Information disclosure in fuse-exfat #185

Open msuhanov opened this issue on May 1 · 2 comments

msuhanov commented on May 1

Affected versions: 1.3.0 and latest code (`11e4f03`).

Details:
In the exFAT file system, each file has a stream extension with the following fields defined (among others): DataLength and ValidDataLength (https://docs.microsoft.com/en-us/windows/win32/fileio/exfat-specification#76-stream-extension-directory-entry). The former refers to the file size, the latter refers to the highest file offset written.

According to the official exFAT specification, bytes after ValidDataLength are undefined and "[i]mplementations shall return zeroes for read operations beyond the valid data length" (https://docs.microsoft.com/en-us/windows/win32/fileio/exfat-specification#765-validdatalength-field).

In Windows, it's possible to allocate a file with DataLength larger than ValidDataLength. One needs to call the NtSetInformationFile function using the FileEndOfFileInformation argument (https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/ntddk/ns-ntddk-_file_end_of_file_information) to set DataLength larger than ValidDataLength. This call will allocate additional clusters for the file as needed but will not zero them out.

When the file handle is closed, these newly allocated clusters will be zeroed out as needed (so data between ValidDataLength and DataLength will consist of null bytes only), but while the handle is open, these clusters contain remnant (deleted) data. If the volume is detached from the system without closing the handle, the following state is observed:
a. the file has ValidDataLength smaller than DataLength;
b. clusters beyond ValidDataLength (up to DataLength) contain remnant (deleted) data.

(It's possible that more than one way to allocate clusters without zeroing them out exist.)

Attaching the same volume to the Windows system again will not wipe (zero out) clusters beyond ValidDataLength. So, there will be a file with some of its clusters containing remnant (deleted) data.

When reading such a file in WIndows, null bytes are returned for offsets beyond ValidDataLength. This behavior is expected and it matches the specification. No remnant (deleted) data is exposed as file data.

When reading such a file using fuse-exfat, remnant data is returned to the caller for offsets beyond ValidDataLength. This behavior violates the specification.

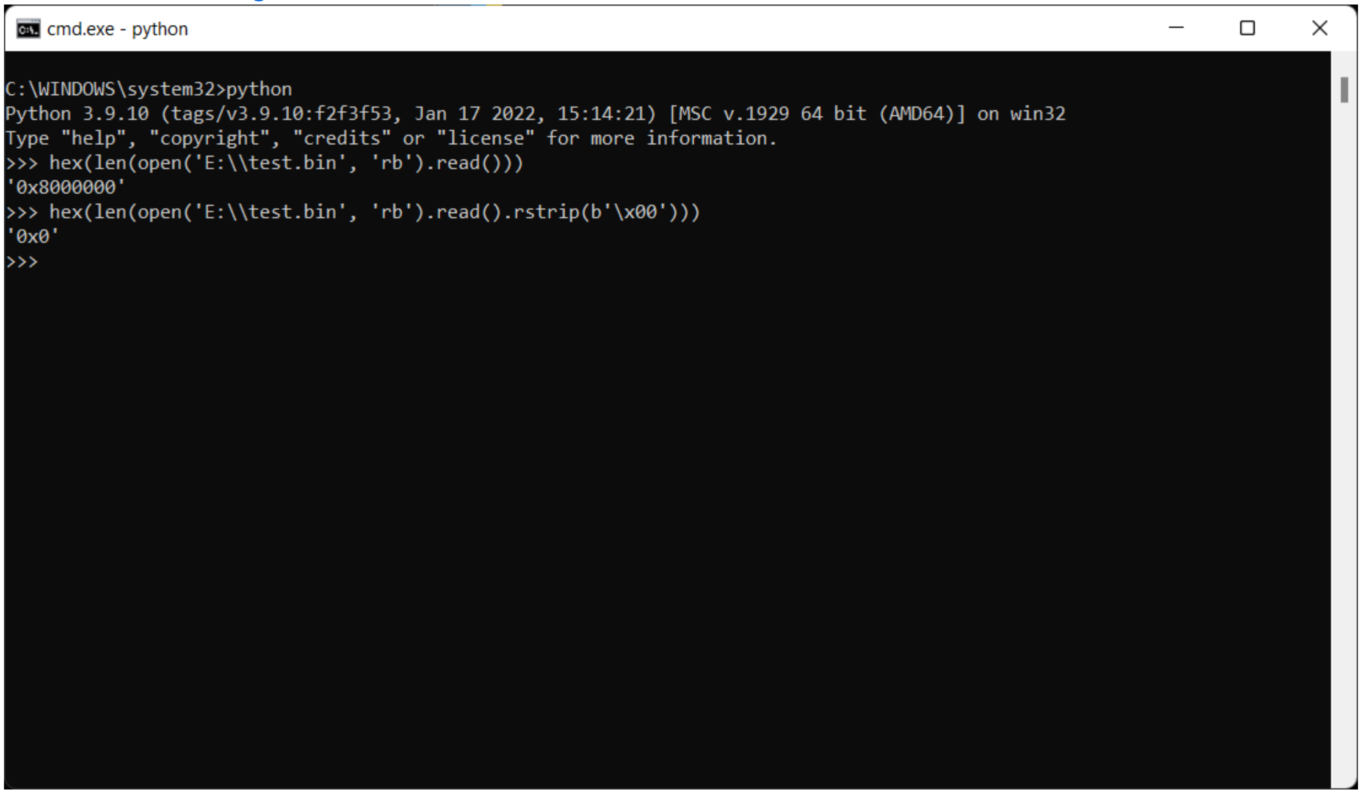
I am attaching two screenshots and a gzipped file system image to demonstrate this behavior (this image was used to produce both screenshots). The image was filled with the "PTRN" byte pattern before the format operation, so returning these bytes to the caller reading the "/test.bin" file is actually exposing remnant data (which existed before the format).

