.exe –nop –windows hidden –noni –command calc.exe
```

Otro método es utilizar el argumento `encodedCommand`. Utilizando este argumento, scripts complejos pueden ser iniciados. La siguiente secuencia de comandos PowerShell codifica un comando:

```
$cmd="calc.exe"
$bytes = [System.Text.Encoding]::Unicode.GetBytes($cmd)
[Convert]::ToBase64String($bytes)
```

La salida codificada base64 se puede utilizar para invocar calc.exe desde PowerShell. El comando siguiente es un ejemplo de un comando codificado que lanza calc.exe:

```
powershell.exe -enc YwBhAGwAYwAuAGUAeABlAA==
```

Iniciar solo aplicaciones ya incluidas en la lista blanca a través de PowerShell no sería muy poderoso, sin embargo, PowerShell puede hacer más que eso. Es posible acceder e invocar cualquier función expuesta por una biblioteca de Windows como `CreateThread()` o `VirtualAlloc()`. Combinando ambas funciones es posible asignar memoria adicional para almacenar código shell en el proceso actual y comenzar un nuevo hilo para ejecutarlo.

Para iniciar PowerShell con los argumentos maliciosos se puede utilizar una de las técnicas mencionadas anteriormente (véase el capítulo " ejecución de código básico"). La selección de una técnica aplicable depende de la vía de ataque específico, la configuración del sistema y el escenario real.

Aquí PowerShell se utilizó como un ejemplo de una aplicación incluida en la lista blanca por defecto que puede ser objeto de abuso por un atacante. Es muy probable que existan otros programas en la lista blanca, que se puede abusar de la misma forma.

Ejemplos de ello son los depuradores, ya que pueden ser utilizados para escribir shellcodes en el espacio de proceso de una aplicación de la lista blanca para iniciar el código en el contexto de ese proceso. Otros ejemplos son intérpretes de script como Python o Perl, ya que pueden ser usados para iniciar la ejecución de código shell. En ambos casos es posible especificar el script completo con los argumentos como se hizo con PowerShell. Python utiliza el argumento -c y Perl el argumento -e para ejecutar un script especificado en los argumentos.

Tenga en cuenta que es muy probable que existan más de este tipo de aplicaciones en la lista blanca que pueden ser objeto de abuso por un atacante.

3.2.2 applets de Java

Uno de los vectores de ataque más comunes hoy en día es incrustar un applet de Java malicioso en un sitio web y engañar a la víctima para iniciarlo. Las pruebas demostraron que *McAfee Application Control* no impide la ejecución de applets de Java no incluidos en la lista blanca. Por lo tanto es posible lograr la ejecución de código al abusar de los applets de Java si el complemento de ejecución de Java está instalado e incluido en la lista blanca del sistema de destino. Sin embargo, es importante señalar que *McAfee Application Control* protege contra muchas applets maliciosas debido a su naturaleza de ataque. Normalmente, el código de ataque simplemente extrae un archivo de recursos del sistema de archivos y después lo ejecuta. Debido a que la aplicación es revisada en las listas blancas, la ejecución de un ejecutable recién creado se evita. Existen enfoques múltiples para evitar esto. Por ejemplo, es posible iniciar PowerShell y pasar el código shell como argumento como ya se ha descrito. Incluso si PowerShell fuera eliminado de la lista de la lista blanca de aplicaciones por defecto un atacante puede ejecutar el código shell

directamente dentro de la aplicación Java. Más detalles sobre este enfoque se pueden encontrar en ([xiii]) y ([xiv]).

3.2.3 Macros de Office

Las macros de Office son una vieja técnica utilizada en ataques de ingeniería social para infectar un sistema. Si la víctima permite macros en un archivo malicioso de Word o Excel, el código VBA comenzará a ejecutarse. *McAfee Application Control* no protege contra esto, sin embargo, por lo general el código sólo agrega un binario al sistema de archivos y luego lo inicia. Msfencode es un ejemplo de esto, donde sólo un binario se agrega al sistema de archivos y lo inicia después. Debido a que el binario no estaba en la lista blanca de aplicaciones, su ejecución sería bloqueada. Para superar esto, una de las aplicaciones de la lista blanca, por ejemplo, PowerShell, se puede iniciar en su lugar. Por otra parte, se pueden ejecutar directamente el código shell dentro de la aplicación de Office. El código fuente de este tipo de ataque puede ser consultado en ([xv]), sin embargo este código sólo funciona en sistemas de 32 bits. En un sistema de 64 bits de Windows la declaración de las funciones debe adaptarse mediante el atributo PtrSafe y todas las ocurrencias de tipo long debe ser reemplazado con el tipo LongPtr; consulte ([xvi]) para obtener más información.

3.2.4 Explotación de corrupción de Memoria

Incluso si todas las cuestiones antes mencionadas no existieran, un atacante aún puede explotar una vulnerabilidad de corrupción de memoria en una de las aplicaciones de la lista blanca (por ejemplo, en el navegador, una de las aplicaciones de office, el lector de PDF, etc.). En tal un caso shellcode puede ser ejecutado en nombre del proceso atacado.

McAfee Application Control intenta prevenir dichos ataques mediante la aplicación de protección de desbordamiento de búfer.

