New issue

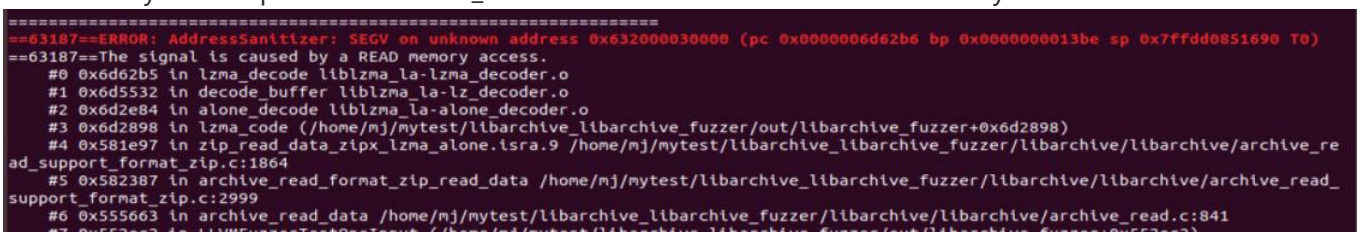
Jump to bottom

The libarchive lib exist a READ memory access Vulnerability #1672

Closed **icycityone** opened this issue on Feb 25 · 47 comments

icycityone commented on Feb 25 • edited

hello,when i use libfuzzer to write code to call archive_read_data function,i find a READ memory access Vulnerability.see the picture! The lzma_decode function crashed when decode my testcase.



kientzle commented on Feb 25 Contributor

Can you share the test case? That would help us to identify and fix this issue. It would also help if you could provide other details: What system were you using? What version of libarchive? What version of liblzma?

👍 2

icycityone commented on Feb 25 Author

hello,the crash testcase see the attachments。
The environment list below:
ubuntu 18
libfuzzer
liblzma 5.2.2 libarchive3.6.0
...

icycityone commented on Feb 25 Author

My github name is icycityone.I can not send mail by qq mail.This is my new mail.

hello,the crash testcase see the attachments。
The environment list below:
ubuntu 18
libfuzzer
liblzma 5.2.2 libarchive3.6.0
...

mmatuska commented on Feb 26 Member

@icycityone what we need is a sample test file to reproduce the vulnerability

icycityone commented on Feb 26 Author

hello,the testcase 、 source file 、crash file see the attachments。Besides，i have Submit cve aim at this question
...

mmatuska commented on Feb 26 Member

I am sorry, but I don't see any attachments to this issue. If you want you can mail the testcase to me directly at martin@matuska.org

icycityone commented on Feb 26 Author

> I am sorry, but I don't see any attachments to this issue. If you want you can mail the testcase to me directly at martin@matuska.org

you can see the next mail .

icycityone commented on Mar 1 Author

hello,do you received the message

mmatuska commented on Mar 5 Member

No, I didn't get an e-mail with the sample nor it is attached to this issue.

icycityone commented on Mar 5 — Author

oh i am sorry,my emai may be go wrong.can you see the attachedments now.
...

mmatuska commented on Mar 6 — Member

Hi, I got your testcase. I am unable to reproduce the error with contrib/archivetest.
Try to compile contrib/archivetest.c with CFLAGS="-g -fsanitize=address" and test the file.

icycityone commented on Mar 6 — Author

you should give a curpus when i work tomorrow i will send you it.and the compile file
...

mmatuska commented on Mar 6 — Member

Just to note, the archivetest output from reading your sample looks like this:

```
$ ./archivetest -f crash-58af2238755ec09600f15fed6e3e606c09638f42
Data source: filename: crash-58af2238755ec09600f15fed6e3e606c09638f42
Filter: none
Format: ZIP 0.7 (lzma)
Entry 1: ARCHIVE_OK, pathname unreadable, data: OK
Entry 2: ARCHIVE_OK, pathname unreadable, data: OK
Entry 3: ARCHIVE_OK, pathname unreadable, data: OK
Entry 4: ARCHIVE_OK, pathname unreadable, data: OK
Entry 5: fatal error reading header
Last return code: ARCHIVE_FATAL (-30)
Error string: Truncated ZIP file data
```

