Heap-buffer-overflow in ntfs_dinode_lookup #1829

Closed · **Google-Autofuzz** opened this issue on Feb 14, 2020 · 2 comments · Fixed by #1837

Google-Autofuzz commented on Feb 14, 2020

Hello sleuthkit team,

As part of our fuzzing efforts at Google, we have identified an issue affecting sleuthkit (tested with revision * develop `c05a93a`).

To reproduce, we are attaching a Dockerfile which compiles the project with LLVM, taking advantage of the sanitizers that it offers. More information about how to use the attached Dockerfile can be found here: https://docs.docker.com/engine/reference/builder/

Instructions:

```
unzip artifacts_149443351.zip
docker build --build-arg SANITIZER=address --tag=autofuzz-sleuthkit-149443351 autofuzz_149443351
docker run --entrypoint /fuzzing/repro.sh --cap-add=SYS_PTRACE -v $PWD/autofuzz_149443351/autofuzz_149443351:/tmp/poc autofuzz-sleuthkit-149443351 "fls ntfs" /tmp/poc
docker run --cap-add=SYS_PTRACE -v $PWD/autofuzz_149443351/autofuzz_149443351:/tmp/poc -it autofuzz-sleuthkit-149443351
```

Alternatively, and depending on the bug, you could use gcc, valgrind or other instrumentation tools to aid in the investigation. The sanitizer error that we encountered is here:

```
fls -- ntfs
=================================================================
==9==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x615000000500 at pc 0x000000577889 bp 0x7ffcd77ac1b0 sp 0x7ffcd77ac1a8
READ of size 1 at 0x615000000500 thread T0
    #0 0x577888 in ntfs_dinode_lookup /fuzzing/sleuthkit/tsk/fs/ntfs.c:413:17
    #1 0x57c476 in ntfs_inode_lookup /fuzzing/sleuthkit/tsk/fs/ntfs.c:2892:9
    #2 0x5328f9 in tsk_fs_file_open_meta /fuzzing/sleuthkit/tsk/fs/fs_file.c:128:9
    #3 0x579b7e in ntfs_open /fuzzing/sleuthkit/tsk/fs/ntfs.c:5217:13
    #4 0x4fc712 in main /fuzzing/sleuthkit/tools/fstools/fls.cpp:309:23
    #5 0x7feacf75f09a in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2409a)
    #6 0x4213c9 in _start (/fuzzing/sleuthkit/tools/fstools/fls+0x4213c9)

0x615000000500 is located 0 bytes to the right of 512-byte region [0x615000000300,0x615000000500)
allocated by thread T0 here:
    #0 0x4c9213 in __interceptor_malloc (/fuzzing/sleuthkit/tools/fstools/fls+0x4c9213)
    #1 0x5dc37e in tsk_malloc /fuzzing/sleuthkit/tsk/base/mymalloc.c:32:16
    #2 0x57c458 in ntfs_inode_lookup /fuzzing/sleuthkit/tsk/fs/ntfs.c:2887:25
    #3 0x5328f9 in tsk_fs_file_open_meta /fuzzing/sleuthkit/tsk/fs/fs_file.c:128:9
    #4 0x579b7e in ntfs_open /fuzzing/sleuthkit/tsk/fs/ntfs.c:5217:13
    #5 0x4fc712 in main /fuzzing/sleuthkit/tools/fstools/fls.cpp:309:23
    #6 0x7feacf75f09a in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2409a)

SUMMARY: AddressSanitizer: heap-buffer-overflow /fuzzing/sleuthkit/tsk/fs/ntfs.c:413:17 in ntfs_dinode_lookup
Shadow bytes around the buggy address:
  0x0c2a7fff8050: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2a7fff8060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c2a7fff8070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c2a7fff8080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c2a7fff8090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c2a7fff80a0:[fa]fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2a7fff80b0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2a7fff80c0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2a7fff80d0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2a7fff80e0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2a7fff80f0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
  Shadow gap:              cc
==9==ABORTING
```

We will gladly work with you so you can successfully confirm and reproduce this issue. Do let us know if you have any feedback surrounding the documentation.

Once you have reproduced the issue, we'd appreciate to learn your expected timeline for an update to be released. With any fix, please attribute the report to "Google Autofuzz project".

We are also pleased to inform you that your project is eligible for inclusion to the OSS-Fuzz project, which can provide additional continuous fuzzing, and encourage you to investigate integration options.

Don't hesitate to let us know if you have any questions!

Google AutoFuzz Team
artifacts_149443351.zip

micrictor commented on Feb 22, 2020 Contributor

The issue is in the guarding logic at

sleuthkit/tsk/fs/ntfs.c
Line 378 in 8e646f9

```
378     if (tsk_getu16(fs->endian, mft->upd_off) + sizeof(ntfs_upd) > a_ntfs->mft_rsize_b) {
```

.

I'll make a PR to correct this in the next couple days.

micrictor mentioned this issue on Feb 22, 2020

Correct guarding conditional to account for array #1837
Merged

carnil commented on Mar 9, 2020

This issue appears to be relating to CVE-2020-10233.

bcarrier closed this as completed in #1837 on Apr 1, 2020

osiewers pushed a commit to trufflepig-forensics/sleuthkit that referenced this issue on Oct 2, 2020

`Correct guarding conditional to account for array` ··· 8ad8cd6

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
Successfully merging a pull request may close this issue.
Correct guarding conditional to account for array

3 participants