Las siguientes citas fueron tomadas de la página principal del producto:

"Además, evita que las aplicaciones enumeradas en la lista blanca puedan ser explotadas a través de ataques de desbordamiento de búfer de memoria (buffer overflow) en sistemas Windows de 32 y 64 bits." [i]

"Ventajas principales: Protege contra ataques de día cero y APT sin actualizaciones de firmas."[ii]

"Los programas agregados a la lista blanca que pudieran contener algunas vulnerabilidades inherentes no podrán ser explotados a través de un desbordamiento de búfer. "[iii]

La explotación en sistemas Windows XP

Para verificar las declaraciones anteriores, múltiples exploits fueron probados en diferentes sistemas Windows que ejecutan *McAfee Application Control*. Después de solidificar y reiniciar el sistema, las características “MP” (protección de memoria) y "MP-CASP" fueron habilitados por defecto, mp-vasr está desactivado en Windows XP, en Windows 7 está habilitado pero hablaremos más sobre esto más adelante. A pesar de que Windows XP esta fuera de soporte, y no debería utilizarse, la seguridad adicional proporcionada por *McAfee Application Control* fue verificada. Esto se hizo por dos razones. En primer lugar, Windows XP no incluye la protección de memoria que son distribuidas en la actualidad por defecto en los sistemas operativos más recientes y, por tanto, la protección de *McAfee Application Control* se puede estudiar mejor. La segunda razón fue que los sistemas Windows XP son utilizados hoy en día en entornos SCADA y por lo tanto la seguridad de estos sistemas sigue siendo importante.

De 30 vulnerabilidades analizadas de diferentes aplicaciones, *McAfee Application Control* inicialmente bloqueo 28 de ellas. Sólo un exploit de Firefox (Vulnerabilidad .reduceRight () CVE-2011-2371) y un exploit VLC (vulnerabilidad .s3m, CVE-2011-1574) fueron capaces de saltarse las protecciones sin ninguna modificación. Estos dos exploits fueron desarrollados para atacar sistemas más nuevos y por lo tanto contienen una cadena ROP (programación orientada a retorno) para deshabilitar el DEP (Data Execution Prevention), mientras que los otros no contienen una cadena ROP.

Las protecciones de memoria de *McAfee Application Control* funcionan mediante la inyección de una DLL en todos los procesos en ejecución. Tan pronto como un proceso está depurado con *McAfee Application Control* habilitado distintas violaciones de acceso se produjeron dentro del depurador. Esto se debe al hecho de que *McAfee Application Control* modifica las opciones de protección de la memoria de la cabecera del kernel32.dll DOS/PE para que no sea legible. Tan pronto como una instrucción intenta leer la cabecera PE, una violación de acceso se activa y el código asignado por scinject.dll (en nombre de ntdll.dll) maneja la violación de acceso.

Si primero se carga la dirección de varias funciones y luego se llama a ZwQueryVirtualMemory para comprobar si la instrucción de activación pertenece a una página marcada como ejecutable. Si la comprobación pasa entonces llama a scinject.casp_inject_save_addr que copia la instrucción de disparo a un nuevo espacio de memoria asignado, a continuación, llama ZwContinue para continuar la ejecución de la instrucción y después de eso salta de regreso a scinject.dll en 0x66441120. Antes de ejecutar la instrucción, la función situada en 0x7c9B0060 se utiliza para hacer la cabecera PE leíble y la función situada en 0x664410f0 elimina posteriormente la protección de lectura.

Con este enfoque *McAfee Application Control* implementa una especie de “software DEP". Dado que la arquitectura/OS no es compatible con DEP tiene que implementar los controles dentro del software. Esto se consigue al forzar una excepción durante la ejecución del *shellcode*. Típicamente el shellcode tiene que analizar la cabecera PE de la kernel32.dll para recuperar la dirección de

diferentes funciones y por lo tanto este método puede ser utilizado para hacer una pausa en la ejecución del shellcode para aplicar comprobaciones adicionales. En este caso, sólo un simple control se realiza para verificar si la página asociada se ha marcado como ejecutable.

Para sortear la protección dos diferentes métodos pueden ser utilizados. O bien la cabecera PE puede ser marcada de nuevo como legible o el shellcode puede ser marcado como ejecutable. En ambas situaciones una función tal como `VirtualProtect()` o `VirtualAlloc()` debe ser llamada. Como estas funciones pertenecen a kernel32.dll, sus direcciones no pueden obtenerse sin activar los controles. Para hacer frente a este problema, la función profunda ZwProtectVirtualMemory de ntdll.dll se puede utilizar en su lugar.

En la configuración anterior se pueden hacer tres observaciones importantes:

1. La memoria asignada por scinject.dll se marca como RWX y debido a que ASLR no es compatible con Windows XP la memoria siempre se encuentra en 0x7C9B0000. Esto significa que el shellcode puede copiarse a sí mismo a esta ubicación para eludir la protección (por ejemplo 0x7C9B0750 no es utilizada por el código y, por tanto, un buen objetivo).

2. Dentro de la localidad asignada, un puntero a ZwProtectVirtualMemory se almacena en 0x7C9B0695. El shellcode puede leer el dword almacenado en ese lugar para obtener un puntero a ZwProtectVirtualMemory y llamarlo para marcar su propia ubicación como ejecutable.

3. Las funciones almacenadas en 0x66441120 y 0x7c9B0060 pueden ser llamadas para establecer las opciones de protección para cualquier página. Por lo tanto, se les puede llamar para marcar el shellcode como ejecutable o el encabezado PE como leíble.

