



author Hawkins Jiawei <yin31149@gmail.com> 2022-09-01 00:09:38 +0800
committer Greg Kroah-Hartman <gregkh@linuxfoundation.org> 2022-11-26 09:24:52 +0100
commit [b63ddb3ba61e2d3539f87e095c881e552bc45dab](#) (patch)
tree [5e1101c335520d83f3651656216227ca11e8280b](#)
parent [ab6a1bb17e3c2f6670020d7edeea2fbfe6466690](#) (diff)
download [linux-b63ddb3ba61e2d3539f87e095c881e552bc45dab.tar.gz](#)

diff options

context: ▼
space: ▼
mode: ▼

ntfs: check overflow when iterating ATTR_RECORDs

commit 63095f4f3af59322bea984a6ae44337439348fe0 upstream.

Kernel iterates over ATTR_RECORDs in mft record in ntfs_attr_find(). Because the ATTR_RECORDs are next to each other, kernel can get the next ATTR_RECORD from end address of current ATTR_RECORD, through current ATTR_RECORD length field.

The problem is that during iteration, when kernel calculates the end address of current ATTR_RECORD, kernel may trigger an integer overflow bug in executing `a = (ATTR_RECORD*)((u8*)a + le32_to_cpu(a->length))`. This may wrap, leading to a forever iteration on 32bit systems.

This patch solves it by adding some checks on calculating end address of current ATTR_RECORD during iteration.

Link: <https://lkml.kernel.org/r/20220831160935.3409-4-yin31149@gmail.com>

Link: <https://lore.kernel.org/all/20220827105842.GM2030@kadam/>

Signed-off-by: Hawkins Jiawei <yin31149@gmail.com>

Suggested-by: Dan Carpenter <dan.carpenter@oracle.com>

Cc: Anton Altaparmakov <anton@tuxera.com>

Cc: chenxiaosong (A) <chenxiaosong2@huawei.com>

Cc: syzkaller-bugs <syzkaller-bugs@googlegroups.com>

Signed-off-by: Andrew Morton <akpm@linux-foundation.org>

Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>

Diffstat

```
-rw-r--r-- fs/ntfs/attrib.c 8
```

1 files changed, 8 insertions, 0 deletions

diff --git a/fs/ntfs/attrib.c b/fs/ntfs/attrib.c

index f171563e510657...c0881d39d36a9c 100644

--- a/fs/ntfs/attrib.c

+++ b/fs/ntfs/attrib.c

```
@@ -617,6 +617,14 @@ static int ntfs_attr_find(const ATTR_TYPE type, const ntfschar *name,
                        return -ENOENT;
                        if (unlikely(!a->length))
                            break;
+
+    /* check whether ATTR_RECORD's length wrap */
+    if ((u8 *)a + le32_to_cpu(a->length) < (u8 *)a)
+        break;
```

```
+      /* check whether ATTR_RECORD's length is within bounds */
+      if ((u8 *)a + le32_to_cpu(a->length) > mrec_end)
+          break;
+
+      if (a->type != type)
+          continue;
+  /*
```