Fast16: The Cyberweapon That Predates Stuxnet by Five Years

hackingpassion.com

Bulls Eye, traducción de Aitor Saiz Lasheras.

Durante 21 años, un arma cibernética llamada **fast16** permaneció totalmente inadvertida. Esta no destruía equipos ni provocaba explosiones. Corrompía los cálculos matemáticos. Los científicos que realizaban simulaciones nucleares y de ingeniería obtenían resultados que parecían totalmente normales: todas las cifras cuadraban, todos los resultados tenían sentido y, sin embargo, todo estaba deliberadamente erróneo. Salió a la luz la semana pasada. Es cinco años anterior a Stuxnet.

Los investigadores de **SentinelOne**, **Vitaly Kamluk** y **Juan Andrés Guerrero-Saade**, presentaron el análisis completo de fast16 en **Black Hat Asia** la semana pasada. El binario principal de fast16 tiene una marca de tiempo de compilación del **30 de agosto de 2005**. La infraestructura C&C de Stuxnet se estableció en noviembre de ese mismo año.

La mayoría de los expertos en seguridad conocen **Stuxnet** como el gusano que destruyó las centrifugadoras de la instalación nuclear iraní de Natanz alrededor de 2010, llevándolas más allá de sus límites mecánicos mientras engañaba al software de monitorización sobre lo que estaba sucediendo. Fue el primer arma cibernética conocida diseñada para causar destrucción física, y durante años se consideró el punto de partida de toda esta era. Fast16 fue el primero, y durante mucho tiempo fue el único.

Kamluk partió de una corazonada. Se había dado cuenta de que las familias de malware patrocinadas por Estados más sofisticadas que conocía compartían todas un mismo rasgo técnico: cada una tenía incorporado un pequeño motor de scripts llamado **Lua**. Lua funciona como un mando a distancia para el malware: permite a los operadores cambiar lo que hace el implante mientras ya se está ejecutando en una máquina objetivo, sin necesidad de enviar un archivo completamente nuevo. Quería saber si algo más antiguo había hecho lo mismo antes, y se puso a buscar en colecciones antiguas.

Lo que encontró fue un archivo en **VirusTotal** llamado `svcmgmt.exe`, subido en octubre de 2016 y marcado por casi nadie. Parecía un aburrido envoltorio de servicio de Windows de la era XP. Pero en su interior había una **máquina virtual Lua 5.0** incrustada, código byte cifrado y una ruta que apuntaba a un controlador del núcleo llamado `fast16.sys`. Eso convierte a fast16 en el **primer malware de Windows conocido en incorporar un motor Lua**, precediendo al siguiente ejemplo conocido por tres años completos.

Hay algo más que confirma la cronología. Fast16 solo se ejecuta en **procesadores de un solo núcleo**, fabricados en una época en la que la mayoría de las máquinas aún funcionaban con un solo núcleo y los multinúcleos apenas empezaban a llegar al mercado.

El *framework* se ejecuta en tres capas. La capa exterior es `svcmgmt.exe`, un portador que se comporta de forma diferente según cómo se inicie. Si se le pasa `-p`, se propaga por la red. Si se le pasa `-i`, se instala como un servicio de Windows y ejecuta la carga útil integrada. Si se le pasa `-r`, ejecuta la carga útil sin instalarse. Dentro del portador hay tres elementos almacenados de forma cifrada: el código de bytes Lua que gestiona la lógica operativa, una DLL que se engancha al

sistema de conexión por módem y VPN de Windows, y el propio `fast16.sys`. Vale la pena examinar más de cerca esa DLL. Cada vez que un equipo se conecta a una red remota, escribe los detalles de la conexión en una *pipe* con nombre que los operadores pueden leer. Así, mientras `fast16.sys` corrompía los cálculos en el disco, la DLL mapeaba silenciosamente qué equipos se conectaban a qué redes, proporcionando a los operadores una imagen en tiempo real de la estructura interna de las instalaciones.

Parte de lo que hace interesante esa capa exterior es cómo se propaga. El mecanismo funciona como un camión de reparto con múltiples compartimentos. Cada compartimento, denominado **wormlet**, puede transportar una carga útil diferente con un propósito distinto. El portador se copia a sí mismo a través de recursos compartidos de red con autenticación débil y se inicia como servicio en cada máquina a la que llega. SentinelOne denomina a esto **arquitectura de munición de racimo**. En la muestra recuperada, solo uno de esos compartimentos está lleno. Los demás están vacíos, lo que plantea una pregunta obvia sobre si existen otras variantes con cargas útiles diferentes que nadie haya encontrado aún.

Antes de que nada de esto se ejecute, el código comprueba el registro en busca de software de seguridad. Si encuentra **Kaspersky**, **Symantec**, **McAfee**, **F-Secure**, **Zone Labs** o una docena de otros productos que eran comunes a mediados de la década de 2000, se detiene inmediatamente. Esa lista no era una suposición. Refleja exactamente lo que los operadores esperaban encontrar en las máquinas que buscaban.

