XZ Utils Backdoor

Alessandro Iena

Linux Day 2024

WWW.LINUXFM.ORG

alessandro@linuxday2024:/root $ whoami && \
alias FeeDz='Studente@UNICAM\ Informatica'

Contenuti

01

XZ Utils

Cos’è XZ Utils?

- xz utils è una suite di strumenti open source per la compressione dati
- Utiliza principalmente l’algoritmo di compressione LZMA (Lempel-Ziv-Markov chain algorithm)

A cosa serve ?

- Il principale scopo di xz utils è la compressione e decompressione di file. È utilizzato per ridurre la dimensione dei file mantenendo un buon livello di efficienza
- Spesso usato in sistemi Unix e Linux per comprimere archivi di grandi dimensioni, inclusi i pacchetti software
- Molti sistemi operativi e gestori di pacchetti (ad esempio, Debian, Fedora) utilizzano xz utils per comprimere pacchetti software

https://github.com/tukaani-project/xz

Esempio di utilizzo XZ

- Quando una distribuzione Linux (come Debian o Fedora) rilascia aggiornamenti o nuovi pacchetti software, questi vengono compressi
- Un pacchetto software può essere creato in formato tar.xz, che combina l'archiviazione di più file in un archivio .tar e poi la compressione usando xz utils

esempio:

```
tar -cf pacchetto.tar /path/to/files
xz -z pacchetto.tar
```

- In questo esempio, il comando tar crea un archivio di file e successivamente xz -z comprime l'archivio in un file .tar.xz

Grazie all'algoritmo LZMA utilizzato da xz utils, la compressione è più efficiente rispetto ad altri metodi come gzip o bzip2

XZ è largamente utilizzato in linux

- rende i files più piccoli anche del 30-40 % rispetto a gzip e bzip2
- è molto veloce nell’apertura dei file
- è usato in progetti molto grandi e in molte distribuzioni Linux
- è la prima scelta secondo la community open source

02

Backdoor

Cos’è una Backdoor?

- Una backdoor è un metodo segreto per bypassare i normali controlli di sicurezza ed entrare in un sistema, applicazione o rete
- Consente a un utente non autorizzato di ottenere accesso remoto o privilegi di amministratore senza autenticazione
- Una backdoor può essere inserita volontariamente (da sviluppatori malintenzionati o governativi) o introdotta tramite vulnerabilità sfruttabili
- Una volta attivata, permette l'accesso senza rilevamento, consentendo l'esecuzione di comandi, il furto di dati, o la modifica di sistemi compromessi

Software backdoor: Codice nascosto all'interno di software che consente accesso non autorizzato. (tipologia della backdoor utilizzata in XZ Utils)

Perchè usare una Backdoor?

- **Attacchi malevoli:** Gli hacker possono inserire backdoor per mantenere il controllo su un sistema o rubare informazioni sensibili
- **Scopi di sorveglianza:** Alcuni governi o agenzie di intelligence possono inserire backdoor per monitorare le attività degli utenti

03

Scoperta della backdoor in XZ Utils

Scoperta della Backdoor in XZ utils

- 29 marzo 2024 Andres Freund (sviluppatore Microsoft) nota un elevato utilizzo della CPU durante le connessioni SSH e decide di fare debugging per risolvere il problema

- Freund nota un uso anomalo della CPU in SSHD anche dopo tentativi di accesso falliti, quando il server non avrebbe dovuto essere sotto pressione

- **SSH (Secure Shell)** permette accessi sicuri a computer remoti.
- **SSHD (SSH Daemon)** è il servizio sul server che gestisce queste connessioni.
  Compromettere SSHD significa violare la "serratura" che protegge l'accesso ai server

Scoperta della Backdoor in XZ utils

- Un'analisi più approfondita ha rivelato una connessione insolita tra SSHD è una libreria condivisa (Dynamic Shared Object), caricata dinamicamente dal server
- Questo legame, inaspettato e sospetto, ha sollevato dubbi sulla presenza di una possibile compromissione
- La librería si agganciava a SSHD, creando una backdoor che permetteva l'accesso remoto non autorizzato a un numero potenzialmente enorme di server e sistemi

```
→  ~ ldd /usr/sbin/sshd | grep lzma
        liblzma.so.5 => /lib/x86_64-linux-gnu/liblzma.so.5 (0x00007916ece88000)
```

Scoperta della Backdoor in XZ utils

- La versione compromessa di XZ modificava il comportamento del demone SSH di OpenSSH sfruttando la libreria systemd, consentendo all'attaccante di eseguire codice arbitrario (RCE)

- Qualsiasi macchina con la versione di XZ vulnerabile con SSH esposto su internet poteva essere potenzialmente a rischio

Distribuzioni affette:

- Fedora Rawhide
- Fedora 41, 40 beta
- openSUSE Tumbleweed
- Debian unstable versions, testing, experimentals
- Versioni Kali Linux aggiornate fra il 26 e il 30 marzo
- Alcune VM basate su Arch
- Alpine Edge (development)

CVE-2024-3094 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Description

Malicious code was discovered in the upstream tarballs of xz, starting with version 5.6.0. Through a series of complex obfuscations, the liblzma build process extracts a prebuilt object file from a disguised test file existing in the source code, which is then used to modify specific functions in the liblzma code. This results in a modified liblzma library that can be used by any software linked against this library, intercepting and modifying the data interaction with this library.