Sin embargo, todos estos métodos presuponen que *McAfee Application Control* está instalado en el sistema destino. Un mejor enfoque es obtener la dirección de ZwProtectVirtualMemory mediante la búsqueda dentro de la lista de módulos cargados y analizar la tabla de exportación para encontrarla. Esto también funciona de forma confiable en sistemas que no tienen instalado *McAfee Application Control*, por lo que garantiza la máxima fiabilidad en la ejecución del exploit. El shellcode para ese método fue desarrollado y mediante su uso, fue posible eludir las protecciones de memoria de *McAfee Application Control* en los 30 exploits.

Explotación en sistemas con Windows

Puesto que en Windows 7 las propiedades "mp-vasr" y el "-mp-vasr-forced-relocation" están adicionalmente habilitadas por defecto. Estas técnicas de mitigación correspondan a las técnicas de mitigación ASLR y obliga a iniciar ASLR desde el sistema operativo. Por lo tanto la seguridad global no se incrementa porque los sistemas actualizados ya incluyen ambas protecciones (ASLR forzada desde la actualización KB2639308 en Windows 7, ver ([v])).

La biblioteca scinject.dll inyectada ahora es compatible con las protecciones comunes como ASLR, DEP y SafeSEH. Sin embargo, cuando scinejct.dll asigna memoria para el código adicional, es

marcado con las propiedades de lectura, escritura y ejecutable como fue mencionado en la parte de Windows XP.

Estas secciones escribibles y ejecutables pueden ser objeto de abuso por atacantes mediante diversas formas para eludir la DEP, sin embargo, en la mayoría de los casos no hace una gran diferencia. Esto es debido al hecho de que la localización base de estas regiones es aleatorizada por ASLR. Un atacante primero tiene que de alguna manera eludir la ASLR y tan pronto como se eluda la ASLR no importa si una cadena ROP deba ser desarrollada para desactivar la DEP (por ejemplo, una llamada a `VirtualProtect()`, `VirtualAlloc()`, etc.) o para escribir el shellcode a uno de los lugares RWX. Sólo en algunas situaciones especiales de esta sección puede ejecutarse en las manos del atacante. Un ejemplo sería si las aplicaciones de seguridad adicionales están en su lugar, EMET por mencionar. EMET protege las llamadas a `VirtualProtect()` o `VirtualAlloc()` para detectar los ataques en curso. En tal caso, un atacante puede saltarse todas las protecciones de EMET abusando de los lugares RWX en lugar de llamar a una de estas funciones protegidas.

Explotación de sistemas Windows 8.1 de 64 bits

En los sistemas Windows 8.1 de 64 bits, las características de protección de memoria desaparecieron en *McAfee Application Control*. Esto es más probable debido a que el propio sistema operativo ya tiene implementadas todas estas protecciones y por lo tanto no se requiere más para inyectar una biblioteca en todos los procesos.

En resumen, la seguridad global desde la perspectiva de un desarrollador de exploits en un sistema con *McAfee Application Control* instalado es exactamente lo mismo que la seguridad de un sistema operativo estándar sin *McAfee Application Control* instalado. La seguridad puede ser incluso menor en algunas situaciones especiales si *McAfee Application Control* se instala y hace más sensible a abusos de la región de memoria RWX asignadas, como se mencionó anteriormente. Sólo si el sistema operativo subyacente está sin parchear y por lo tanto no tiene soporte para ASLR, la seguridad es mayor porque *McAfee Application Control* implementa esta característica.

Durante la investigación se pudo explotar varias vulnerabilidades de corrupción de memoria en sistemas Windows XP, Windows 7 y Windows 8.1 con *McAfee Application Control* instalado. La mayor parte de los exploits fueron desarrolladas mucho tiempo antes de la revisión de *McAfee Application Control* y ejecutados con éxito sin ninguna modificación.

Explotación de la aplicación ZIP instalada

En los capítulos anteriores, se describieron técnicas para explotar las vulnerabilidades de corrupción de memoria en sistemas con *McAfee Application Control*. Sin embargo, antes de eludir estas mitigaciones se debe encontrar una vulnerabilidad tal como la de corrupción de memoria. La identificación de un defecto de este tipo puede ser considerado como no demasiado difícil, ya que la aplicación se utiliza a menudo en entornos donde faltan los parches críticos (por ejemplo, los entornos SCADA debido a requisitos de confiabilidad). Esto demuestra por qué es tan importante

aplicar las actualizaciones a tiempo. Otro hecho a tener en cuenta es que las vulnerabilidades de corrupción de memoria local no son corregidas a menudo. Esto es debido al hecho de que la explotación de una vulnerabilidad típicamente no trae ninguna ventaja adicional para el atacante, ya que no puede hacer una escalada de privilegios. Sólo el código en el contexto de otra aplicación puede ser ejecutado, sin embargo, el mismo objetivo se puede lograr mediante el uso de una técnica de inyección de código. Por lo tanto este tipo de vulnerabilidades son a menudo marcados como poco importantes y no son remediadas. Estas vulnerabilidades son muy útiles si las listas blancas de aplicaciones deben ser evitadas, ya que son fáciles de encontrar y pueden ser utilizadas para lograr la ejecución de código completo.

