


New issue

Jump to bottom

WavPack crashes with SEGFAULT #91

 Open wakolzlin opened this issue on Dec 27, 2020 · 16 comments

Assignees



wakolzlin commented on Dec 27, 2020 • edited

Hello!

This bug was found by Crusher (fuzzer developing in ISP RAS), thanks to following colleagues:
Vitalii Akolzin, Shamil Kurmangaleev, Maxim Mishechkin, Fedor Nis'kov, Ivan Gulakov, Denis Straghkov, Andrey Fedotov, Alexey Vishnyakov, Daniil Kutz, Alexander Novikov.

Product version

Commit [948a8b7](#) (latest commit on master at current moment).

Environment

Ubuntu 16.04/18.04

To reproduce

1. Extract `crash.wav` from [crash.wav.tar.gz](#)
2. Run:

```
wavpack -y crash.wav
```

Program crashes with SEGFAULT.

Error message:

```
WAVPACK Hybrid Lossless Audio Compressor Linux Version 5.3.2
Copyright (c) 1998 - 2020 David Bryant. All Rights Reserved.

creating crash.wv, Segmentation fault (core dumped)
```

Valgrind output:

```
creating crash.wv,==29284== Invalid write of size 4
==29284== at 0x122C08: WavpackPackSamples (pack_utils.c:662)
==29284== by 0x1111E0: pack_audio (wavpack.c:2487)
==29284== by 0x1111E0: pack_file (wavpack.c:1911)
==29284== by 0x108792: main (wavpack.c:1278)
==29284== Address 0x559d3f0 is 0 bytes after a block of size 162,800 alloc'd
==29284== at 0x4C2F80F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==29284== by 0x12282B: WavpackPackInit (pack_utils.c:561)
==29284== by 0x110FE6: pack_audio (wavpack.c:2323)
==29284== by 0x110FE6: pack_file (wavpack.c:1911)
==29284== by 0x108792: main (wavpack.c:1278)
==29284==
==29284== Invalid write of size 4
==29284== at 0x122C17: WavpackPackSamples (pack_utils.c:663)
==29284== by 0x1111E0: pack_audio (wavpack.c:2487)
==29284== by 0x1111E0: pack_file (wavpack.c:1911)
==29284== by 0x108792: main (wavpack.c:1278)
==29284== Address 0x559d3f4 is 4 bytes after a block of size 162,800 alloc'd
==29284== at 0x4C2F80F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==29284== by 0x12282B: WavpackPackInit (pack_utils.c:561)
==29284== by 0x110FE6: pack_audio (wavpack.c:2323)
==29284== by 0x110FE6: pack_file (wavpack.c:1911)
==29284== by 0x108792: main (wavpack.c:1278)
==29284==
0% done...--29284-- VALGRIND INTERNAL ERROR: Valgrind received a signal 11 (SIGSEGV) - exiting
--29284-- si_code=1; Faulting address: 0x7f0563d49e; sp: 0x1002dade30

valgrind: the 'impossible' happened:
Killed by fatal signal

host stacktrace:
==29284== at 0x58053139: ??? (in /usr/lib/valgrind/memcheck-amd64-linux)
==29284== by 0x5800BA84: ??? (in /usr/lib/valgrind/memcheck-amd64-linux)
==29284== by 0x5800BC66: ??? (in /usr/lib/valgrind/memcheck-amd64-linux)
==29284== by 0x5809F785: ??? (in /usr/lib/valgrind/memcheck-amd64-linux)
==29284== by 0x580AED50: ??? (in /usr/lib/valgrind/memcheck-amd64-linux)

sched status:
running_tid=1

Thread 1: status = VgTs_Runnable (lwpid 29284)
==29284== at 0x4C2F80F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==29284== by 0x1211E2: pack_streams (pack_utils.c:955)
==29284== by 0x122DB1: WavpackFlushSamples (pack_utils.c:727)
==29284== by 0x112537: pack_audio (wavpack.c:2444)
==29284== by 0x112537: pack_file (wavpack.c:1911)
==29284== by 0x108792: main (wavpack.c:1278)
```

Analysis

1. In

WavPack/src/pack_utils.c
Line 561 in 940a8b7

```
561 wps->sample_buffer = malloc (wpc->max_samples * (wps->wphdr.flags & MONO_FLAG ? 4 : 8));
```

malloc 's argument overflows size_t type and results in small positive number.
So, only a short memory region is allocated.