Metrics

CVSS Version 4.0 | CVSS Version 3.x | CVSS Version 2.0

NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

CVSS 3.x Severity and Vector Strings:

CNA: Red Hat, Inc. **Base Score:** 10.0 CRITICAL **Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

https://nvd.nist.gov/vuln/detail/CVE-2024-3094

04

Pressioni sociali

Com’è potuto accadere?

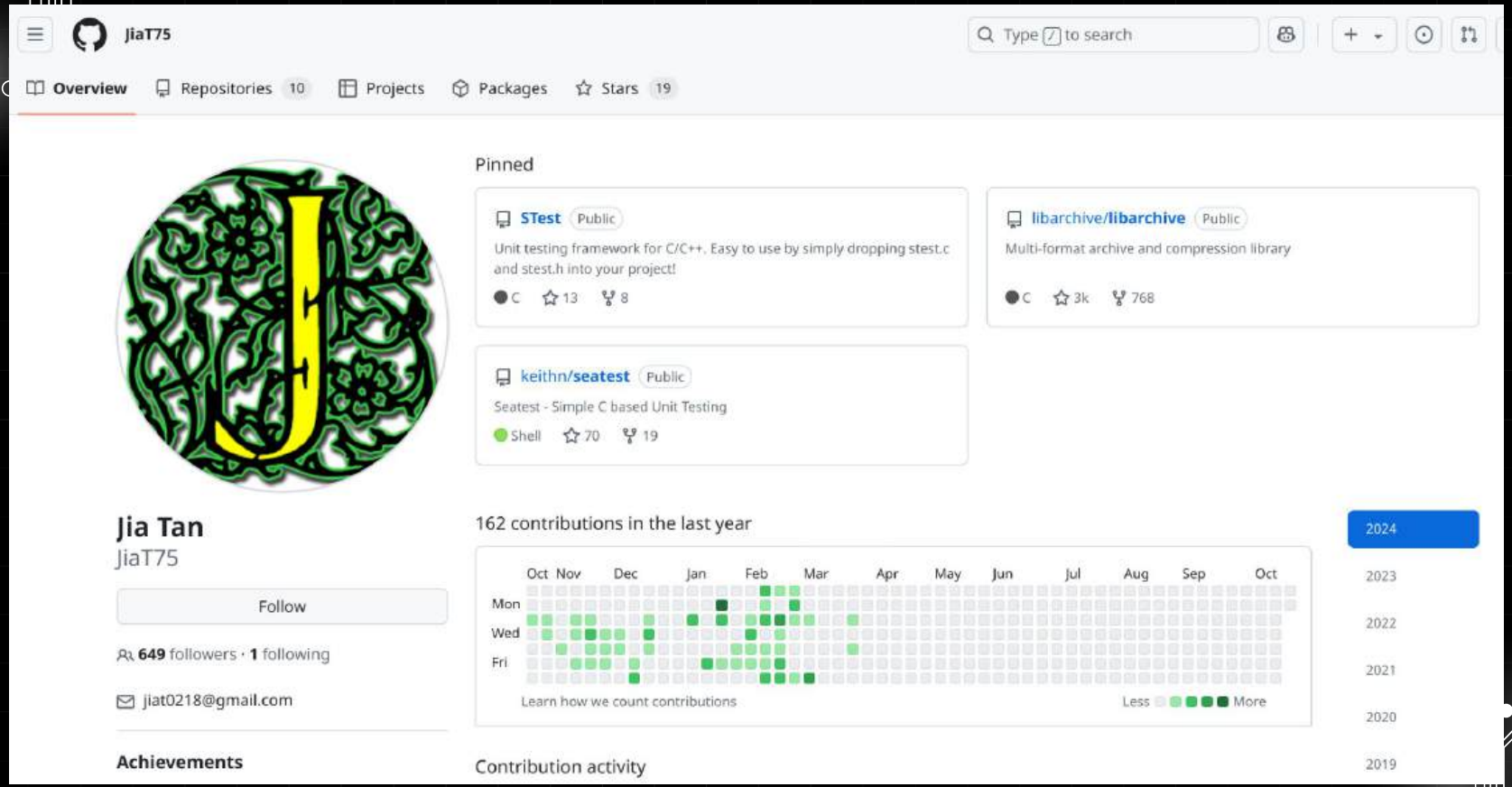
compromettere il demone SSH attraverso una libreria di compressione che viene linkata dinamicamente?