Encontrar una vulnerabilidad, en gran medida depende del sistema de destino y, por tanto, la explotación es muy específica. Para superar este problema fue desarrollado un enfoque genérico. Para un método genérico el ejecutable que puede ser atacado debe estar presente en todos los sistemas. Esto restringe la selección a los ejecutables específicos de la aplicación. Por lo tanto se buscó la carpeta de instalación de *McAfee Application Control* por otros ejecutables. Se encontró una aplicación ZIP que data de 1999 que se instala por default por *McAfee Application Control*.

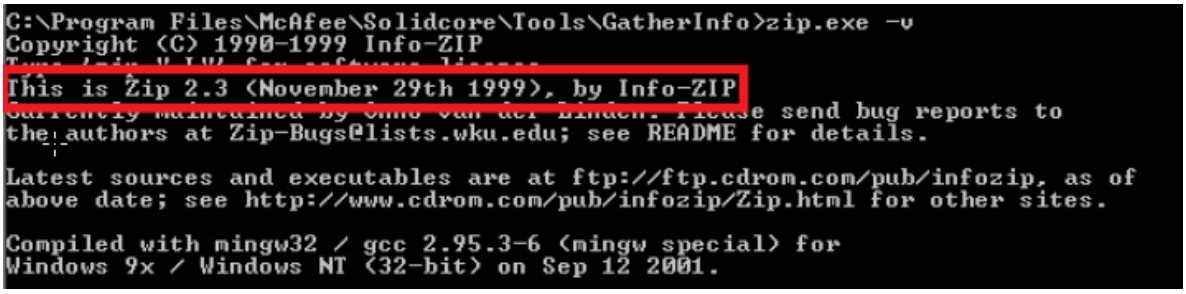

Ilustración 1: Aplicación ZIP fechada en 1999, se instala en todos los sistemas

Debido a que el ejecutable fue compilado en 2001 no es compatible con las funciones de seguridad típicas como DEP, ASLR, CFG o SafeSEH. Encontrarle una vulnerabilidad es bastante trivial, una simple llamada, como la siguiente conduce a un desbordamiento de búfer:

```
zip.exe -r a.zip aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa * reduced *
```

3.3 Elusión del control de cuentas de usuario (UAC- User Account Control)

A partir de Windows Vista, Microsoft introdujo el concepto de Control de Cuentas de usuario (UAC-*User Account Control*). La idea detrás de esta tecnología es la de limitar los privilegios de las aplicaciones, incluso si están iniciados por un usuario administrador. Si un administrador inicia sesión en el sistema dos diferentes tokens de privilegio son creadas - uno para un usuario normal y otro para el usuario administrador-. Normalmente las aplicaciones eran iniciadas usando el token de usuario normal. Si una aplicación requiere privilegios adicionales del token administrador un cuadro de diálogo especial debe ser confirmado por el usuario final.

Una técnica comentada en el siguiente capítulo requiere privilegios administrativos porque el código debe ser inyectado en los servicios privilegiados. Para aplicar esta técnica, UAC debe ser evitada de alguna manera. Por lo tanto, es importante discutir primero cómo el UAC se puede omitir en sistemas que tienen instalado *McAfee Application Control*. Tenga en cuenta que la técnica para eludir la protección contra escritura también funciona con cuentas con privilegios estándar. Sólo algunos casos especiales requieren privilegios de administrador, por ejemplo, la inyección de código en un servicio (esta técnica no es requerida para eludir las protecciones, sin embargo, es la más simple; las otras técnicas no requieren privilegios de administrador, pero son un poco más complejas).

El típico y más común tipo de bypass de UAC (que es por ejemplo utilizado por Meterpreter) no está funcionando porque *McAfee Application Control* impide la carga de bibliotecas que no están en la lista blanca. El siguiente texto da una muy breve reseña sobre los diferentes métodos de eludir el UAC. Este texto no da una visión completa, ni tiene todos los detalles discutidos en profundidad. En cambio, la idea básica detrás de las técnicas se describe para discutir por qué ciertos métodos están o no trabajando con *McAfee Application Control* habilitado.

Lo que comúnmente es compartido por muchos evasores de UAC es el hecho de que los procesos de auto-elevación son abusados. Microsoft dio un conjunto de programas de privilegios especiales que les permiten auto-elevar peticiones al UAC. Esto se hizo para minimizar el número de cuadros de dialogo de UAC solicitados al usuario final. Si un atacante logra ejecutar código en el contexto de un proceso de auto-elevación el UAC está anulado. Sin embargo, no es posible inyectar directamente código en un proceso de este tipo debido a los privilegios de administrador que faltan. Microsoft dio un segundo conjunto de aplicaciones de permisos especiales. Estas aplicaciones (por ejemplo, explorer.exe, notepad.exe, calc.exe, etc.) pueden interactuar con objetos COM auto-elevados para hacer acciones concretas como la creación de un nuevo directorio en una carpeta protegida sin solicitar un cuadro de diálogo de UAC. Dado que estas aplicaciones se ejecutan con los mismos privilegios que el código del atacante, es posible inyectar código en su espacio de proceso utilizando la técnica estándar con `VirtualAllocEx`, `WriteProcessMemory` y `CreateRemoteThread`. Después de eso, es posible crear archivos y directorios dentro de carpetas protegidas (por ejemplo, C:\Windows\system32) abusando de los objetos COM.

