percona / **percona-xtrabackup** Public

Code   **Pull requests** 12   Actions   Projects   Wiki   Security   ···

New issue

Jump to bottom

PXB-2854 - Quicklz decompression memory corruption issue fix #1366

Open **Chaloff** wants to merge 1 commit into `percona:8.0` from `Chaloff:github-quicklz-fix`

Conversation 14   Commits 1   Checks 1   Files changed 3

Chaloff commented on Aug 19

There is a memory corruption issue inside the quicklz.c source file that ships with Percona XtraBackup. Specifically the problem happens on copying user-supplied binary data over heap allocated memory buffers of user-controlled size. This allows corruption of heap data structures and potential arbitrary code execution.

The code in question is inside the qlz_decompress function of quicklz.c file:

```
size_t qlz_decompress(const char *source, void *destination, qlz_state_decompress *state)
{
        size_t dsiz = qlz_size_decompressed(source);

        if (state->stream_counter + qlz_size_decompressed(source) - 1 >= QLZ_STREAMING_BUFFER)
        {
                if((*source & 1) == 1)
                {
                        reset_table_decompress(state);
                        dsiz = qlz_decompress_core((const unsigned char *)source, (unsigned char *)destination, dsiz, state, (const unsigned char *)destination);
                }
                else
                {
                        memcpy(destination, source + qlz_size_header(source), dsiz);
                }
                state->stream_counter = 0;
                reset_table_decompress(state);
        }
        else
        {
                unsigned char *dst = state->stream_buffer + state->stream_counter;
                if((*source & 1) == 1)
                {
                        dsiz = qlz_decompress_core((const unsigned char *)source, dst, dsiz, state, (const unsigned char *)state->stream_buffer);
                }
                else
                {
                        memcpy(dst, source + qlz_size_header(source), dsiz);
                        reset_table_decompress(state);
                }
                memcpy(destination, dst, dsiz);
                state->stream_counter += dsiz;
        }
        return dsiz;
}
```

Note the first memcpy invocation: that does copy data from user-provided compressed file into a heap-allocated buffer for which size is also controlled by the user via the compressed file header. This allows heap corruption with user-controlled data. Potentially this means arbitrary code execution for the processes that utilize the vulnerable function - one example is xbstream with —decompress flag.

Steps to reproduce:

- Create a compressed file, e.g. with qpress from some file larger than 65535 bytes.
- Edit compressed file so that the four bytes at offset 8 are changed to be less than 0x10000, for example set to 0x1000 instead.
- Edit the file so that the byte at offset 50 is an even value to pass the test: if((*source & 1) == 1)
- Replace the bytes of actual file with some recognizable pattern, e.g. 0x41 0x42 0x43 0x44
- Add the file to an xbstream file: xbstream -c Demo.qp > Demo.xbstream
- Now try to extract with decompression using xbstream under a debugger, e.g. gdb and observe the corruption: xbstream —decompress -x < Demo.xbstream

````
head -c 100000 </dev/urandom > payload.bin

qpress payload.bin payload.qp

ls -l payload.qp -rw-r--r-- 1 me me 100107 Feb 17 18:08 payload.qp

printf '\x00\x01\x00' | dd of=payload.qp bs=1 seek=8 count=3 conv=notrunc

printf '\x10' | dd of=payload.qp bs=1 seek=49 count=1 conv=notrunc

python -c 'import sys; sys.stdout.write("A"*100040)' | dd of=payload.qp bs=1
seek=50 count=100040 conv=notrunc

xbstream-80 -c payload.qp > corrupted.xbstream

$ xbstream-80 --decompress -x < corrupted.xbstream Segmentation fault ```

Fix by prevent XtraBackup read/write outside array bounds
````

All new code of the whole pull request, including one or several
files that are either new files or modified ones, are contributed under the
BSD-new license. I am contributing on behalf of my employer Amazon Web Services, Inc.

it-percona-cla commented on Aug 19 • edited

CLA not signed yet

Thank you for your submission! We really appreciate it. Like many open source projects, we ask that you sign our Contributor License Agreement before we can accept your contribution.
You have signed the CLA already but the status is still pending? Let us recheck it.

ottok commented on Aug 19

Related to this we have also submitted PierreLvx/qpress#6

ottok commented on Aug 22

Related blog post: https://lavaux.lv/2022/08/21/qpress-file-archiver-security-update.html

Chaloff force-pushed the `github-quicklz-fix` branch from **`7c41171`** to **`2aad9cd`** 3 months ago

Compare

altmannmarcelo self-assigned this on Aug 23

altmannmarcelo self-requested a review 3 months ago

ottok commented on Sep 15

Any possibility to get a review on this one?

altmannmarcelo commented on Sep 15

Contributor

Hi **@ottok** and **@Chaloff** .

First of all, thanks for providing the patch for this issue. We have raised an internal bug to keep track of it https://jira.percona.com/browse/PXB-2854.

This issue is currently a blocker for our next release. We are in the process of working on the issues that will be part of the release and this PR will get reviewed soon.

Thanks

👍 1

altmannmarcelo commented on Oct 5

Contributor

@Chaloff I am working on reviewing this fix and merging it to our next release branch. Can you please sign the CLA agreement at #1366 (comment)

✉ **ottok** commented on Oct 5

AWS does not sign CLAs. We contribute this with the open source license of
the project.

altmannmarcelo requested changes on Oct 5

View changes

altmannmarcelo left a comment

Contributor

I will get back on the license once I hear back internally.