Facciamo un passo indietro

- **Lasse Collin**, con l'aiuto di altri, progetta il formato di file .xz utilizzando l'algoritmo di compressione LZMA
- Nel tempo, questo formato diventa ampiamente utilizzato per comprimere file tar, immagini del kernel Linux e molti altri utilizzi

Prima comparsa di Jia Tan

- il 29 ottobre 2021 Jia Tan entra in scena e invia la prima patch innocua alla mailing list xz-devel
- il 7 febbraio 2022 il founder Lasse Collin fa primo merge di un commit di Jia Tan chiamato *“liblzma: Add NULL checks to LZMA and LZMA2 properties encoders”*
- Jia Tan nel frattempo continua ad inviare altre patch innocue alla mailing list di xz


JiaT75
Type / to search
Overview
Repositories 10
Projects
Packages
Stars 19
Pinned
STest Public
Unit testing framework for C/C++. Easy to use by simply dropping stest.c and stest.h into your project!
C 13 8
libarchive/libarchive Public
Multi-format archive and compression library
C 3k 768
keithn/seatest Public
Seatest - Simple C based Unit Testing
Shell 70 19
Jia Tan
JiaT75
Follow
649 followers · 1 following
jiat0218@gmail.com
Achievements
162 contributions in the last year
Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct
Mon
Wed
Fri
Learn how we count contributions
Less More
2024
2023
2022
2021
2020
2019
Contribution activity

Le pressioni a Lasse Collin

- il 22 aprile 2022 "Jigar Kumar" invia la prima di alcune email in cui si lamenta del fatto che la patch di Jia Tan non sia stata ancora integrata.

"Le patch trascorrono anni su questa mailing list. Non c'è motivo di pensare che qualcosa arriverà presto." A questo punto, Lasse Collin ha già integrato quattro delle patch di Jia Tan, contrassegnate dalla dicitura "Grazie a Jia Tan" nel messaggio di commit.

- Lasse Collin risponde scusandosi per la lentezza e aggiunge:

“Jia Tan mi ha aiutato fuori dalla mailing list con XZ Utils e potrebbe avere un ruolo più importante in futuro, almeno con XZ Utils. È chiaro che le mie risorse sono troppo limitate (da qui le molte email in attesa di risposta), quindi qualcosa deve cambiare nel lungo termine.”

Le pressioni a Lasse Collin

- il 27 maggio 2022 Jigar Kumar invia un altra email di pressione al thread della patch

  “Oltre 1 mese e non siamo ancora vicini al merge. Non è una sorpresa.”

- il 7 luglio 2022 Jigar Kumar invia un ulteriore email di pressione al thread di Java

  “Non ci saranno progressi finché non ci sarà un nuovo mantainer. Anche XZ per C ha pochi commit. Dennis (rispondendo ad un altra persona), faresti meglio ad aspettare un nuovo manutentore o a fare un fork tu stesso. Inviare patch qui non ha più senso al giorno d'oggi. L'attuale manutentore ha perso interesse o non si interessa più di mantenere. È triste vedere una situazione del genere per una repository come questa.”

Le pressioni a Lasse Collin

- 8 agosto 2022: Lasse Collin risponde

  “Non ho perso interesse, ma la mia capacità di occuparmene è stata piuttosto limitata principalmente a causa di problemi di salute mentale a lungo termine, ma anche per altre ragioni. Recentemente ho lavorato un po' fuori dalla mailing list con Jia Tan su XZ Utils e forse avrà un ruolo più importante in futuro, vedremo. È anche importante tenere a mente che questo è un progetto hobbistico non retribuito.”

Le pressioni a Lasse Collin

- sempre Jigar Kumar

  “Con il vostro attuale ritmo, dubito seriamente di vedere il rilascio della versione 5.4.0 quest'anno. L'unico progresso da aprile sono stati piccoli cambiamenti al codice di test. Ignorate le molte patch che stanno marcendo su questa mailing list. In questo momento state soffocando la vostra repository. Perché aspettare la versione 5.4.0 per cambiare il manutentore? Perché ritardare ciò di cui il vostro repository ha bisogno?”

- Dennis Ens rispondendo a Lasse Collin

  “Mi dispiace per i tuoi problemi di salute mentale, ma è importante essere consapevoli dei propri limiti. Capisco che questo sia un progetto hobbistico per tutti i collaboratori, ma la comunità desidera di più. Perché non trasferire la manutenzione di XZ per C in modo da poter prestare maggiore attenzione a XZ per Java? O trasferire XZ per Java a qualcun altro per concentrarsi su XZ per C? Cercare di mantenere entrambi significa che nessuno dei due viene mantenuto bene.”

Considerazioni fino a questo punto

- Lasse Collin inizia a lavorare a stretto contatto con Jia Tan
- Molte delle email utilizzate per fare pressioni al Founder di XZ non sono mai apparse in Internet, nemmeno presenti in data breach
- Le email sembrano essere state create appositamente per spingere Lasse a concedere a Jia una posizione di rilievo