Para eludir completamente el UAC se requiere un paso final que inyecta código en uno de los procesos de auto-elevación. Para ello, se utiliza una técnica simple de inyección DLL. Si un ejecutable importa una biblioteca, una ruta de búsqueda específica se utiliza para encontrar la biblioteca (véase ([vi]) para más información). El código puede ser inyectado en un proceso autoelevado mediante la colocación de una biblioteca maliciosa muy temprano en la ruta de búsqueda para forzar su carga en lugar de la ruta válida. Esto no suele ser posible porque la ruta de búsqueda consiste en carpetas protegidas. Sin embargo, mediante la aplicación de la técnica discutida anteriormente, los archivos se pueden escribir en dichos lugares.

Se puede encontrar más información sobre tales derivaciones en ([vii]) y ([viii]). Especialmente ([viii]) es un gran recurso y código fuente para varias de las técnicas discutidas posteriormente y que allí se encuentran. Con el tiempo, más métodos fueron desarrollados, sin embargo, la técnica básica subyacente en la mayoría de los casos sigue siendo la misma. Por ejemplo, es posible abusar de la aplicación WUSA (*Windows Update Standalone Installer*) para extraer los archivos comprimidos en un directorio protegido en lugar de inyectar código en explorer.exe. El ejecutable auto-elevado atacado también cambia con diferentes métodos (por ejemplo, abusar de cliconfg.exe en lugar de sysprep.exe), pero todas estas técnicas comparten el concepto de que una biblioteca recién creado se ve obligado a ser cargado en el espacio de procesos auto- elevados.

Exactamente esto es impedido por *McAfee Application Control* porque la biblioteca de nueva creación no está incluida en la lista blanca del sistema y por lo tanto no se puede cargar. Sin embargo, existen otras técnicas para eludir el UAC. Un ejemplo es la técnica de redirección AppCompact / cuña. El siguiente texto ofrece una visión muy rápida sobre esta técnica.

Microsoft introdujo el concepto de cuñas con el Kit de herramientas de aplicaciones compatibles. La idea detrás de esta característica es proporcionar la capacidad de hacer que las aplicaciones antiguas fueran compatibles con los sistemas operativos más recientes. Por ejemplo en los documentos de Windows XP están almacenados en "`C:\documentos`", mientras que en Windows 7 los documentos se almacenan en "`C:\Usuarios\<nombre de usuario>\documentos`". Para hacer aplicaciones viejas compatibles con esta nueva estructura de archivos (y otros cambios) es posible instalar una cuña. Este complemento se encuentra entre la aplicación y las bibliotecas que enganchan el IAT (*Import Address Table* o tabla de direcciones de importación) y modifica los parámetros para adaptarse a la nueva estructura. No sólo es posible cambiar las trayectorias, también es posible inyectar bibliotecas o re-direccionar la ejecución a otro archivo ejecutable (Consulte más información sobre las cuñas las puede encontrar en ([ix]) y ([x])). Por ejemplo, una cuña se puede crear con una regla redirectEXE que redirige cmd.exe a Calc.exe. Tan pronto como un usuario intenta abrir cmd.exe la calculadora aparece en su lugar. La misma técnica se puede utilizar para redirigir uno de los ejecutables auto-elevados a otro ejecutable (por ejemplo PowerShell.exe). Debido a que el manifiesto (incluyendo los permisos de autoelevación) se utiliza desde el archivo original del proceso recién engendrado el cual tiene privilegios de administrador porque los cuados de diálogo UAC son auto-elevados. Desde este enfoque no se requiere la carga de archivos de nueva creación, sino que también funciona en sistemas con *McAfee Application*

Control habilitados. Esta derivación sólo es aplicable en sistemas x86 porque los sistemas Windows de 64 bits no son compatibles con la norma redirectEXE.

Otro método que funciona en sistemas Windows de 64 bits y 32 bits es desactivar permanentemente el UAC. La desventaja de este método es que requiere un reinicio del sistema. Fue utilizado por primera vez por el malware simba ([viii]) y abusa el objeto COM no documentado IsecurityEditor. Este objeto COM se puede acceder desde explorer.exe y se puede utilizar para establecer los permisos de acceso de una clave del registro. El ataque funciona mediante la inyección de código en explorer.exe que luego utiliza el objeto COM ISecurityEditor para hacer la siguiente ruta de registro escribible:

```
MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\policies\\system
```

Después de esto, las claves pueden ser creadas y modificadas para desactivar permanentemente el UAC.

Muchas otras técnicas de bypass UAC existen que también son muy propensas a trabajar con sistemas con *McAfee Application Control* habilitado (por ejemplo, véase ([xi]) y ([xii])). Porque las técnicas anteriormente discutidas ya trabajaban sin más esfuerzo que el gastado en buscar nuevos métodos.

4 Elusión de la protección de escritura y lectura

4.1 Inyección de código en procesos de actualización.

La característica principal de *McAfee Application Control* es evitar acciones de escritura en los archivos de la lista blanca. Esto es significativo ya que un atacante podría simplemente sobrescribir el contenido de una aplicación de la lista blanca para ejecutar su propio código.

