The XZ Utils Backdoor

A Near-Miss in the Open-Source Supply Chain

Name: Gianluca Milani

Location: Perugia, Umbria, Italy

edX username: Gianluca-25

GitHub username: M1lo25

Recording date: 16/10/2025

Incident time: *March-april 2024*

CVE: CVE-2024-3094

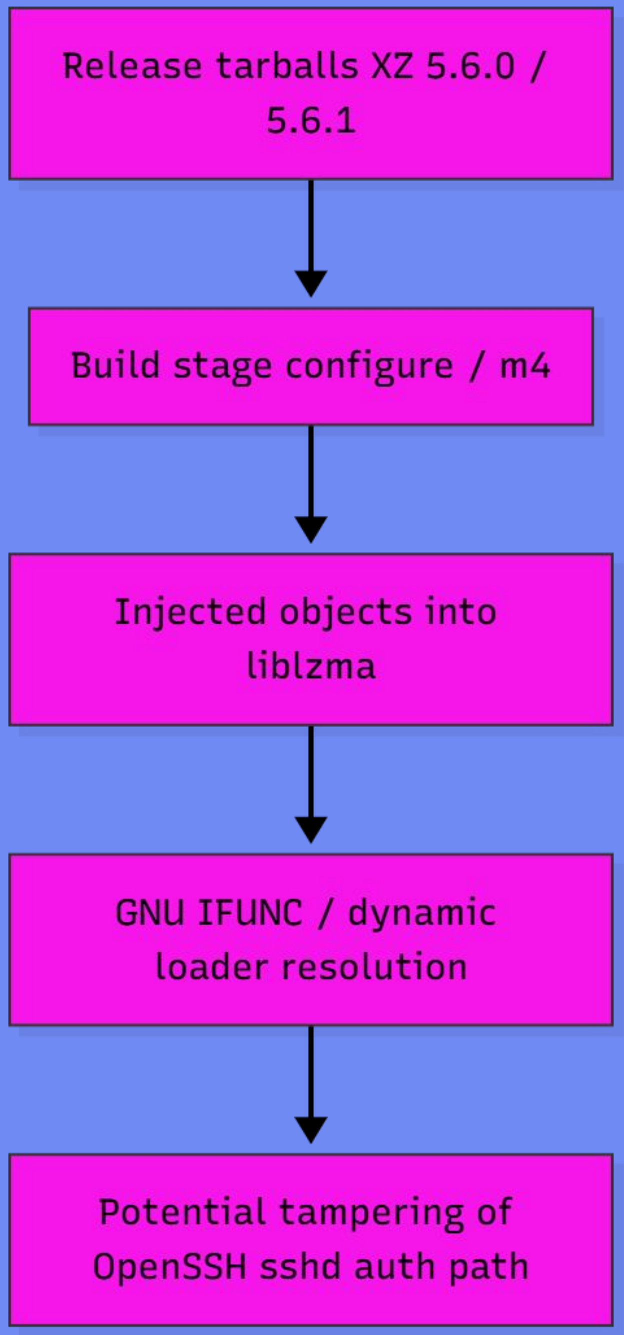
What Happened?

In March 2024, the release **tarballs** of xz-utils versions 5.6.0 and 5.6.1 contained malicious code absent from the upstream Git

Hidden objects were extracted into **liblzma** during the build phase

The payload leveraged **IFUNC** and dynamic symbol resolution, which in some builds interfered with the authentication of **OpenSSH**

It was discovered after SSH performance anomalies by **Andres Freund**



Release tarballs XZ 5.6.0 / 5.6.1

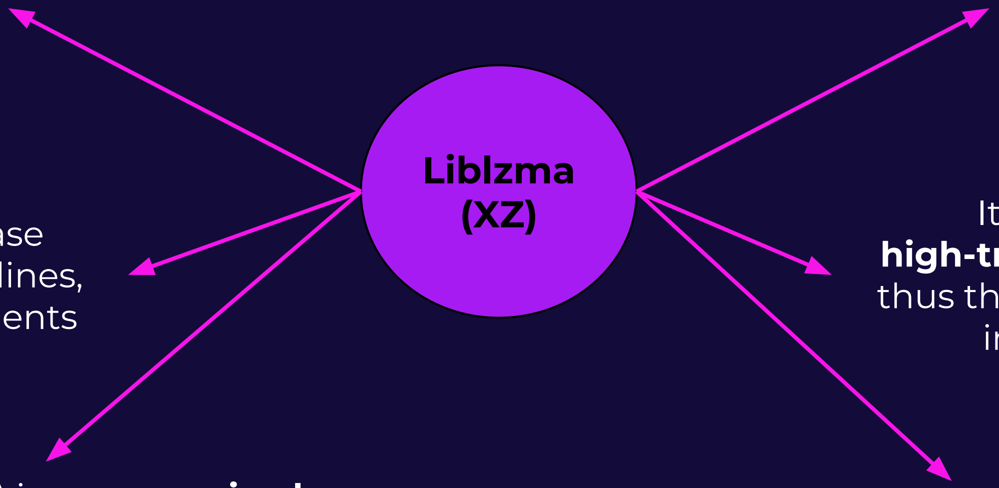
Build stage configure / m4

Injected objects into liblzma

GNU IFUNC / dynamic loader resolution

Potential tampering of OpenSSH sshd auth path

Why XZ Matters



Liblzma (XZ)