Considerazioni fino a questo punto

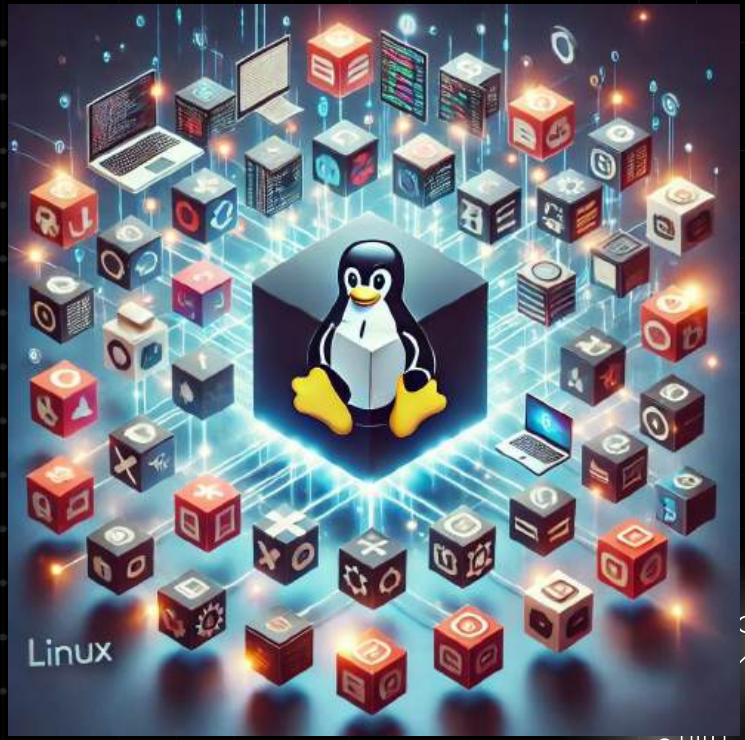
- Ma a chi interessa davvero una nuova feature di XZ Utils?
- Mmmmh una nuova feature di una libreria di compressione xD
- Lo spam delle email, la pressione sociale e il social engineering erano sicuramente componenti chiave dell'operazione

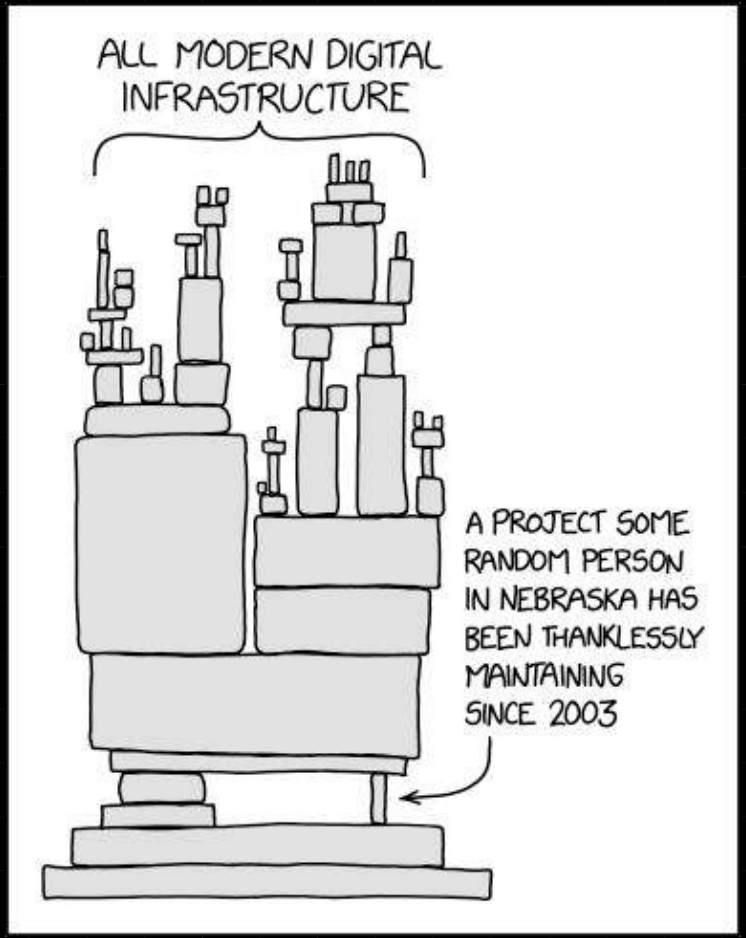
Manipolazione emotiva in XZ Utils

- Creazione di un senso di urgenza per accelerare decisioni
- Pressioni tramite email
- Possibile induzione a implementare nuove funzionalità senza revisione
- Rischio di decisioni affrettate da parte dei manutentori (Lasse Collin)

Linux e problemi di sicurezza

- Le distribuzioni Linux sono composte da un insieme di pacchetti software
- Possono provenire da fornitori terzi, comunità open source o progetti individuali
- Una vulnerabilità in uno di essi può esporre l'intero sistema a potenziali attacchi





https://www.explainxkcd.com/wiki/index.php/2347:_Dependency

05

Backdoor design & analisi

Backdoor Design

- La backdoor è molto complessa ed è stata ben offuscata
- Il processo di build di liblzma estrae un file oggetto precompilato
- Il file oggetto è nascosto all'interno di un file di test
- Il file oggetto viene utilizzato per modificare specifiche funzioni nel codice di liblzma

Risultato:

- libreria liblzma modificata

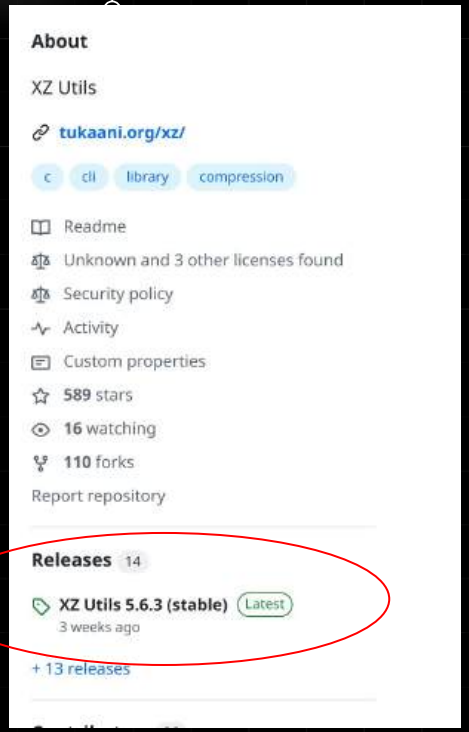
Esempio offuscamento

```c
#include <stdio.h>

int main() {
    int sum = 0;
    for (int i = 0; i < 10; i++) {
        if (i % 2 == 0) {
            sum += i;
        }
    }
    printf("Sum of even numbers: %d\n", sum);
    return 0;
}
```

```c
#include <stdio.h>

int cryptic_cond(int val_z, int mod_q, int res_true, int res_false) {
    return (val_z % mod_q == 0) ? res_true : res_false;
}

int main() {
    int x9 = 0;
    int a1 = 0;

    goto begin;

loop_label:
    x9 += cryptic_cond(a1, 2, a1, 0);
    a1++;
    if (a1 == 10) goto finish;

begin:
    goto loop_label;

finish:
    printf("Sum of even numbers: %d\n", x9);
}
```

Esempio: somma dei primi 10 numeri pari
versione “in chiaro” a sinistra, versione “offuscata” a destra

Backdoor Design



- La versione rilasciata (tarball) 5.6.0 / 5.6.1 non conteneva lo stesso codice presente nella repository GitHub
- Nei progetti scritti in C può capitare che le release differiscono per alcuni script di build o configurazione
- Nel caso di XZ il file `build-to-host.m4` è diverso rispetto a quello presente nella repository

Backdoor Design

23/02/2024 Jia Tian fa il merge di
`tests/files/bad-3-corrupt_lzma2.xz`

`tests/files/good-large_compressed.lzma`

- integra questi due binari malevoli all'interno della cartella dei tests

  "Questa directory contiene un insieme di file per testare la gestione dei file .xz, .lzma (LZMA_Alone) e .lz (lzip) nelle implementazioni dei decoder. Molti dei file sono stati creati a mano con un editor esadecimale, quindi non esiste un 'codice sorgente' migliore dei file stessi."

- Jia Tan ha sfruttato questa situazione per aggiungere alcuni file che non sarebbero stati esaminati con attenzione

Fase di build

- il primo stage della backdoor si attiva durante la fase di build
- `build-to-host.m4` estrae il prossimo stage dell'exploit dal file di test `tests/files/bad-3-corrupt_lzma2.xz` (apparentemente rinominato come file corrotto)
- il file una volta decodificato estrae uno script shell che avvia il secondo stage (https://lwn.net/Articles/967979/)
- lo stage due è nascosto nel file `tests/files/good-large_compressed.lzma` , in questo file si trovano altri binari che compongono la backdoor
- viene decompresso un file precompilato chiamato `liblzma_la-crc64-fast.o` che viene poi linkato a liblzma

Fase di build

Target 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
```

https://www.openwall.com/lists/oss-security/2024/03/29/4

Building with gcc and the gnu linker