I got no errors from ASAN.

icycityone commented on Mar 6 — Author

if you compile with the needed lib .my computer is not by hand .i cannot seethe detail now
...

icycityone commented on Mar 6 — Author

 you can do follow this. Look forward to you reply.
step 1:
apt-get update && apt-get install -y make autoconf automake libtool pkg-config \
        libbz2-dev liblzo2-dev liblzma-dev liblz4-dev libz-dev \
        libxml2-dev libssl-dev libacl1-dev libattr1-dev

step 2:
git clone --depth 1 https://github.com/libarchive/libarchive.git
cd libarchive
./build/autogen.sh
./configure
make -j$(nproc) all

step 3
build fuzzer(s)
clang++ -g0 -std=c++11 -fsanitize=address,fuzzer -llibarchive \
    ../$SRC/libarchive_fuzzer.cc -o ../$OUT/libarchive_fuzzer \
     .libs/libarchive.a \
    -Wl,-Bstatic -lbz2 -llzo2 -lxml2 -llzma -lz -lcrypto -llz4 -licuuc \
    -licudata -Wl,-Bdynamic -ldl
step4
./libarchive_fuzzer crashfile
or
./libarchive_fuzzer testcases/   #testcase file see the attachedments
...

icycityone commented on Mar 9 — Author

hello have you reproduced the vul?
...

mmatuska commented on Mar 9 — Member

Yes, I have managed to reproduce the error. I have used contrib/oss-fuzz/libarchive_fuzzer.cc
I currently don't have time to investigate on this issue.

icycityone commented on Mar 9 — Author

ok，can i apply for cve on this subject ?
...

icycityone commented on Mar 10 — Author

ok，can i apply for cve on this subject ?
...

mmatuska commented on Mar 10 — Member

Well, the actual invalid read happens in liblzma. We have to find out if there is a mistake in libarchive by wrongly calling liblzma or if there is an error in liblzma. A CVE against libarchive with a bug in liblzma would be counterproductive.

10.03.2022 15:20:23 icycityone ***@***.***>:
...

icycityone commented on Mar 10 — Author

You mean i can file cVE against Liblzma if the problem is caused by Liblzma. Have you identified which caused it

thanks.
...

icycityone commented on Mar 11 — Author

You mean i can file cVE against Liblzma if the problem is caused by Liblzma. Have you identified which caused it

thanks.
...

icycityone commented on Mar 14 — Author

hello,Have you identified which caused it thanks.
...

JiaT75 commented on Mar 18 — Contributor

@mmatuska **@icycityone** can either of you provide the crashfile in this thread or to my email jiat0218@gmail.com? I have been working with liblzma quite a bit recently and I can help determine if this is a libarchive or liblzma problem

icycityone commented on Mar 21 Author

hello,i have discussed with security engineer from liblzma,the vulnerability may caused by zipx_lzma_alone_init().if you have time,please debug to check it.look forward to you reply thanks.
The Message records bellow:

On 2022-03-18 jun ma wrote:
> oh， thanks for your reply ， i have debuged it using gdb and discussed
> with security engineer from libarchive,it
>
> high possibility caused by liblzma.when you have time, hope you can
> take some time to have a look.thanks!

The problem is that lzma_code() is called with lzma_stream.avail_in set
to 18446744073709551607 which equals -9 when interpret as a signed
value. That is, it's a bug in libarchive. I attached a patch that adds
a few fprintf() calls to libarchive which show how the incorrect value
comes from zipx_lzma_alone_init() but I didn't debug how to fix it.
Here is the output:

