(CVE-2022-0216) QEMU LSI SCSI Use After Free

March 28, 2022 · 7 min · Muhammad Alifa Ramdhan (@n0psledbyte)

CVE: CVE-2022-0216

Tested Versions:

- QEMU < v6.0.0

Product URL(s):

- https://www.qemu.org/

Description of the vulnerability

Technical Details

The vulnerability resides in the `hw/scsi/lsi53c895a.c` specifically in `lsi_do_msgout` function. `lsi_do_msgout` function is used to receive messages from the OS, and do something based on that message. In this case, one message only has one-byte size.

```
static void lsi_do_msgout(LSIState *s)
{
    uint8_t msg;
    int len;
    uint32_t current_tag;
    lsi_request *current_req, *p, *p_next;

    if (s->current) {
        current_tag = s->current->tag;
        current_req = s->current; // [1]
    } else {
        current_tag = s->select_tag;
        current_req = lsi_find_by_tag(s, current_tag);
    }

    trace_lsi_do_msgout(s->dbc);
    while (s->dbc) { // s->dbc is controlled
        msg = lsi_get_msgbyte(s);
        s->sfbr = msg;

        switch (msg) {
                ...
        case 0x0d:
            /* The ABORT TAG message clears the current I/O process only. */
            trace_lsi_do_msgout_abort(current_tag);
            if (current_req) { // current_req = s->current
                scsi_req_cancel(current_req->req); // [1] cancel scsi request, s-
            }
            lsi_disconnect(s);
            break;
                ...
        }
    }
    return;
bad:
    qemu_log_mask(LOG_UNIMP, "Unimplemented message 0x%02x\n", msg);
    lsi_set_phase(s, PHASE_MI);
    lsi_add_msg_byte(s, 7); /* MESSAGE REJECT */
    s->msg_action = LSI_MSG_ACTION_COMMAND;
}
```

`s->current` is `lsi_request` object which is created and allocated every time we request SCSI command from OS. Using `lsi_do_msgout` , we can cancel the current SCSI request by calling the `lsi_do_msgout` function and sending `0x0d` message. By sending `0x0d` message in `lsi_do_msgout` function, it will call `scsi_req_cancel` with `current_req->req` as argument. The problem is that after the SCSI request is canceled, `s->current` will be freed, but `current_req` is not null-ed which will point to the freed buffer. Because `current_req` is freed, by sending the next message byte with `0x0d` again. `current_req->req` will have an invalid value and can lead to undefined behavior or crash.