If a particular setup allows an unprivileged user to read from a mounted exFAT volume, exposing remnant (deleted) data to this user is a vulnerability (for example, a similar vulnerability is described here: https://access.redhat.com/security/cve/cve-2021-4155).

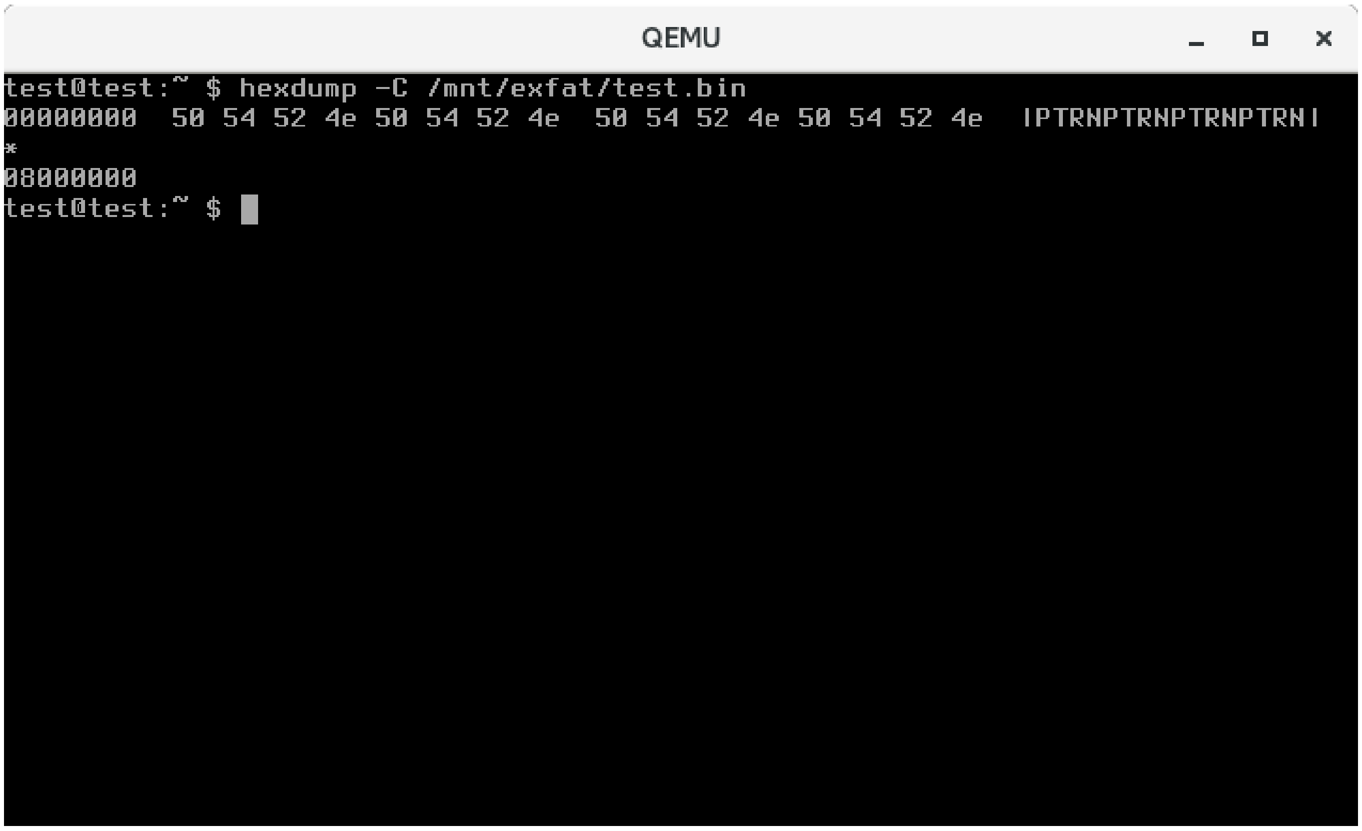
Solution: respect the ValidDataLength field when reading a file, return null bytes for offsets beyond this value.

Report date (#180): 2022-01-27.

Attached files:
exfat_allocation.raw.gz



```
C:\WINDOWS\system32>python
Python 3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022, 15:14:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> hex(len(open('E:\\test.bin', 'rb').read()))
'0x8000000'
>>> hex(len(open('E:\\test.bin', 'rb').read().rstrip(b'\x00')))
'0x0'
>>>
```



```
test@test:~ $ hexdump -C /mnt/exfat/test.bin
00000000  50 54 52 4e 50 54 52 4e  50 54 52 4e 50 54 52 4e  |PTRNPTRNPTRNPTRN|
*
08000000
test@test:~ $
```

relan commented on May 2 Owner

The fix (needs a bit of polishing): https://github.com/relan/exfat/tree/validsize

ajakk commented on May 2

Why not open a PR with the fix?

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

3 participants