La segunda capa es el gusano, que se propaga utilizando el control de servicios estándar de Windows y las API de intercambio de archivos, nada personalizado. Se basa en contraseñas de administrador débiles o predeterminadas en recursos compartidos de red para desplazarse de un equipo a otro, lo cual era una suposición realista para muchas redes internas en 2005.

La tercera capa es `fast16.sys`, y aquí es donde realmente se produce el sabotaje. Un **controlador del núcleo** se encuentra en lo más profundo del sistema operativo, por debajo de donde el software antivirus suele buscar. `Fast16.sys` se carga al arrancar y se sitúa por encima de todas las capas de almacenamiento del equipo: NTFS, FAT, el sistema de archivos de red. Lo primero que hace al cargarse es desactivar el **Prefetcher de Windows**, un sistema que normalmente almacena en caché los archivos de uso frecuente para acelerar el proceso. Con eso desactivado, cada archivo que se lee tiene que pasar por toda la pila de almacenamiento y por el controlador. Todo lo que se lee del disco pasa primero por él. Y luego simplemente espera. No ocurre nada hasta que alguien inicia sesión y se inicia el escritorio. Solo entonces comienza a vigilar cada ejecutable que se abre.

El controlador no persigue todos los archivos que ve. Busca software compilado con una herramienta específica: el **compilador Intel C++** deja una pequeña cadena identificativa en cada ejecutable que produce, justo después del encabezado de la última sección. Los desarrolladores sabían exactamente qué compilador utilizaban sus objetivos y construyeron la lógica de selección en torno a esa huella digital.

Por cada archivo que coincide, el controlador intercepta las **rutinas de cálculo en coma flotante** en memoria mientras el archivo se lee del disco. Los cálculos en coma flotante son las matemáticas que hay detrás de las simulaciones de precisión, del tipo que te dicen si el diseño de un puente

aguantará la carga, o si el detonador de un explosivo detonará en el momento adecuado. El controlador modifica esas rutinas utilizando **101 reglas de coincidencia de patrones**, inyecta un bloque de instrucciones FPU que cambia silenciosamente los valores en las matrices de cálculo internas y permite que el archivo se cargue como si nada hubiera pasado. El código original en el disco permanece intacto. El software funciona con normalidad. Los resultados son erróneos.

La aplicación de esas 101 reglas al software de esa época señaló tres objetivos específicos.

El primero es **LS-DYNA 970**, un paquete de simulación utilizado para modelar explosiones, fallos estructurales e impactos a alta velocidad. El Instituto para la Ciencia y la Seguridad Internacional publicó en septiembre de 2024 una revisión de 157 artículos académicos que mostraban que investigadores iraníes utilizaron LS-DYNA en trabajos relacionados con el desarrollo de armas nucleares, concretamente para modelar los detonadores explosivos que inician la detonación de las ojivas. Si fast16 se ejecutaba en esas máquinas, los científicos no tenían forma de saber que sus resultados eran erróneos. Cada decisión de diseño basada en esos números se construyó sobre resultados corruptos.

El segundo objetivo es **PKPM**, y esta es la parte que la mayoría de la cobertura pasa por alto por completo. PKPM es el software de ingeniería estructural dominante en China, desarrollado por la Universidad de Tsinghua y la Academia China de Investigación de la Construcción, y utilizado en proyectos de construcción chinos durante más de tres décadas. Lo que lo convierte en algo más que una herramienta estándar de ingeniería civil es que PKPM también se utiliza para el **análisis estructural sísmico de instalaciones de reactores nucleares**. Un artículo de 2024 publicado en *Advances in Civil Engineering* documenta el uso de PKPM para modelar el comportamiento estructural del reactor de sales fundidas de torio TMSR-LF1 de China en condiciones sísmicas. SentinelOne no puede confirmar quién era el objetivo de PKPM ni dónde se ejecutaba fast16. Si esto iba dirigido a un segundo país objetivo es una cuestión que queda abierta.

El tercero es **MOHID**, una plataforma de modelización hidrológica de código abierto desarrollada en el Instituto Superior Técnico de Lisboa. Se utiliza para modelizar sistemas hidrológicos costeros, el transporte de sedimentos, el comportamiento de presas y el impacto medioambiental de grandes proyectos de construcción cerca del agua. SentinelOne afirma abiertamente que no puede identificar cuál habría sido el efecto de sabotaje previsto sobre este software, y está pidiendo ayuda a la comunidad investigadora. El motivo por el que fue objetivo podría estar aún en una muestra que nadie ha encontrado todavía.

La conexión con la **NSA** proviene de una lista incluida en la filtración de **ShadowBrokers**. En abril de 2017, ShadowBrokers publicó una gran colección de materiales que, según la opinión generalizada, procedían del **Equation Group** de la NSA. En su interior había un archivo llamado `drv_list.txt`, básicamente una lista de «no tocar» para los operadores. Cuando un equipo llegaba a una máquina objetivo y encontraba un controlador de esa lista, esta les indicaba si dicho controlador pertenecía a una operación amiga y si debían dejarlo en paz. Era un sistema para garantizar que los diferentes equipos no interfirieran accidentalmente en el trabajo de los demás.