2. Then

WavPack/src/pack_utils.c
Line 612 in 940a8b7

```
612 dptr = wps->sample_buffer + wpc->acc_samples * (wps->wphdr.flags & MONO_FLAG ? 1 : 2);
```

dptr now points to address in previously allocated region

3. Finally

WavPack/src/pack_utils.c
Line 662 in 940a8b7

```
662 *dptr++ = (signed char) sptr [0];
```

we write in memory by dptr pointer. But in one moment dptr points to memory outside of allocated region. Segmentation fault.



dbry self-assigned this on Dec 27, 2020

carnil commented on Dec 28, 2020

Appears that [CVE-2020-35738](#) was assigned for this issue.

utkarsh2102 commented on Dec 28, 2020 • edited

Hi, not sure if it helps but..

I couldn't reproduce this in version 4.70.0 (Debian Jessie):

```
root@18227e5bad53:/# wavpack -y crash.wav

WAVPACK Hybrid Lossless Audio Compressor Linux Version 4.70.0
Copyright (c) 1998 - 2013 Conifer Software. All Rights Reserved.

created crash.wv in 0.09 secs (lossless, 59.17%)
```

However, could reproduce this in version 5.0.0 (Debian Stretch):

```
root@8eb0c583e3cf:/# wavpack -y crash.wav

WAVPACK Hybrid Lossless Audio Compressor Linux Version 5.0.0
Copyright (c) 1998 - 2016 David Bryant. All Rights Reserved.

creating crash.wv, Segmentation fault (core dumped)
```

And can also reproduce in versions thereafter, for eg: in version 5.1.0 (Debian Buster):

```
root@d307c2c3c018:/# wavpack -y crash.wav

WAVPACK Hybrid Lossless Audio Compressor Linux Version 5.1.0
Copyright (c) 1998 - 2017 David Bryant. All Rights Reserved.

creating crash.wv, Segmentation fault (core dumped)
```

By this, I am also inclined towards believing that vulnerable code wasn't present in v4.70.0. Please let me know if I am mistaken somehow :)

carnil commented on Dec 28, 2020

@utkarsh2102 FYI: If you want to be sure you should not rely only on reproducibility with a POC but inspect the code as well (but note that in [38dba75](#) a code reorganization and cleanup was done).

utkarsh2102 commented on Dec 28, 2020

Aah, well. Right, let's wait for the fix and then re-analyze the situation from there.

dbry added a commit that referenced this issue on Dec 29, 2020


issue #91: filter out invalid negative sample rates to prevent encode...

63f3ec7

 dbry added a commit that referenced this issue on Dec 29, 2020

 issue #91: fix integer overflows resulting in buffer overruns and san... ...

✓ 89df160

 wakolzin closed this as completed on Dec 30, 2020

utkarsh2102 commented on Dec 30, 2020 • edited ▼

Aah, well. Right, let's wait for the fix and then re-analyze the situation from there.

Hm, so as I see it, in Debian Jessie (v4.70.0), there's no check on `sample_rate` so [63f3ec7](#) doesn't apply. As for [89df160](#), the only applicable diff is:

```
--- a/src/wputils.c
+++ b/src/wputils.c
@@ -1096,10 +1096,10 @@ int WavpackPackInit (WavpackContext *wpc)
     else
         wpc->block_samples = wpc->config.sample_rate;

-    while (wpc->block_samples * wpc->config.num_channels > 150000)
+    while ((int64_t) wpc->block_samples * wpc->config.num_channels > 150000)
         wpc->block_samples /= 2;

-    while (wpc->block_samples * wpc->config.num_channels < 40000)
+    while ((int64_t) wpc->block_samples * wpc->config.num_channels < 40000)
         wpc->block_samples *= 2;

    if (wpc->config.block_samples) {
```

I am not sure if backporting this diff is worthy enough to avoid causing buffer overruns and avoiding integer overflows. Does anybody have any opinion on this? Do you think it's still "safe and recommended" to backport this diff to v4.70.0?

dbry commented on Dec 30, 2020


Owner

First of all, thank you [@wakolzin](#) for finding and reporting this issue! This is now fixed in the master branch.

I have also carefully analyzed the issue and I believe that this is an extremely low risk from a security perspective. The buffer overflow could be exploited to corrupt the next heap entry and possibly force the next malloc to return an arbitrary address, but unfortunately that next malloc is for the compressed audio block and there's no method for the attacker to control what gets written there. Another complicating factor is that the compressor modifies the audio data "in-place" during decorrelation, so again there's no way to control the final contents. I'm certainly no expert in this, but I think it would be extraordinarily difficult, if not impossible, to create an exploit for this overrun.