DEBUG: libarchive/archive_read_support_format_zip.c:1607: init
DEBUG: libarchive/archive_read_support_format_zip.c:1728: zip->entry_bytes_remaining = 1
DEBUG: libarchive/archive_read_support_format_zip.c:1859: zip->entry_bytes_remaining = 1, bytes_avail = 94559
DEBUG: libarchive/archive_read_support_format_zip.c:1916: zip->entry_bytes_remaining = 1, to_consume = 1
DEBUG: libarchive/archive_read_support_format_zip.c:1607: init
DEBUG: libarchive/archive_read_support_format_zip.c:1728: zip->entry_bytes_remaining = 1
DEBUG: libarchive/archive_read_support_format_zip.c:1859: zip->entry_bytes_remaining = 1, bytes_avail = 94501
DEBUG: libarchive/archive_read_support_format_zip.c:1916: zip->entry_bytes_remaining = 1, to_consume = 1
DEBUG: libarchive/archive_read_support_format_zip.c:1607: init
DEBUG: libarchive/archive_read_support_format_zip.c:1728: zip->entry_bytes_remaining = -9
DEBUG: libarchive/archive_read_support_format_zip.c:1859: zip->entry_bytes_remaining = -9, bytes_avail = 94443

Can you forward the above information to libarchive developers?

Thank you for your effort to look for bugs and reporting them!

Two minor things for the future:

I understand that sending the bad .zip file directly as an attachment
doesn't work for some destinations because antivirus products can block

such emails (GMail does). I could extract the .rar with 7z from p7zip so it wasn't a problem for me, but in general free software developers prefer exchanging information using fully open formats like .zip, .7z, or .tar + some compressor. .zip with its very old (and insecure) encryption is supported widely. I tested this with GMail:

    zip -e badfile.zip badfile.bin # password set to 123456

This worked. Without encryption it was rejected. I think the filename inside the encrypted .zip must be something like .bin instead of .zip since filenames aren't encrypted.

The second thing is that I did receive the email with the subject crash-58af2238755ec09600f15fed6e3e606c09638f42 but I wasn't on my computer during those days, so I was slow to reply, sorry. (My email provider doesn't block attachments as easily as GMail does.) The email contained a 46-megabyte attachment (base64-encoded size) which I suppose was the test program binary. That is a fairly big file to attach and most people (me included) won't run binaries received in email.

Again, thanks for your help in finding and reporting bugs!
...

1

JiaT75 mentioned this issue on Mar 22

Moved negative check for input byte count to after min calculation #1682

Closed

kientzle commented on Mar 23 — Contributor

The analysis above points to a bug in `zipx_lzma_alone_init`. In line 1678, we read 9 bytes from the entry

```
if((p = __archive_read_ahead(a, 9, NULL)) == NULL) {
```

At line 1720, after successfully parsing the 9-byte header and initializing the decompressor, we subtract 9 bytes from the entry size:

```
__archive_read_consume(a, 9);
zip->entry_bytes_remaining -= 9;
```

I believe this is where the `-9` is coming from in the trace above. I believe the bug is that in line 1678 above, we read 9 bytes *even if there weren't 9 bytes in the entry*.

So I believe the correct answer is to change the test in line 1678 to ensure that there are in fact 9 bytes available in the entry before we try to read those 9 bytes:

```
/* Read magic1,magic2,lzma_params from the ZIPX stream. */
if(archive->entry_bytes_remaining < 9 || (p = __archive_read_ahead(a, 9, NULL)) == NULL) {
        archive_set_error(&a->archive, ARCHIVE_ERRNO_FILE_FORMAT,
            "Truncated lzma data");
        return (ARCHIVE_FATAL);
}
```

icycityone commented on Mar 23 — Author

so the bug is certainly caused by libarchive not by liblzma. Can i apply for cve on this subject **@kientzle**

mmatuska pushed a commit to mmatuska/libarchive that referenced this issue on Mar 24

`ZIP reader: fix out-of-bounds read in zipx_lzma_alone_init()` ··· ✕ `855f83c`

kientzle commented on Mar 24 — Contributor

@icycityone Yes, please go ahead and apply for a CVE.