McAfee Application Control está diseñado para almacenar el nombre y la ruta de todas las aplicaciones incluidas en la lista blanca en una base de datos. Tan pronto como se inicia un archivo (o una carga de librería) la ruta y el nombre se comparan con las entradas de base de datos de la lista blanca. No es necesario utilizar este enfoque de sobrecarga adicional (por ejemplo, cálculo del hash de un ejecutable) y por lo tanto el rendimiento del sistema no se ve afectado demasiado. Tenga en cuenta que la documentación menciona que los hashes también se almacenan en la base de datos (la cual se almacena encriptada en `<unidad>\Solidcore\scinv`). Durante las pruebas, el mecanismo de hash no pudo ser verificado, sin embargo, todavía fue posible que el hash se produjera, pero esto se hace de forma transparente para el usuario final y, por tanto, también para el atacante. Esto significa que un atacante simplemente puede sobrescribir el contenido de un archivo utilizando la siguiente técnica sin preocuparse por sumas de hash posiblemente incorrectas.

El enfoque descrito de almacenar la ruta absoluta junto con la protección de escritura tiene un segundo "beneficio". A menudo, las aplicaciones incluyen un proceso de actualización que asegura que la última versión de la aplicación está instalada. Si este actualizador descarga y reemplaza el ejecutable principal de la aplicación no sería capaz de ejecutarse porque los hashes cambiaron. Un administrador del sistema tendría que actualizar la lista blanca de nuevo para la aplicación. Un mejor procedimiento es automatizar el proceso de volver a actualizar las listas blancas para los ejecutables actualizados reduciendo así el trabajo de mantenimiento de los administradores y usuarios finales.

Para implementar esto, *McAfee Application Control* permite especificar una lista de procesos especiales de actualización. Estos procesos se identifican basándose únicamente en sus nombres (pero deben ser fortalecidos y marcados como ejecutables), y tienen permitido sobrescribir cualquier archivo en el sistema y por lo tanto pueden ser utilizados para eludir por completo la protección de escritura que ofrece *McAfee Application Control*.

Desde la perspectiva de un atacante no se requiere una elusión tal, sino que haga las cosas mucho más fáciles. Por ejemplo, como se describe en el capítulo 2, es posible lograr la ejecución de código. Sin embargo, utilizando el procedimiento discutido solamente el shellcode puede ser ejecutado. Con este código shell es posible hacer todo lo que un atacante quiera, pero escribirlo como shellcode le consume bastante tiempo. Típicamente un atacante ya tiene un montón de herramientas para varias tareas (por ejemplo, vertimiento de información del sistema, robo de credenciales, Iniciar un key-logger, comprometer aún más la red interna y así sucesivamente). Debido a la aplicación de listas blancas, estas herramientas no pueden ser iniciadas, pero tan

pronto como la protección contra escritura se elude, es posible sobrescribir una aplicación incluida en la lista blanca con el contenido de la herramienta requerida.

Una lista de los actualizadores predeterminados pre-configurados puede ser objeto de dumping utilizando el comando "`sadmin updaters list`". Consiste principalmente en diferentes actualizadores de productos de McAfee, Apache, Apple, Adobe Flash Player, Oracle Java, Mozilla Firefox, y muchos más.

El último paso es obligar a uno de estos procesos de actualización a sobrescribir el contenido de una aplicación permitida por la lista blanca. Para lograrlo, este código deberá estar en el contexto de un proceso de actualización que pueda ser controlado. Esto se puede hacer mediante el uso de una de las técnicas de inyección de código muy conocidas. El enfoque más sencillo es abrir un manejador para el proceso y luego asignar memoria para este proceso mediante la función `VirtualAllocEx()`. A continuación, utilizar un shellcode que sobrescriba un archivo incluido en la lista blanca (por ejemplo, un sencillo `CopyFileA-shellcode`) puede ser copiado en el espacio de memoria recién asignado para el proceso de destino mediante la función `WriteProcessMemory()`. Por último, el shellcode puede ser ejecutado en nombre del proceso de actualización al iniciar un nuevo hilo mediante el uso de la función `CreateRemoteThread()`.

El ataque descrito anteriormente puede ser remediado en caliente mediante la eliminación de todas las reglas de actualización y por lo tanto un atacante no puede migrar a un proceso de actualización. Otro problema sería si no existe un actualizador predeterminado instalado en el sistema. Dado que las reglas predeterminadas contienen a "`spoolsv.exe`" siempre hay un proceso estándar que se está ejecutando con privilegios de actualizador. Sin embargo, "`spoolsv.exe`" se está ejecutando como SYSTEM y, por tanto, el UAC debe ser eludido para inyectar el código en ese proceso. Diferentes técnicas para lograr esto ya fueron descritos en el capítulo 2.3.

Los actualizadores también tienen una segunda posibilidad. El comando "`sadmin read-protect -i C:\secret.txt`" se puede utilizar para hacer que el archivo secret.txt este protegido de lectura. Sin embargo, los actualizadores todavía pueden leer dichos archivos protegidos y por lo tanto la protección de lectura puede ser también fácilmente eludida.

4.2 Inyección de código en scsrvc.exe

Otro objetivo de la inyección es el archivo "scsrvc.exe", que es el principal servicio de *McAfee Application Control*. La inyección de código en este proceso tiene algunas ventajas especiales. Sin embargo, debido a que este proceso se ejecuta como SYSTEM, UAC debe ser primero eludida.