But more problematic, even if it *were* possible to create an exploit, it would be nearly impossible to trigger it from a practical standpoint. The WavPack command-line program is not a default application for WAV files (or any file type) and is not supplied by default in any distro. Therefore the attacker would have to convince a user to install the command-line program, manually download the maliciously crafted file, and attempt, *using the terminal*, to compress it using WavPack. If you can get someone to do that, there are far simpler ways of gaining control of their system.

 2

 dbry reopened this on Dec 30, 2020

dbry commented on Dec 30, 2020 • edited ▼

Owner

[@utkarsh2102](#) With respect to 4.70.0, I actually find it a little frightening that that 7+ year-old release is still being patched, but I'm sure there are some good reasons for this. Personally, I am of the opinion that it's good practice to maintain backward compatibility unless absolutely unavoidable (maybe I'm just old) and I have worked very hard to make sure that the latest version is a "drop-in" replacement for older versions (see first paragraph of [porting guide](#)).

In any event, assuming all the other "real" security issues fixed since 2013 have been patched, the patch you suggest is safe, and I would add the other checks too:

```
@@ -947,6 +947,21 @@ int WavpackSetConfiguration (WavpackContext *wpc, WavpackConfig *config, uint32_
    int num_chans = config->num_channels;
    int i;

+    if (config->sample_rate <= 0) {
+        strcpy (wpc->error_message, "sample rate cannot be zero or negative!");
+        return FALSE;
+    }
+
+    if (num_chans <= 0 || num_chans > NEW_MAX_STREAMS * 2) {
+        strcpy (wpc->error_message, "invalid channel count!");
+        return FALSE;
+    }
+
+    if (config->block_samples && (config->block_samples < 16 || config->block_samples > 131072)) {
+        strcpy (wpc->error_message, "invalid custom block samples!");
+        return FALSE;
+    }
+
    wpc->total_samples = total_samples;
    wpc->config.sample_rate = config->sample_rate;
    wpc->config.num_channels = config->num_channels;
@@ -1096,10 +1111,10 @@ int WavpackPackInit (WavpackContext *wpc)
    else
        wpc->block_samples = wpc->config.sample_rate;

-    while (wpc->block_samples * wpc->config.num_channels > 150000)
+    while ((int64_t) wpc->block_samples * wpc->config.num_channels > 150000)
         wpc->block_samples /= 2;

-    while (wpc->block_samples * wpc->config.num_channels < 40000)
+    while ((int64_t) wpc->block_samples * wpc->config.num_channels < 40000)
         wpc->block_samples *= 2;

    if (wpc->config.block_samples) {
```

As for whether these are recommended, I refer you to my statements above. I really appreciate the work of volunteer maintainers, and I hate to see time and effort spent on vulnerabilities that aren't real threats. I guess it depends on the threat level eventually assigned to this CVE (which I hope is very low).

Thanks!

wakolzin commented on Dec 30, 2020

Author

@dbry Thank you for analysis and fix!

juanfra684 commented on Jan 6, 2021

@dbry are you going to release a new version or should we backport the changes to 5.3.0?

dbry commented on Jan 7, 2021

Owner

@juanfra684 I will release a new version in a few days from current source (so it will include a few other updates also, mostly minor fixes but also the `quick verify (-vv)` feature).

@wakolzin If you have any other issues like this that you know about it would be great to see them before I generate a release.

Thanks!

wakolzin commented on Jan 8, 2021

Author

@dbry This issue was the only I know about.



dbry commented on Jan 9, 2021

Owner

Note that this issue has two patches associated with it, [63f3ec7](#) and [89df160](#). Unfortunately the first one failed the Travis build (for an unrelated reason), but they are both in the commit log tagged with issue [#91](#).

It's a little tricky to verify that both were picked up everywhere, but I noticed on [this page](#) that at least Ubuntu may not have.

The good news is that [89df160](#) is sufficient by itself to resolve the SEGFAULT because the potentially negative `sample_rate` is assigned to `block_samples`, which is unsigned. I verified this with both crashers.

However, if anyone mistakenly *only* brings in [63f3ec7](#), then that would be insufficient to resolve the issue.

dbry commented on Jan 12, 2021

Owner

WavPack 5.4.0 has been released with this fix. Thanks again!

utkarsh2102 commented on Jan 14, 2021

Hello @dbry,

But more problematic, even if it were possible to create an exploit, it would be nearly impossible to trigger it from a practical standpoint. The WavPack command-line program is not a default application for WAV files (or any file type) and is not supplied by default in any distro.

That isn't true. The vulnerability isn't in the wavpack binary but in the "libwavpack" which is being used widely by a bunch of other programs and applications. For example, ffmpeg, cmus, audiotools, et al.