ASAN Crash Log

```
user@nopwn:~$ sudo setpci -s 00:04.0 4.B=7
[sudo] password for user:
user@nopwn:~$ sudo insmod hello.ko
=================================================================
==85627==ERROR: AddressSanitizer: heap-use-after-free on address 0x60400003d1d0 a
READ of size 8 at 0x60400003d1d0 thread T6
    #0 0x5654fb37e3e1 in lsi_do_msgout ../../hw/scsi/lsi53c895a.c:1033
    #1 0x5654fb380f18 in lsi_execute_script ../../hw/scsi/lsi53c895a.c:1267
    #2 0x5654fb38df20 in lsi_reg_writeb ../../hw/scsi/lsi53c895a.c:1984
    #3 0x5654fb39559f in lsi_mmio_write ../../hw/scsi/lsi53c895a.c:2095
    #4 0x5654fb936452 in memory_region_write_accessor ../../softmmu/memory.c:491
    #5 0x5654fb9368dd in access_with_adjusted_size ../../softmmu/memory.c:552
    #6 0x5654fb943b74 in memory_region_dispatch_write ../../softmmu/memory.c:1502
    #7 0x5654fbb93b1c in flatview_write_continue ../../softmmu/physmem.c:2746
    #8 0x5654fbb93ef2 in flatview_write ../../softmmu/physmem.c:2786
    #9 0x5654fbb94870 in address_space_write ../../softmmu/physmem.c:2878
    #10 0x5654fbb9492c in address_space_rw ../../softmmu/physmem.c:2888
    #11 0x5654fbbd8347 in kvm_cpu_exec ../../accel/kvm/kvm-all.c:2517
    #12 0x5654fbdebd07 in kvm_vcpu_thread_fn ../../accel/kvm/kvm-accel-ops.c:49
    #13 0x5654fc4559f9 in qemu_thread_start ../../util/qemu-thread-posix.c:521
    #14 0x7fb12c7bb608 in start_thread /build/glibc-eX1tMB/glibc-2.31/nptl/pthrea
    #15 0x7fb12c6e2292 in __clone (/lib/x86_64-linux-gnu/libc.so.6+0x122292)

0x60400003d1d0 is located 0 bytes inside of 48-byte region [0x60400003d1d0,0x6040
freed by thread T6 here:
    #0 0x7fb12d90c7cf in __interceptor_free (/lib/x86_64-linux-gnu/libasan.so.5+0
    #1 0x5654fb379df6 in lsi_request_free ../../hw/scsi/lsi53c895a.c:748
    #2 0x5654fb379fac in lsi_request_cancelled ../../hw/scsi/lsi53c895a.c:757
    #3 0x5654fb1cb19f in scsi_req_cancel_complete ../../hw/scsi/scsi-bus.c:1527
    #4 0x5654fb1cbc67 in scsi_req_cancel ../../hw/scsi/scsi-bus.c:1576
    #5 0x5654fb37e3f0 in lsi_do_msgout ../../hw/scsi/lsi53c895a.c:1033
    #6 0x5654fb380f18 in lsi_execute_script ../../hw/scsi/lsi53c895a.c:1267
    #7 0x5654fb38df20 in lsi_reg_writeb ../../hw/scsi/lsi53c895a.c:1984
    #8 0x5654fb39559f in lsi_mmio_write ../../hw/scsi/lsi53c895a.c:2095
    #9 0x5654fb936452 in memory_region_write_accessor ../../softmmu/memory.c:491
    #10 0x5654fb9368dd in access_with_adjusted_size ../../softmmu/memory.c:552
    #11 0x5654fb943b74 in memory_region_dispatch_write ../../softmmu/memory.c:150
    #12 0x5654fbb93b1c in flatview_write_continue ../../softmmu/physmem.c:2746
    #13 0x5654fbb93ef2 in flatview_write ../../softmmu/physmem.c:2786
    #14 0x5654fbb94870 in address_space_write ../../softmmu/physmem.c:2878
    #15 0x5654fbb9492c in address_space_rw ../../softmmu/physmem.c:2888
    #16 0x5654fbbd8347 in kvm_cpu_exec ../../accel/kvm/kvm-all.c:2517
    #17 0x5654fbdebd07 in kvm_vcpu_thread_fn ../../accel/kvm/kvm-accel-ops.c:49
    #18 0x5654fc4559f9 in qemu_thread_start ../../util/qemu-thread-posix.c:521
    #19 0x7fb12c7bb608 in start_thread /build/glibc-eX1tMB/glibc-2.31/nptl/pthrea

previously allocated by thread T6 here:
    #0 0x7fb12d90cdc6 in calloc (/lib/x86_64-linux-gnu/libasan.so.5+0x10ddc6)
    #1 0x7fb12d418ef0 in g_malloc0 (/lib/x86_64-linux-gnu/libglib-2.0.so.0+0x57ef
    #2 0x5654fb380ef0 in lsi_execute_script ../../hw/scsi/lsi53c895a.c:1261
    #3 0x5654fb38df20 in lsi_reg_writeb ../../hw/scsi/lsi53c895a.c:1984
    #4 0x5654fb39559f in lsi_mmio_write ../../hw/scsi/lsi53c895a.c:2095
    #5 0x5654fb936452 in memory_region_write_accessor ../../softmmu/memory.c:491
    #6 0x5654fb9368dd in access_with_adjusted_size ../../softmmu/memory.c:552
    #7 0x5654fb943b74 in memory_region_dispatch_write ../../softmmu/memory.c:1502
    #8 0x5654fbb93b1c in flatview_write_continue ../../softmmu/physmem.c:2746
    #9 0x5654fbb93ef2 in flatview_write ../../softmmu/physmem.c:2786
    #10 0x5654fbb94870 in address_space_write ../../softmmu/physmem.c:2878
    #11 0x5654fbb9492c in address_space_rw ../../softmmu/physmem.c:2888
    #12 0x5654fbbd8347 in kvm_cpu_exec ../../accel/kvm/kvm-all.c:2517
    #13 0x5654fbdebd07 in kvm_vcpu_thread_fn ../../accel/kvm/kvm-accel-ops.c:49
    #14 0x5654fc4559f9 in qemu_thread_start ../../util/qemu-thread-posix.c:521
    #15 0x7fb12c7bb608 in start_thread /build/glibc-eX1tMB/glibc-2.31/nptl/pthrea

Thread T6 created by T0 here:
    #0 0x7fb12d839805 in pthread_create (/lib/x86_64-linux-gnu/libasan.so.5+0x3a8
    #1 0x5654fc455e59 in qemu_thread_create ../../util/qemu-thread-posix.c:558
    #2 0x5654fbdec1a3 in kvm_start_vcpu_thread ../../accel/kvm/kvm-accel-ops.c:73
    #3 0x5654fbc2dd81 in qemu_init_vcpu ../../softmmu/cpus.c:628
    #4 0x5654fb7481ad in x86_cpu_realizefn ../../target/i386/cpu.c:6910
    #5 0x5654fc2260e1 in device_set_realized ../../hw/core/qdev.c:761
    #6 0x5654fbe61f58 in property_set_bool ../../qom/object.c:2257
    #7 0x5654fbe5c74d in object_property_set ../../qom/object.c:1402
    #8 0x5654fbe653b8 in object_property_set_qobject ../../qom/qom-qobject.c:28
    #9 0x5654fbe5cd70 in object_property_set_bool ../../qom/object.c:1472
    #10 0x5654fc222a27 in qdev_realize ../../hw/core/qdev.c:389
    #11 0x5654fb700c39 in x86_cpu_new ../../hw/i386/x86.c:111
    #12 0x5654fb700f35 in x86_cpus_init ../../hw/i386/x86.c:138
    #13 0x5654fb671cfb in pc_init1 ../../hw/i386/pc_piix.c:159
    #14 0x5654fb674c04 in pc_init_v6_1 ../../hw/i386/pc_piix.c:427
    #15 0x5654fae1d754 in machine_run_board_init ../../hw/core/machine.c:1237
    #16 0x5654fbc0d1be in qemu_init_board ../../softmmu/vl.c:2514
    #17 0x5654fbc0d598 in qmp_x_exit_preconfig ../../softmmu/vl.c:2588
    #18 0x5654fbc12cb8 in qemu_init ../../softmmu/vl.c:3611
    #19 0x5654faac6554 in main ../../softmmu/main.c:49
    #20 0x7fb12c5e70b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x27

SUMMARY: AddressSanitizer: heap-use-after-free ../../hw/scsi/lsi53c895a.c:1033 in
Shadow bytes around the buggy address:
  0x0c087ffff9e0: fa fa fd fd fd fd fd fd fa fa fd fd fd fd fd fd
  0x0c087ffff9f0: fa fa fd fd fd fd fd fd fa fa fd fd fd fd fd fd
  0x0c087ffffa00: fa fa fd fd fd fd fd fd fa fa fd fd fd fd fd fd
  0x0c087ffffa10: fa fa fd fd fd fd fd fd fa fa fd fd fd fd fd fd
  0x0c087ffffa20: fa fa fd fd fd fd fd fd fa fa fd fd fd fd fd fd
=>0x0c087ffffa30: fa fa fd fd fd fd fd fd fa fa[fd]fd fd fd fd fd
  0x0c087ffffa40: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c087ffffa50: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c087ffffa60: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c087ffffa70: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c087ffffa80: 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
==85627==ABORTING
```