Has anyone verified that my suggested change above fixes this issue?

icycityone commented on Mar 24 — Author

hello,thanks for your reply.Is it possible to pass this commit by https://cveform.mitre.org/ or submit by other means. And how long will it take to get the reply. Look forward to your reply. **@kientzle**

mmatuska closed this as completed in `cfaa281` on Mar 25

mmatuska commented on Mar 25 — Member

I have submitted the patch. Btw. is a out-of-bounds read that does nothing actually eligible for a CVE? The application will read some bytes beyond the boundary but it will always die and stop processing.

icycityone commented on Mar 25 — Author

oh thank to your reply. I have apply a cve after all i have take a lot of time to test.
...

mmatuska commented on Mar 26 — Member

Linking with OSS-Fuzz issue 38766 that got resolved by the fix:
https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=38766#c4

risicle commented on Apr 2

@mmatuska the process will only be killed if the read is off the edge of a page boundary and the next page is not readable. That still leaves up to 4095 bytes that could be read unimpeded, even if the next page is unreadable. OOB reads can still be very serious if, for example, the process also has credentials in its address space.

mmatuska commented on Apr 3 — Member

@risicle the data read here is treated in libarchive as "external data", so libarchive will erroneously consider the data to be from the archive and interpret the rest of the 9-byte header. All subsequent reads (based or not based on the header data received) will fail on the empty entry_bytes_remaining.

mcandre commented on Apr 7

When can we tag and release a new, patched version?

Will fixes be backported to earlier libarchive release series, too?

jpuhlman pushed a commit to MontaVista-OpenSourceTechnology/poky that referenced this issue on Apr 8


`CVE-2022-26280 libarchive: out of bounds in zipx_lzma_alone_init ...` `48f1f5c`


CamberLoid mentioned this issue on Apr 14

libarchive: security update to 3.6.1 AOSC-Dev/aosc-os-abbs#3931

Merged

10 tasks

praharsh23 added a commit to praharsh23/libarchive that referenced this issue on May 3

`Verify fix for libarchive#1672` ✕ `0f4b585`

carnil commented on May 6

CVE-2022-28066 appers to have been assigned to this issue.

Cristian-Bejan commented on May 9

Any reason as to why 2 CVE (CVE-2022-26280 CVE-2022-28066) have been assigned to this issue?

carnil commented on May 9

I suspect this was assigned then by error. Asked MITRE if one of both
should be rejected.

icycityone commented on May 9 Author

oh the cve is sumbmit by me
...

icycityone commented on May 9 Author

i have submited twice.May they have not cheched it
...

Cristian-Bejan commented on May 9

@carnil Thank you! My understanding is that these are both OOB with a DoS impact and 37d6aff fixes them. Is the root cause the thing that makes them apart?

Cristian-Bejan commented on May 9

@icycityone That might be the case. You submitted different descriptions so I'm assuming they thought it's a different vulnerability.

icycityone commented on May 9 Author

i have not noticed this since i received the reply from MITRE.I recived the cve numbers is 28260.

icycityone commented on May 9 Author

> **@icycityone** That might be the case. You submitted different descriptions so I'm assuming they thought it's a different vulnerability.

maybe,because this is my first time to submit,so to ensure they recevied,i have sumbmited twice

srccn commented on May 11

@icycityone: regarding to CVE reporting,

1. could you please give detailed affected previous release versions in report? something like from version 3.4.0 to 3.6.0? that will help NVD to populate CPE list. It will be even better if providing enumerated all affected released versions.
2. Definitely suggest to keep one CVE open, suggest to close CVE-2022-28066 and keep CVE-2022-26280, because CVE-2022-26280 already have CVSS score assigned.

icycityone commented on May 12 Author

oh i am busy those days and have no time to test other versions.ANd i have searched but did not know how to cancel the cve
...

carnil commented on May 12

Requesting for REJECT of a CVE can be done via
https://cveform.mitre.org/ -> Select a request type -> Request an update to an existing CVE entry -> Type of update requested -> Rejection.

mfrw mentioned this issue on Jun 13

libarchive: bump version to 3.6.1 to address CVE-2022-26280 microsoft/CBL-Mariner#3154

Merged

12 tasks

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging a pull request may close this issue.

Moved negative check for input byte count to after min calculation
JiaT75/libarchive

9 participants