@utkarsh2102 With respect to 4.70.0, I actually find it a little frightening that that 7+ year-old release is still being patched, but I'm sure there are some good reasons for this.

Haha, yes. There exists Debian LTS and ELTS, where the packages in the Debian archive are supported (only against security vulnerabilities) for extended 2 years for LTS and then another 2 years for ELTS.

Personally, I am of the opinion that it's good practice to maintain backward compatibility unless absolutely unavoidable (maybe I'm just old) and I have worked very hard to make sure that the latest version is a "drop-in" replacement for older versions (see first paragraph of [porting guide](#)).

In any event, assuming all the other "real" security issues fixed since 2013 have been patched, the patch you suggest is safe, and I would add the other checks too.

Thank you so much for your help and effort in writing this patch and of course, for your hard work in best maintaining the backward compatibility! ❤️

This really helps in maintaining and backporting security patches to the older versions! \o/

dbry commented on Jan 15, 2021

Owner

Hello @dbry,

But more problematic, even if it were possible to create an exploit, it would be nearly impossible to trigger it from a practical standpoint. The WavPack command-line program is not a default application for WAV files (or any file type) and is not supplied by default in any distro.

That isn't true. The vulnerability isn't in the wavpack binary but in the "libwavpack" which is being used widely by a bunch of other programs and applications. For example, ffmpeg, cmus, audiotools, et al.

Yes, the vulnerability is in libwavpack, and that's widely distributed and used, but it's in the *encoder* portion, not the *decoder*. The decoder is constantly fuzzed in Google's OSS-Fuzz because that's where serious vulnerabilities will be located given that libwavpack is one of the *only consumers* of WavPack files (aside from the native Ffmpeg decoder). Many of the programs that use libwavpack only do decoding (like cmus that you mention), so they're going to be fine.

The vulnerability here is when crazy values for sample rate and/or channel count are passed to libwavpack for encoding. So any program using the library for encoding regular audio is not going to be affected because the parameters are going to be fixed (like a CD ripper). The program would have to be attempting to convert one of these crafted files and the most likely scenario is that the program would immediately recognize the corruption and would never end up trying to pass it to libwavpack (or the command-line program). For example, I just tested Ffmpeg, Foobar2000, and Gstreamer and they all refused to even open either of these files.

So this vulnerability is best suited for the command-line WavPack program because it accepts a wide range of encoding parameters (because the Wavpack *format* supports a wide range of parameters). I would be very surprised if another libwavpack application could be coaxed into even triggering this crash, much less used in an actual exploit.

Thank you so much for your help and effort in writing this patch and of course, for your hard work in best maintaining the backward compatibility!

This really helps in maintaining and backporting security patches to the older versions! \o/

I'm glad to help, and I will be happy to respond to your other queries as time permits! 🙌

utkarsh2102 commented on Jan 15, 2021

That isn't true. The vulnerability isn't in the wavpack binary but in the "libwavpack" which is being used widely by a bunch of other programs and applications. For example, ffmpeg, cmus, audiotools, et al.

Yes, the vulnerability is in libwavpack, and that's widely distributed and used, but it's in the *encoder* portion, not the *decoder*. The decoder is constantly fuzzed in Google's OSS-Fuzz because that's where serious vulnerabilities will be located given that libwavpack is one of the only *consumers* of WavPack files (aside from the native Ffmpeg decoder). Many of the programs that use libwavpack only do decoding (like cmus that you mention), so they're going to be fine.

The vulnerability here is when crazy values for sample rate and/or channel count are passed to libwavpack for encoding. So any program using the library for encoding regular audio is not going to be affected because the parameters are going to be fixed (like a CD ripper). The program would have to be attempting to convert one of these crafted files and the most likely scenario is that the program would immediately recognize the corruption and would never end up trying to pass it to libwavpack (or the command-line program). For example, I just tested Ffmpeg, Foobar2000, and Gstreamer and they all refused to even open either of these files.

So this vulnerability is best suited for the command-line WavPack program because it accepts a wide range of encoding parameters (because the Wavpack *format* supports a wide range of parameters). I would be very surprised if another libwavpack application could be coaxed into even triggering this crash, much less used in an actual exploit.

Aah, okay. Thanks for the explanation! 🙌

I'm glad to help, and I will be happy to respond to your other queries as time permits! +1

Great, thank you!

 bob-beck pushed a commit to openbsd/ports that referenced this issue on Jan 15, 2021



SECURITY: ...

ac05eba

Assignees

 dbry

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

5 participants