Después de la inyección de código en el proceso "scsrvc.exe" el archivo de contraseñas se puede leer. Este archivo se almacena en `C:\Archivos de programa\McAfee\Solidcore\passwd` y está protegido contra lectura. La lectura de este archivo está prohibida incluso para los procesos de actualización, pero a "scsrvc.exe" se le permite leer el archivo.

Por otra parte, "scsrvc.exe" tiene permiso para eliminar el archivo. Tan pronto como se elimina el archivo, es posible utilizar el comando "sadmin.exe" sin necesidad de proporcionar una contraseña y por lo tanto todas las reglas pueden ser modificadas o deshabilitadas.

También es posible modificar las reglas directamente al cambiar valores de registro del proceso `scsrvc.exe`. La configuración se almacena en la siguiente ruta:

```
HKEY_LOCAL_MACHINE \ SYSTEM \ CurrentControlSet \ Services \ Swin \ Parameters
```

Un atacante puede, por ejemplo, cambiar las reglas en "TrustedVolumeRules" para añadir un directorio desde el cual se puede iniciar cualquier ejecutable. Esto puede ser usado para eludir completamente la protección de escritura y ejecución de código de una manera fácil.

5 Vulnerabilidades del controlador del núcleo.

Se identificaron varias vulnerabilidades de kernel durante una prueba de fuzzing simple. *McAfee Application Control* carga el controlador del núcleo C:\Windows\system32\drivers\swin1.sys que tiene varias debilidades al manejar una de las siguientes códigos IOCTL:

- `0xb37031f0`
- `0xb37031f8`
- `0xb37031fc`
- `0xb370320c`
- `0xb3703200`
- `0xb3703204`
- `0xb3703208`
- `0xb3703214`

Estas vulnerabilidades pueden ser lanzadas como un usuario normal (no se requieren privilegios administrativos) y se pueden usar para bloquear el sistema con una pantalla azul. Es muy probable que estas vulnerabilidades se puedan utilizar para hacer un ataque de elevación de privilegios.

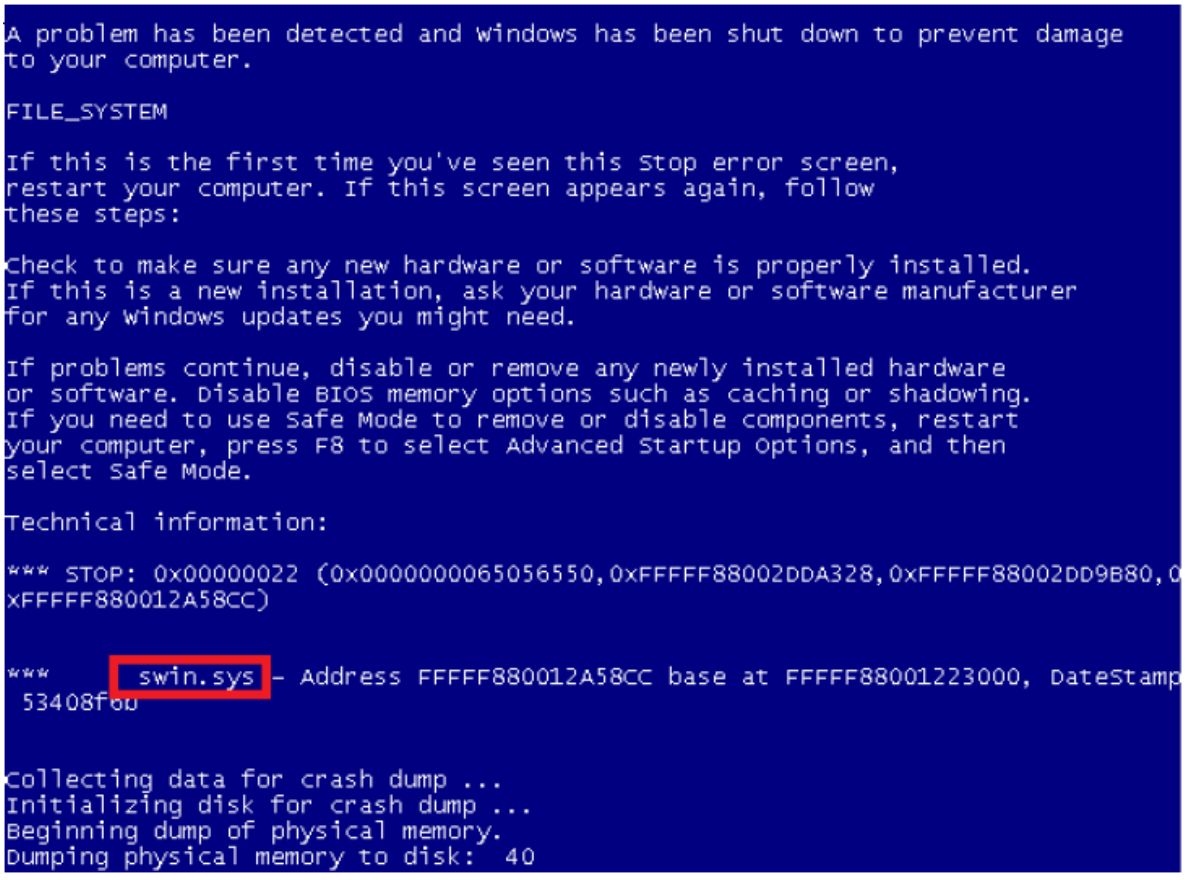

Ilustración 2. Un error en swin.sys provoca una pantalla azul

6. Conclusión

Como se demostró, es posible fácilmente saltarse las protecciones provistas por *McAfee Application Control.* En la mayoría de los casos la aplicación impidió con éxito un ataque en curso, sin embargo, mediante la aplicación de cambios mínimos siempre fue posible eludir los controles. Varios métodos se presentaron que se pueden utilizar para lograr la ejecución de código completo. También se discutieron las técnicas para eludir la protección de lectura y escritura, asi como el UAC. Al combinar todas estas técnicas (primero logar la ejecución de código básico, después escalar la ejecución a código completo, después de eludir el UAC y luego inyectar código en el servicio principal para crear un volumen de confianza) es posible iniciar cualquier aplicación que no esté incluido en la lista blanca. Por lo tanto, es posible que un atacante avanzado pueda eludir las protecciones implementadas. Mientras que *McAfee Application Control* ayuda a prevenir los ataques en masa, la seguridad adicional en el caso de los ataques dirigidos que se producen sobre todo en entornos SCADA, no se puede prevenir.

