Bug 3392810 - Stack-buffer-overflow in disasm on address 0x7ffebdad1a40 at pc 0x00000043e569 bp 0x7ffebdace900 sp 0x7ffebdace8f8

Status: OPEN

Alias: None

Product: NASM
Component: Disassembler (show other bugs)
Version: 2.16 (development)
Hardware: All All

Importance: High blocker
Assignee: nobody

URL:

Depends on:
Blocks:

Reported: 2022-09-21 01:28 PDT by xudong.c
Modified: 2022-09-21 01:29 PDT (History)
CC List: 5 users (show)

Obtained from: Built from git using configure

Attachments	
the POC file. (370 bytes, application/x-zip-compressed) 2022-09-21 01:28 PDT, xudong.c	Details
Add an attachment (proposed patch, testcase, etc.)	

Note
You need to log in before you can comment on or make changes to this bug.

xudong.c 2022-09-21 01:28:53 PDT Description

Created attachment 411850 [details]
the POC file.

Hi, developers of NASM:
I tested the binary ndisasm with my fuzzer, and a crash incurred, i.e., Stack-buffer-overflow error. The version of NASM is the latest (the newest master branch in github (https://github.com/netwide-assembler/nasm.git), version: NASM version 2.16rc0 compiled on Sep 20 2022) and the operation system is Ubuntu 18.04.6 LTS (docker). The following is the details.

```
root@1312a373d471:/fuzz-nasm/ndisasm# ./ndisasm
../out/crashes/id\:000000\,sig\:06\,src\:000003\,op\:havoc\,rep\:8\,344174
00000000  46                inc si
00000001  53                push bx
00000002  48                dec ax
00000003  06                push es
00000004  0000              add [bx+si],al
00000006  0000              add [bx+si],al
00000008  0000              add [bx+si],al
0000000A  0000              add [bx+si],al
```

```
0000000C  0000              add [bx+si],al
0000000E  D800              fadd dword [bx+si]
00000010  0000              add [bx+si],al
00000012  2BFF              sub di,di
00000014  F9                stc
00000015  006000            add [bx+si+0x0],ah
00000018  0000              add [bx+si],al
0000001A  94                xchg ax,sp
0000001B  0000              add [bx+si],al
0000001D  004953            add [bx+di+0x53],cl
00000020  47                inc di
00000021  4E                dec si
00000022  2C00              sub al,0x0
00000024  0000              add [bx+si],al
00000026  0100              add [bx+si],ax
00000028  0000              add [bx+si],al
0000002A  0800              or [bx+si],al
0000002C  0000              add [bx+si],al
0000002E  2000              and [bx+si],al
00000030  0000              add [bx+si],al
00000032  0000              add [bx+si],al
00000034  0000              add [bx+si],al
00000036  0100              add [bx+si],ax
00000038  003A              add [bp+si],bh
0000003A  0300              add ax,[bx+si]
0000003C  0000              add [bx+si],al
0000003E  0000              add [bx+si],al
00000040  0000              add [bx+si],al
00000042  0F0000            sldt [bx+si]
00000045  005356            add [bp+di+0x56],dl
00000048  5F                pop di
00000049  50                push ax
0000004A  4F                dec di
0000004B  53                push bx
0000004C  49                dec cx
0000004D  54                push sp
0000004E  49                dec cx
0000004F  4F                dec di
00000050  4E                dec si
00000051  004F53            add [bx+0x53],cl
00000054  47                inc di
00000055  4E                dec si
00000056  2C00              sub al,0x0
00000058  0000              add [bx+si],al
0000005A  0100              add [bx+si],ax
0000005C  0000              add [bx+si],al
0000005E  0800              or [bx+si],al
00000060  0000              add [bx+si],al
00000062  2000              and [bx+si],al
00000064  0000              add [bx+si],al
00000066  0100              add [bx+si],ax
00000068  0000              add [bx+si],al
0000006A  0000              add [bx+si],al
0000006C  0000              add [bx+si],al
0000006E  0300              add ax,[bx+si]
00000070  0000              add [bx+si],al
00000072  00FC              add ah,bh
00000074  0000              add [bx+si],al
00000076  0F0000            sldt [bx+si]
00000079  005356            add [bp+di+0x56],dl
0000007C  5F                pop di
0000007D  54                push sp
0000007E  41                inc cx
0000007F  52                push dx
00000080  624554            bound ax,[di+0x54]
```

```
00000083  36                 ss
===================================================================
==781089==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7ffebdad1a40
at pc 0x00000043e569 bp 0x7ffebdace900 sp 0x7ffebdace8f8
READ of size 1 at 0x7ffebdad1a40 thread T0
    #0 0x43e568 in do_ea (/fuzz-nasm/ndisasm/ndisasm+0x43e568)
    #1 0x42bd0f in matches (/fuzz-nasm/ndisasm/ndisasm+0x42bd0f)
    #2 0x41cf50 in disasm (/fuzz-nasm/ndisasm/ndisasm+0x41cf50)
    #3 0x40c89c in main (/fuzz-nasm/ndisasm/ndisasm+0x40c89c)
    #4 0x7f6d47827c86 in __libc_start_main (/lib/x86_64-linux-
gnu/libc.so.6+0x21c86)
    #5 0x406759 in _start (/fuzz-nasm/ndisasm/ndisasm+0x406759)

Address 0x7ffebdad1a40 is located in stack of thread T0 at offset 96 in frame
    #0 0x406a8f in main (/fuzz-nasm/ndisasm/ndisasm+0x406a8f)

  This frame has 6 object(s):
    [32, 96) 'buffer' <== Memory access at offset 96 overflows this variable
    [128, 136) 'ep'
    [160, 416) 'outbuf'
    [480, 484) 'synclen'
    [496, 516) 'prefer'
    [560, 561) 'rn_error'
HINT: this may be a false positive if your program uses some custom stack unwind
mechanism, swapcontext or vfork
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow (/fuzz-
nasm/ndisasm/ndisasm+0x43e568) in do_ea
Shadow bytes around the buggy address:
  0x100057b522f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100057b52300: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100057b52310: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100057b52320: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100057b52330: 00 00 00 00 00 00 00 00 00 00 00 00 f1 f1 f1 f1
=>0x100057b52340: 00 00 00 00 00 00 00 00[f2]f2 f2 f2 00 f2 f2 f2
  0x100057b52350: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100057b52360: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100057b52370: f2 f2 f2 f2 f2 f2 f2 f2 04 f2 00 00 04 f2 f2 f2
  0x100057b52380: f2 f2 01 f3 00 00 00 00 00 00 00 00 00 00 00 00
  0x100057b52390: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
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
==781089==ABORTING
```

I uploaded the POC in the attachment. Thank you for your time!

Credit
Xudong Cao (NCNIPC of China)
Han Zheng (NCNIPC of China, Hexhive)

Format For Printing - XML - Clone This Bug - Top of page