For now, I can see that the provided patch breaks the software functionality:

```
xtrabackup --backup --port=3306 --stream=xbstream --parallel=16 --compress --compress-threads=4 --encrypt=AES256 --encrypt-key='percona_xtrabackup_is_awesome___' --encrypt-threads=4 --encrypt-chunk-size=8K > backup.out

mkdir out

xbstream -xv --parallel=1 --decompress --decompress-threads=1 --decrypt=AES256 --encrypt-key='percona_xtrabackup_is_awesome___' --encrypt-threads=1 -C out < backup.out
```

This produces an error:

```
sys/sys_config.ibd.qp.xbcrypt
Error: compressed file was corrupted - header data size and actual data size mismatch - can’t decompress
decompress: error running decompression.
decrypt: write to destination failed.
xbstream: my_write() failed.
exit code: 1
```

altmannmarcelo commented on Oct 24

Contributor

Hi **@Chaloff** **@ottok** - Did not hear any feedback in a few weeks.
Are you interested in continue working on this PR?

Chaloff commented on Oct 24 — Author

> Hi **@Chaloff** **@ottok** - Did not hear any feedback in a few weeks. Are you interested in continue working on this PR?

Yes, sorry - was busy, will proceed with the PR this week

Chaloff force-pushed the `github-quicklz-fix` branch from **`2aad9cd`** to **`9154211`** last month — Compare

altmannmarcelo commented on Oct 28 — Contributor

Hi **@Chaloff** . I am not sure if your last force push is intended to fix the encrypt issue. I tested it and I can still see the error:

```
 ↳▶ $ xbstream -xv --parallel=1 --decompress --decompress-threads=1 --decrypt=AES256 --encrypt-key='percona_xtrabackup_is_awesome___' --encrypt-threads=1 -C out < backup.out
sys/sys_config.ibd.qp.xbcrypt
Error: compressed file was corrupted - header data size and actual data size mismatch - can't decompress
Assertion "threads[i].to_len > 0" failed at /work/pxb/src/8.0/storage/innobase/xtrabackup/src/ds_decompress.cc:241
Aborted (core dumped)
```

Chaloff commented on Oct 28 — Author

> Hi **@Chaloff** . I am not sure if your last force push is intended to fix the encrypt issue. I tested it and I can still see the error:
>
> ```
>  ↳▶ $ xbstream -xv --parallel=1 --decompress --decompress-threads=1 --decrypt=AES256 --encrypt-key='percona_xtrabackup_is_awesome___' --encrypt-threads=1 -C out < backup.out
> sys/sys_config.ibd.qp.xbcrypt
> Error: compressed file was corrupted - header data size and actual data size mismatch - can't decompress
> Assertion "threads[i].to_len > 0" failed at /work/pxb/src/8.0/storage/innobase/xtrabackup/src/ds_decompress.cc:241
> Aborted (core dumped)
> ```

Checking...

`Quicklz decompression memory corruption issue fix` … ✕ `906fec9`

Chaloff force-pushed the `github-quicklz-fix` branch from **`9154211`** to **`906fec9`** last month — Compare

altmannmarcelo changed the title ~~Quicklz decompression memory corruption issue fix~~ **PXB-2854 - Quicklz decompression memory corruption issue fix** 18 days ago

altmannmarcelo commented 18 days ago — Contributor

Hi **@Chaloff**

Using latest commit the same issue still happening:

```
 marcelo  📁 /tmp ▶
 ↳▶ $ xbstream --version
xbstream  Ver 8.0.29-22 for Linux (x86_64) (revision id: 906fec986e5)

 marcelo  📁 /tmp ▶
 ↳▶ $ xbstream -xv --parallel=1 --decompress --decompress-threads=1 --decrypt=AES256 --encrypt-key='percona_xtrabackup_is_awesome___' --encrypt-threads=1 -C out < backup.out
sys/sys_config.ibd.qp.xbcrypt
Error: compressed file was corrupted - header data size and actual data size mismatch - can't decompress
Assertion "threads[i].to_len > 0" failed at /work/pxb/src/8.0/storage/innobase/xtrabackup/src/ds_decompress.cc:241
Aborted (core dumped)
```

Chaloff commented 17 days ago — Author

> Hi **@Chaloff**
>
> Using latest commit the same issue still happening:
>
> ```
>  marcelo  📁 /tmp ▶
>  ↳▶ $ xbstream --version
> xbstream  Ver 8.0.29-22 for Linux (x86_64) (revision id: 906fec986e5)
>
>  marcelo  📁 /tmp ▶
>  ↳▶ $ xbstream -xv --parallel=1 --decompress --decompress-threads=1 --decrypt=AES256 --encrypt-key='percona_xtrabackup_is_awesome___' --encrypt-threads=1 -C out < backup.out
> sys/sys_config.ibd.qp.xbcrypt
> Error: compressed file was corrupted - header data size and actual data size mismatch - can't decompress
> Assertion "threads[i].to_len > 0" failed at /work/pxb/src/8.0/storage/innobase/xtrabackup/src/ds_decompress.cc:241
> Aborted (core dumped)
> ```

I probably need some assistance here if you don't mind. The fix in qpress are pretty simple and well tested - it just check boundaries of two arrays (source and target) before decompress. The problem seems to be in calling this qpress function - qlz_decompress(...) - we need to pass the allocated size of source and target arrays to be able to check against it. I do it like this:
`thd->to_alloc_size = decomp_file->decomp_ctxt->chunk_size; thd->from_alloc_size = qlz_size_compressed(decomp_file->header);`
Looks like it's incorrect way. Can you advise me here how to do it correctly?
Thanks in advance

Reviewers

altmannmarcelo

Assignees

altmannmarcelo

Labels

None yet

Projects

None yet

Milestone

No milestone

4 participants