XZ/liblzma is a commonly used core library for compression in many Linux distros

This library exists in base images, packaging pipelines, and dev/build environments

If liblzma(XZ) is **compromised**, the contamination can propagate **downstream** even if the app code is unmodified

Software that relies on other libraries can load **liblzma(XZ)** and provide a good attack surface

It's often running in **high-trust/privileged context**; thus the potential compromise impact is very high

It has a **high leverage**: one library can control thousands of hosts or containers

The Attack Chain

PHASE1

Artifacts

The malicious artifacts were found only in the **tarballs** pertaining to the release, not in the public **Git** repository

PHASE2

Build-time

Such scripts and macros as **configure** and **m4** extract hidden objects and link them into liblzma

PHASE3

Targeted activation

Methods to obfuscate being masked, gated, and run based on certain parameters to avoid detection

PHASE4

Call redirection

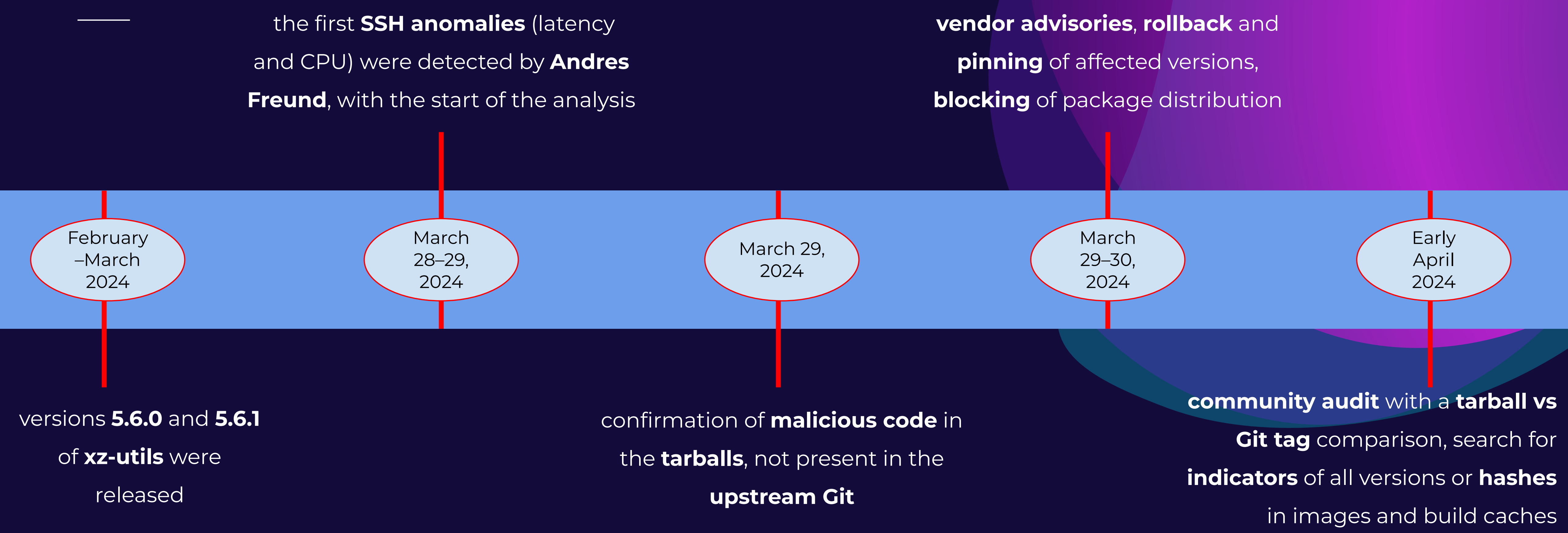
GNU IFUNC and dynamic symbol resolution interpose over sensitive functions

Such altering could yield levies against **OpenSSH** authentication through resolve system dependencies that subsequently load **liblzma**

Observable behaviors

Latency and **CPU spikes** became evident in SSH logins; verification of build-time injection was confirmed by **tarball vs. repo delta**.