La mayoría de las entradas de esa lista incluían una nota para actuar con cautela o retirarse. Fast16 recibió algo diferente:

```
1 "fast16","*** NOTHING TO SEE HERE - CARRY ON ***"
```

Es como si un operador le dijera a otro: «Si encuentras este controlador, no lo toques, es nuestro». Los investigadores del **CrySyS Lab** detectaron esta entrada cuando analizaron el volcado de ShadowBrokers en 2018 y no tenían ninguna muestra con la que relacionarla. Ocho años después, ya hay una. Los materiales de ShadowBrokers se relacionan ampliamente con el Equation Group de la NSA, aunque, como ocurre con todas las filtraciones de inteligencia, desde fuera no se conoce el panorama completo.

Hay otra cosa que destaca en el código. Los archivos fuente contienen marcadores de control de versiones que provienen de entornos de desarrollo Unix de los años 70 y 80, mucho antes de que existiera Windows. Tienen este aspecto:

```
1 @(#)par.h $Revision: 1.3 $
```

Ese tipo de notación, denominada **SCCS/RCS**, es como encontrar un teléfono de disco en una oficina moderna. Nadie la utiliza en el código del núcleo de Windows de 2005, a menos que su experiencia en programación se remonte a décadas atrás, a entornos informáticos gubernamentales y militares de una época completamente diferente. No se trata de hackers de fin de semana ni de autónomos. Se trata de un programa institucional de larga trayectoria creado por personas que han desarrollado su carrera en ámbitos muy específicos.

Lo que empeora aún más todo esto es el historial de detección. `Svcmgmt.exe` se subió a VirusTotal en octubre de 2016 y permaneció allí durante casi una década, completamente a la vista de todos. **Un motor antivirus de entre unos setenta lo marcó**, de forma débil, como generalmente malicioso. Un portador autopropagable que implementa un controlador del núcleo a nivel de arranque con un motor de parches de punto flotante en memoria había estado en una base de datos pública durante nueve años, casi invisible para todos los escáneres que lo analizaron.

Durante su análisis, Kamluk utilizó **Claude** para ayudarle a analizar fast16 y redactar los resultados. En un momento dado, la IA falló repetidamente a la hora de terminar un informe que le había pedido que escribiera. Cuando le preguntó por qué, Claude produjo párrafos de autocrítica, instándose a sí misma a terminarlo de una vez. Finalmente lo hizo, y concluyó que quienquiera que hubiera creado fast16 tenía un conocimiento profundo del software objetivo y que el **sabotaje industrial** era la intención más probable. Un malware de hace 21 años dejó perpleja a una IA moderna el tiempo suficiente como para hacerla reflexionar sobre sus propias limitaciones.

Si trabajas con software de simulación antiguo, en particular con versiones antiguas de LS-DYNA o PKPM de mediados de la década de 2000, SentinelOne ya ha notificado directamente a los proveedores. La medida recomendada es verificar los resultados de los cálculos críticos con un **sistema completamente independiente situado fuera de cualquier red potencialmente afectada**. Si fast16 se propagara por toda una instalación y afectara a todas las estaciones de trabajo, un cálculo comparativo realizado dentro de esa misma red produciría el mismo resultado erróneo. Una máquina completamente fuera de ese entorno no lo haría.

Los indicadores a tener en cuenta:

1. → Driver: `fast16.sys` | MD5: `0ff6abe0252d4f37a196a1231fae5f26`
2. → Carrier: `svcmgmt.exe` | MD5: `dbe51eabebf9d4ef9581ef99844a2944`
3. → Notification DLL: `svcmgmt.dll` | MD5: `410eddfc19de44249897986ecc8ac449`
4. → Named pipe used for reporting: `\\.\pipe\p577`
5. → Device objects created by the driver: `\Device\fast16` and `\??\fast16`
6. → Custom DeviceType value in the driver: `0xA57C`
7. → Service name installed by the carrier: `SvcMgmt`

SentinelOne publicó en su artículo de investigación las **reglas YARA** completas para la búsqueda tanto del portador como del controlador.

Si algo tan sofisticado pasó 21 años sin ser detectado, permaneciendo en VirusTotal durante casi una década sin que prácticamente ningún motor antivirus lo detectara, ¿qué más habrá en colecciones similares en este momento, esperando a que alguien plantee una pregunta diferente? Probablemente más de lo que nadie querría saber.

Fast16 se instalaba como un servicio de Windows, se propagaba a través de recursos compartidos de red, se ejecutaba como un controlador del núcleo y permanecía completamente oculto mientras funcionaba. Los conceptos que hay detrás de ello —explotación, post-explotación, persistencia, escalada de privilegios y movimiento por la red sin ser detectado— son exactamente lo que mi curso de hacking ético cubre paso a paso:

- → **Apúntate a mi curso completo de hacking ético**

El hacking no es un hobby, sino una forma de vida.