```
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
```

Fase di collegamento (link time)

- Il file estratto è una libreria ELF a 64 bit che viene collegata durante la costruzione finale liblzma
- La libreria viene caricata anche in OpenSSH
- Viene sfruttata una funzione speciale di risoluzione dinamica dei simboli chiamata "funzioni indirette" (GNU indirect functions)
- Permette di selezionare quale versione di una funzione usare
- La backdoor fornisce le proprie versioni delle funzioni di checksum `crc32()` e `crc64()`, permettendo così all'exploit di manipolare il processo di collegamento

Fase di collegamento (link time)

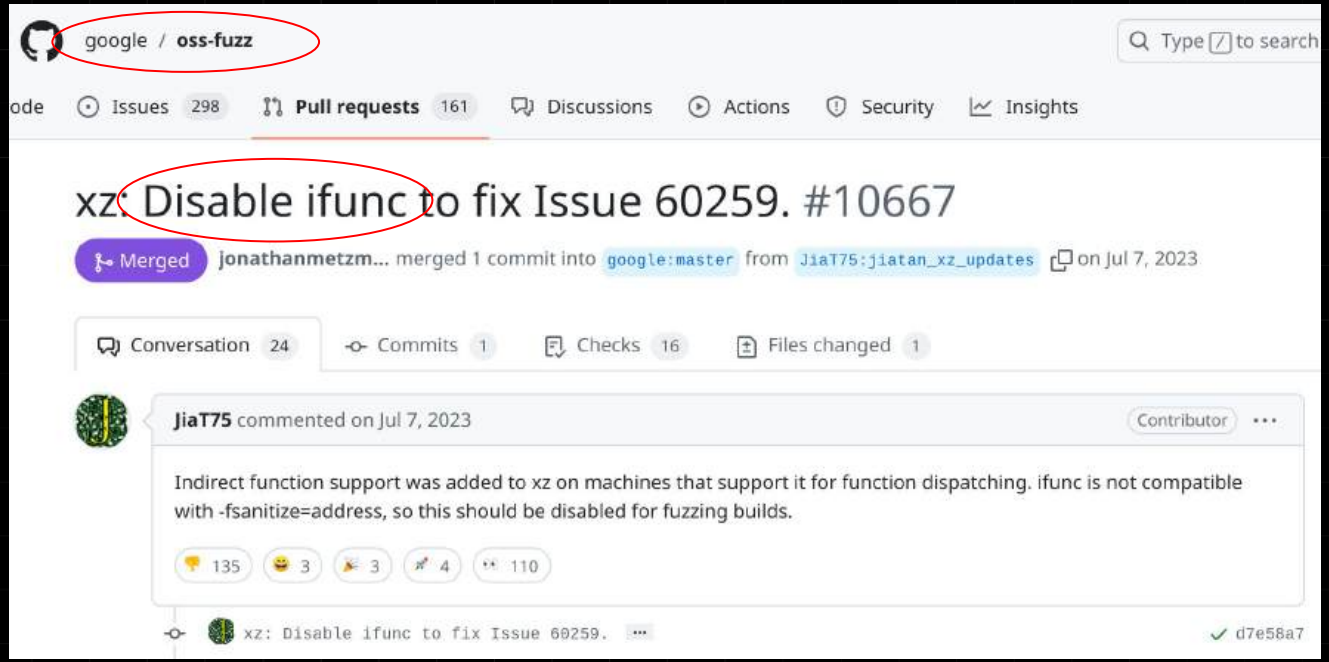
- La backdoor aggiunge un audit hook
- Meccanismo che intercetta il caricamento di funzioni specifiche, in questo caso la funzione `RSA_public_decrypt@got.plt`
- Questa funzione viene utilizzata da OpenSSH per verificare certificati RSA forniti dai client

IFUNC (Indirect function)

- Seleziona dinamicamente la versione di una funzione al momento dell'esecuzione
- La funzione IFUNC permette di avere diverse implementazioni di una stessa funzione

Nella backdoor IFUNC viene utilizzata per eseguire un'implementazione modificata di `RSA_public_decrypt`

Piccola digressione



https://github.com/google/oss-fuzz/pull/10667

jonathanmetzman approved these changes on Jul 7, 2023 View reviewed changes

jonathanmetzman left a comment • edited Contributor ···

tl;dr This patch did not prevent OSS-Fuzz from finding the backdoor, OSS-Fuzz is incapable of finding the backdoor, this patch is not suspicious, and OSS-Fuzz is not meant to find bugs when users do not want them found.

UPDATE:
There is no backdoor in this patch.
This repo provides a testing service to open source developers to help them find bugs in their code.