Timeline & Discovery



February –March 2024

versions **5.6.0** and **5.6.1** of **xz-utils** were released

March 28–29, 2024

the first **SSH anomalies** (latency and CPU) were detected by **Andres Freund**, with the start of the analysis

March 29, 2024

confirmation of **malicious code** in the **tarballs**, not present in the **upstream Git**

March 29–30, 2024

vendor advisories, **rollback** and **pinning** of affected versions, **blocking** of package distribution

Early April 2024

community audit with a **tarball vs Git tag** comparison, search for **indicators** of all versions or **hashes** in images and build caches

Human Factors & Governance

Maintainer fatigue: extensive work activity, voluntary work, and fatigue affects the quality of reviews

Trust dynamics: long-term contributors may establish explicit trust, leading to formal checks being significantly reduced

Release pressure: the push to unblock versions can lead to faster approvals and review processes being skipped

Single-maintainer bottleneck: centralized decision-making primarily relies on one person, creating a **single point of failure**

Lack of artifact parity: systematic checks between **tarballs** and **Git** are missing so differences can go undetected

Build-process weakness: build-time scripts and macros are not always either **code-reviewed** or reproducibly reviewed independently

How the Backdoor Operated

Build & Link

Create triggers

The compilation, configuration, m4 macros, and scripts would add more objects than what was denoted in the **Git tags** thus modifying the **final binary** without a prior indicator.

Silent injection

The objects ended up being part of **final linking** and it got obscured as it was bundled with actual files due to **conditional rules** and identical/mimetic names.

Observable behaviors

Latency and **CPU spikes** became evident in SSH logins; verification of build-time injection was confirmed by **tarball vs. repo delta**.

Runtime Hijack

Runtime hijacking

Calls were hijacked to code controlled by the attacker via **GNU IFUNC** and **PLT/GOT**

Indirect target on SSH

The attack hijacked the **OpenSSH** authentication flow without patching the **SSH** by hijacking **compression/I/O functions**

Impact & Exposure

Vulnerable hosts and build systems
that built or installed **XZ 5.6.0 / 5.6.1** from **tarballs**: these are the most likely candidates for being affected

Potential impact
Compromised credentials and **lateral movement** of adversaries throughout larger environments

Ecosystem
Some **base images** and **containers** included the **compromised** images and deployed them **downstream** in the pipelines

Rapid risk mitigation
Advisories, **pinning** and **rollbacks** contained the initial impact and limited the use of the **malicious packages**

Long tail
Caching in **CI systems**, **internal mirrors**, and **long-standing images** may retain fugitive traces that could appear later

Recommendations

Organization

Artifact provenance

Require **code signing** and **release signing**, and always verify **signatures** before accepting an artifact

Artifact↔Git parity

Automate the **diff** between **tarball** and **Git tag** and configure **CI** to fail when there is any **non-uniftable differences**

Supply-chain hardening

Adopt **Reproducible Builds**, normalize for **SLSA L3+**, and enforce **two-person review** on every release as proces

Identity and keys

Apply timely **rotation** at regular intervals for maintainer and CI identity and access permissions that have **minimum scoping**, and **MFA** for maintainer and CI identity

Attestations & SBOM

Publish and verify **attestations** for builds (example **in-toto**) and keep **SBOM's** up to your best ability fresh and reachable

DevOps

Immediate containment

Pin/block the versions **XZ 5.6.0–5.6.1** and **scan** the respective **images** and **cache** to remove the **compromised version**

Build-time detections

Add **static/dynamic checks** on **macros** used, **build scripts**, and any **extra objects** discovered at **linking time**

Runtime verification

Monitoring **latency** and **CPU usage** on **auth paths (sshd)** and create **alerts** for **performance deviations**

Eradication & hygiene

Purge/rotate CI caches, **artifact repositories**, and **internal mirrors**; **regenerate** and **distribute** image **clean**

Incident readiness

Maintain **timeout playbooks** tested with **table top exercises** on regards to **rollbacks** and have **emergency communication** ready

References

NVD - CVE-2024-3094 (official record)

Openwall oss-security - Initial disclosure by Andres Freund (Mar 29, 2024)

Red Hat - Understanding Red Hat’s response to the XZ incident

Debian - DSA-5649-1 xz-utils security update

CERT-EU - Critical Vulnerability in XZ Utils

SLSA.dev - Supply-chain Levels for Software Artifacts (guidance)

Fedora Magazine - CVE-2024-3094: Urgent alert for Fedora 40/Rawhide users

Reproducible Builds - Why it matters / project resources

THANK YOU!!

GitHub Repo

https://github.com/M1lo25/CS50Cybersecurity

*Credit to **Andres Freund** and security teams for fast response*