Exploitability

To exploit this kind of bug (use after free), we must overwrite the freed value by reallocating (malloc) heap memory with the same size with chunk's size, which we want to overwrite. In this case, we must allocate memory after `scsi_req_cancel` , but unfortunately there's no malloc primitive in this function. The only way to exploit this bug is by using another thread to spray the heap, and hope after `current_req` is freed, another thread will immediately allocating the freed chunk and overwrite `current_req` content/object, if we can control `current_req` content we can make a fake structure and can lead to code execution. But unfortunately, the idea to spray the heap using another thread is likely impossible, because mechanism in QEMU when handle incoming IO/MMIO request (we can read this mechanism in `prepare_mmio_access` in `softmmu/physmem.c`). The current mechanism is VCPU thread which handle LSI SCSI operation, can't run at the same time with another VCPU thread which we made to do heap spray. Because of this restriction, we can't control freed object and exploiting this bug is likely impossible.

Requirement

- Attacker need to run as high-privileged user to load kernel module
- Using LSI SCSI with command `-device lsi53c810,id=scsi -device scsi-hd,drive=SysDisk -drive id=SysDisk,if=none,file=./disk.img`

Proof Of Concept

PoC to trigger crash

- Host OS : Ubuntu
- Guest OS : Ubuntu
- Example QEMU command line used:

```
qemu-system-x86_64 \
  -kernel ./bzImage \
  -m 3G -smp 4\
  -drive file=./groovy-debootsrap.ext2.qcow2 \
  -net nic -net user,hostfwd=tcp::2222-:22 \
  -append "root=/dev/sda console=ttyS0 nokaslr" \
  -device lsi53c810,id=scsi -device scsi-hd,drive=SysDisk -drive id=SysDisk,if=none,f
  -nographic -enable-kvm -display none
```

- Run:

```
$ sudo setpci -s 00:04.0 4.B=7 # enable master bit for lsi scsi pci device
$ sudo insmod hello.ko
```

Mitigations

- Since the PoC must be run at high-privileged on the guest OS, Do not run untrusted code or driver in guest OS.

Timeline:

- 2021-12-28 Vendor disclosure
- 2022-03-28 Vendor patched

« PREV
(CVE-2022-0168) Linux Kernel smb2_ioctl_query_info NULL Pointer Dereference

NEXT »
(CVE-2022-28730) Apache JSPWiki v2.11.1 - Reflected XSS in AjaxPreview.jsp