We allow developers to control 100% of what gets tested by our service. They own the code they submit to us, our reviews only mean we verified this is a project maintainer. Unfortunately we were told by the other XZ maintainer that "Jia Tan" was a maintainer.
If users don't want code tested, an automated tool like this can't stop them and was never intended to.

OSS-Fuzz is not impacted by the backdoor, because the backdoor never ran on OSS-Fuzz. In addition, OSS-Fuzz runs code in a sandboxed environment so malicious developers/projects cannot harm OSS-Fuzz.

UPDATE 2:
I wanted to highlight the half-dozen reasons why the backdoor would never have been discovered by OSS-Fuzz. It's totally implausible that it would have been discovered here.

1. This pull request fixed a build breakage caused by a compiler issue. The compiler issue was never fixed, so without this pull request, OSS-Fuzz would have never fuzzed an XZ version after June 2023, the backdoor was added in March 2024.
2. The XZ backdoor only built in the tarballs included in the 5.6.0 and 5.6.1 branches and not in the master branch of the git repo that OSS-Fuzz was cloning.
3. The XZ backdoor was only built when using build options for DEB or RPM packages. OSS-Fuzz was using neither: https://openwall.com/lists/oss-security/2024/03/29/4#:~:text=Running%20as%20part%20of%20a%20debian%20or%20RPM%20package%20build (this analysis is from the person who discovered the backdoor).
4. The XZ backdoor only ran when `argv[0]` was `/usr/sbin/sshd` This would never happen in OSS-Fuzz, only in a real environment https://openwall.com/lists/oss-security/2024/03/29/4#:~:text=argv%5B0%5D%20needs%20to%20be%20/usr/sbin/sshd

Il cambio della mail su OSS-Fuzz

Ricevere prima di Lasse Collin possibili mail su vuln / altre problematiche