Por otra parte, varios puntos débiles fueron identificados. Estos van desde la asignación de regiones RWX en el espacio de memoria de todas las aplicaciones protegidas sobre las aplicaciones instaladas desde 1999 a las vulnerabilidades del kernel críticas que se pueden usar para bloquear por completo el sistema. Debido al área típica de aplicación de *McAfee Application Control* tales ataques sobre la confiabilidad de un sistema puede causarle problemas significativos. En la mayoría de los casos una herramienta tan sólo se utiliza si la seguridad de un sistema crítico debe aumentarse. Exactamente en estas situaciones la confiabilidad del sistema es muy importante y por lo tanto debe estar asegurado. Mediante la instalación de *McAfee Application Control* el usuario es engañado en la ponderación de una seguridad adicional engañosa, pero en realidad la aplicación muestra agujeros en la seguridad general del sistema.

Fuera de nuestra experiencia en *SEC Consult* considera necesario que las infraestructuras críticas instalen regularmente las nuevas versiones, utilizar sólo el software revisado por profesionales de la seguridad y aumentar aún más la conciencia de los usuarios finales con capacitaciones de seguridad. Para este tipo de sistemas no es suficiente confiar únicamente en una capa de seguridad tales como listas blancas de aplicaciones. Más bien, el valor subyacente del propio sistema debe ser aumentado. No vemos una razón para no usar listas blancas de aplicaciones si el software es seguro y no se noten los agujeros en la seguridad general del sistema, pero es importante entender que no sustituye a las medidas de seguridad robustas.

7 Acerca de

7.1 Acerca del autor

René Freingruber ha trabajado como consultor profesional de seguridad para SEC Consult por varios años, Conduce investigaciones en los campos de análisis de malware, ingeniería inversa y desarrollo de exploits. Durante su tesis de licenciatura desarrollo cientos de exploits para estudiar diferentes técnica de mitigación implementadas por modernos sistemas operativos, y como pueden ser eludidas por los atacantes. Con temas como la forma de eludir el toolkit ENET de Microsoft, ha dado pláticas en varias conferencias de gran seguridad incluida RuxCon, ToorCon, ZeroNights, DeepSec, NorthSec, IT-Secx and 31C3.

7.2 Acerca del Laboratorio de Vulnerabilidades

Los miembros del Laboratorio de vulnerabilidades de SEC Consult realizan investigación de seguridad en varias áreas de seguridad de información técnica. Los proyectos incluyen investigación de vulnerabilidades y el desarrollo de herramientas y metodologías de seguridad de frontera, y son financiados por socios como la Universidad Tecnológica de Viena. El laboratorio ha publicado vulnerabilidades de seguridad en muchos productos de software de perfil alto, y trabajos seleccionados han sido presentados en conferencias de alta seguridad como Blackhat y DeepSec.

Para más información, consulte http://www.sec-consult.com

7. Referencias

[i] http://www.mcafee.com/us/products/application-control.aspx
[ii] http://www.mcafee.com/us/resources/data-sheets/ds-application-control.pdf
[iii] http://www.mcafee.com/mx/resources/solution-briefs/sb-app-control-legacy-windows-xp.pdf
[iv] https://github.com/trustedsec/social-engineer-toolkit
[v] http://support.microsoft.com/en-us/kb/2639308
[vi] https://msdn.microsoft.com/en-us/library/windows/desktop/ms682586%28v=vs.85%29.aspx
[vii] http://www.pretentiousname.com/misc/win7_uac_whitelist2.html
[viii] https://github.com/hfiref0x/UACME
[ix] http://blogs.technet.com/b/askperf/archive/2011/06/17/demystifying-shims-or-using-the-app-compat-toolkit-to-make-your-old-stuff-work-with-your-new-stuff.aspx
[x] http://www.alex-ionescu.com/?p=39
[xi] https://code.google.com/p/google-security-research/issues/detail?id=118
[xii] https://code.google.com/p/google-security-research/issues/detail?id=222
[xiii] https://github.com/schierlm/JavaPayload
[xiv] BSidesCHS 2013 Session 02, Java Shellcode Execution, Ryan Wincey
[xv] http://blog.didierstevens.com/2009/05/06/shellcode-2-vbscript/
[xvi] https://msdn.microsoft.com/en-us/library/office/ee691831%28v=office.14%29.aspx