Fase di esecuzione (run time)

- a run time la backdoor intercetta la risoluzione di `RSA_public_decrypt` sostituendola con la versione modificata malevola
- viene letto il certificato RSA fornito dall'attaccante
- viene verificato che sia firmato dalla chiave privata dell'attaccante
- a questo punto se i controlli vanno a buon fine, possono essere eseguiti comandi con i privilegi dell'utente che esegue sshd (di solito root)
- si ha quindi RCE

Requisiti per il funzionamento:

- variabile d’ ambiente TERM non impostata
- argv[0] deve essere /usr/sbin/sshd
- LD_DEBUG e LD_PROFILE non devono essere impostate
- LANG deve essere impostato
- ambienti di debug come rr e gdb possono essere rilevati (quindi XZ si comporterà in maniera “normale”)

Si è cercato di far girare l’exploit solo su sistemi di produzione, senza strumenti di debugging / monitoraggio attivi

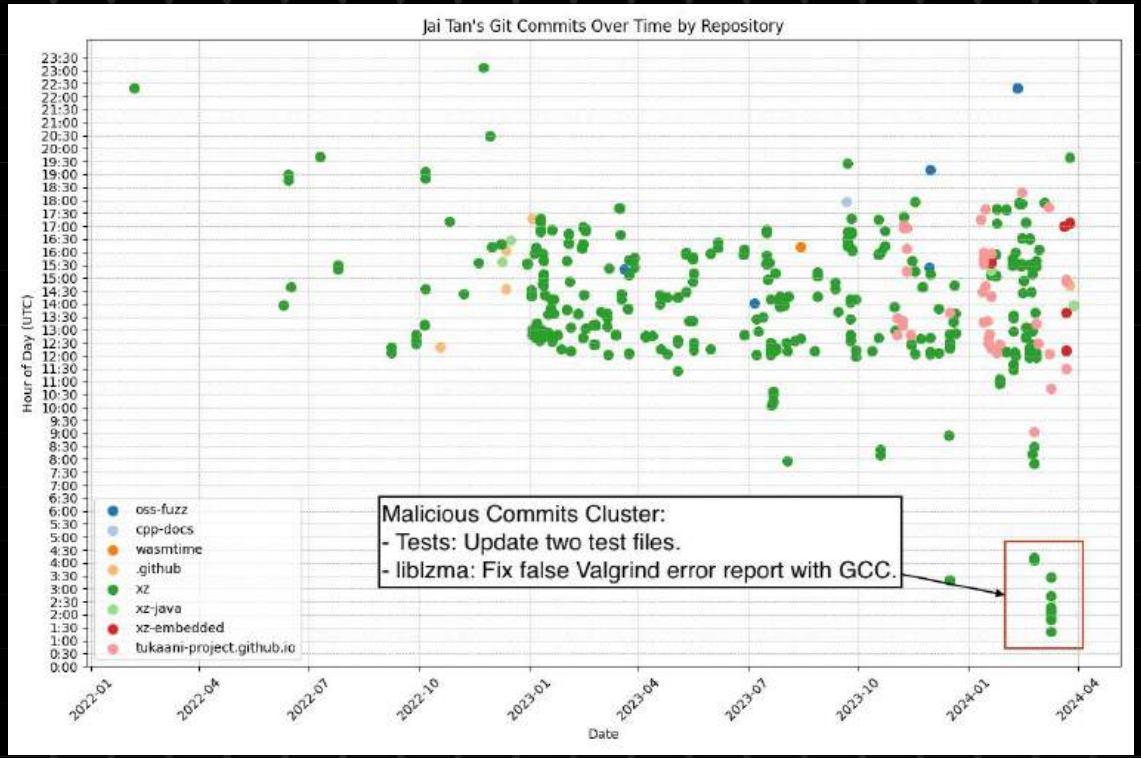
Rimozione della backdoor

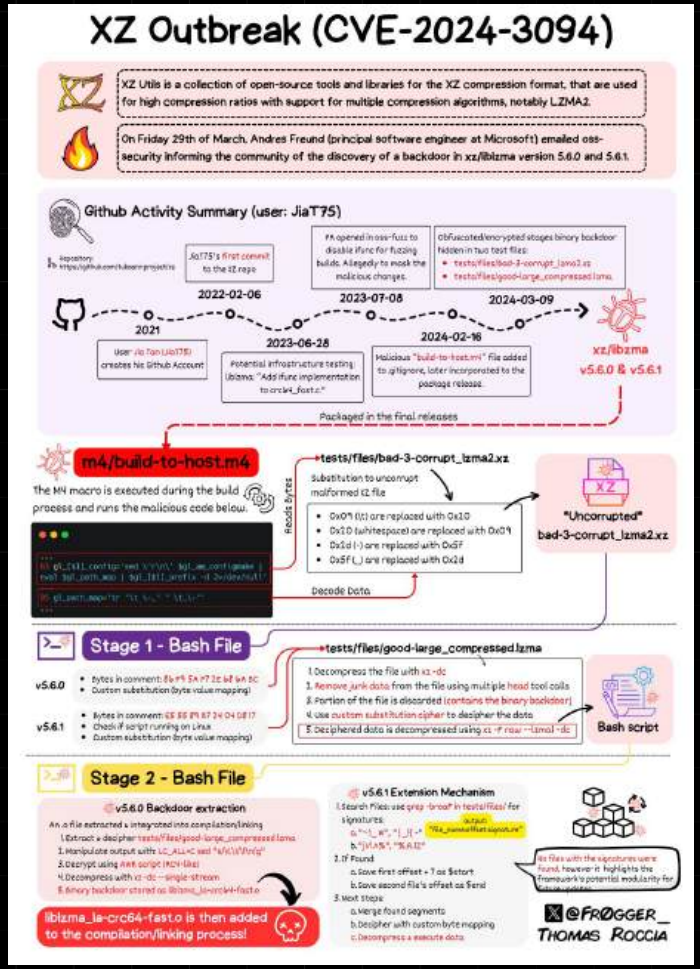
https://github.com/tukaani-project/xz/commit/e93e13c8b3bec925c56e0c0b675d8000a0f7f754#diff-e23c4ee4e2bf72b06ca17b08a69e54fef84e5fe19b3ff117a7d9566a798866b7

06

Conclusioni

Attribuzione dell’attacco





https://twitter.com/frOgger /status/1774342248437813525

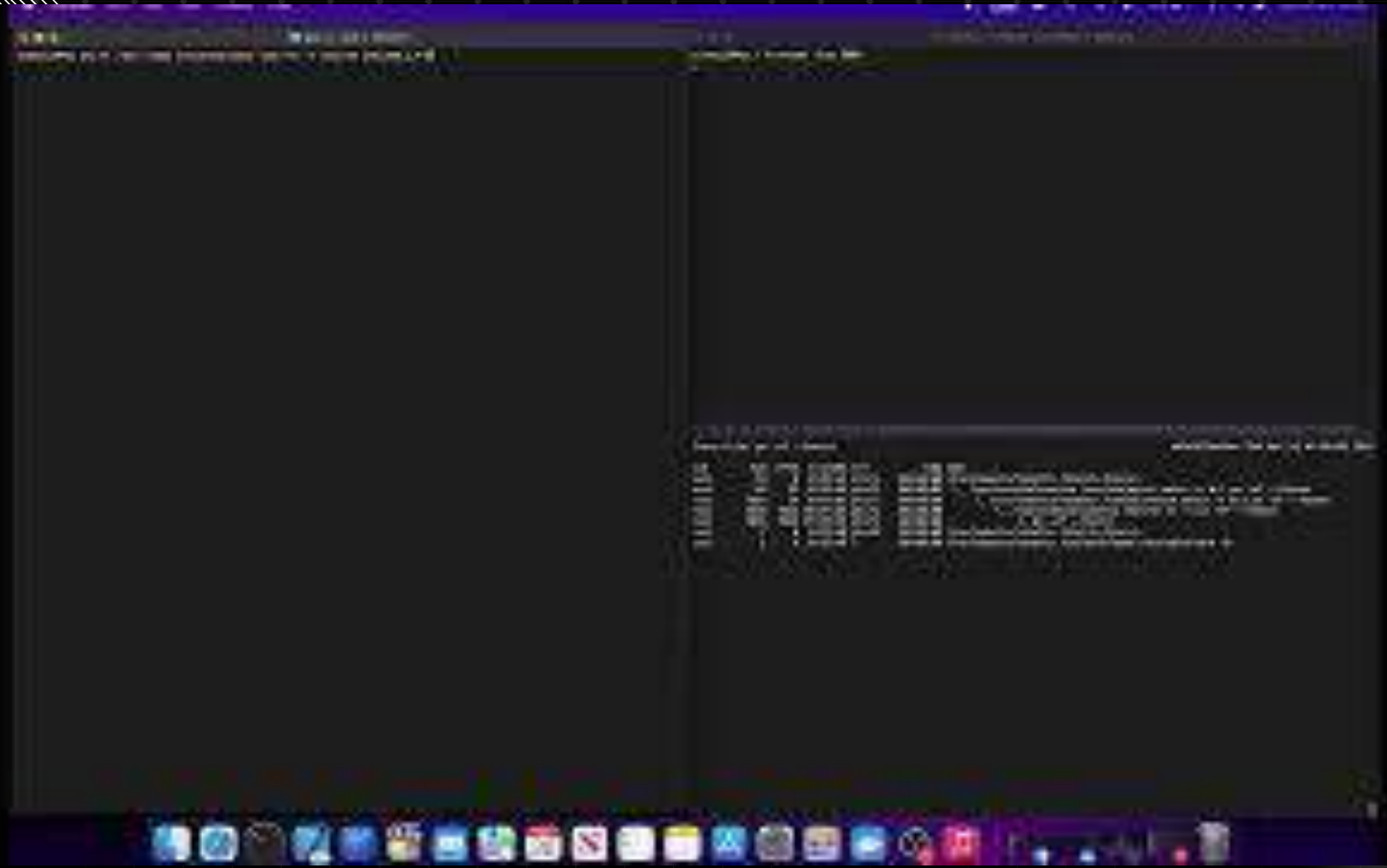
XZ Utils, lezione imparata?

- Definire ruoli chiari
- Analizzare il comportamento del contributore nel tempo (commit frequenti, tipologia di modifiche)
- Limitare chi può approvare e fare merge delle pull request, garantendo che solo membri fidati abbiano tali permessi
- Creare processi che garantiscono che le modifiche siano valutate tecnicamente, non per pressione esterna

Quanto possiamo fidarci?

Ora passiamo alla PoC dell’attacco

Alessio Ciccola



Grazie!

Template by Slidesgo, immagini DALL-E