[<prev] [next>] [thread-next>] [day] [month] [year] [list]

```
Date: Thu, 7 Apr 2022 10:15:42 +0800 (GMT+08:00)
From: kangel <kangel@....edu.cn>
To: oss-security@...ts.openwall.com
Cc: pgn@....edu.cn, qiuhao@...ec.org
Subject: Linux kernel: x86/kvm: null-ptr-deref in kvm_dirty_ring_push
```

-----原始邮件-----
发件人:kangel <kangel@....edu.cn>
发送时间:2022-04-06 21:11:39 （星期三）
收件人： security@...nel.org, linux-distros@...openwall.org
抄送： secalert@...hat.com, pbonzini@...hat.com, pgn@....edu.cn, qiuhao@...ec.org
主题： [vs] x86/kvm: null-ptr-deref in kvm_dirty_ring_push

Hi developers,

We found a null-ptr-deref in the kvm module which can lead to DoS. This flaw is in kvm_dirty_ring_push in virt/kvm/dirty_ring.c. The linux kernel version is 5.17.0-rc8. We would appreciate a CVE ID if this is a security issue.

------------[Description]------------

When we call kvm_vcpu_release(), it will call kvm_dirty_ring_free() which will free ring->dirty_gfns and set it to NULL. Then if we can set kvm->dirty_ring_size != NULL[1] and make vcpu->arch.st.preempt to NULL[2], it will call kvm_dirty_ring_push() and lead to null-ptr-deref in virt/kvm/dirty_ring.c:159.

The condition of [1] can be set by do ioctl$KVM_CAP_DIRTY_LOG_RING and the condition of [2] can be set by race of doing ioctf$KVM_RUN.

------------[Reproducer]------------

qemu run:

```
qemu-system-x86_64 -m 512M -smp 2 -kernel /home/zju/linux-5.17-rc8/arch/x86/boot/bzImage -append
"console=ttyS0 root=/dev/sda earlyprintk=serial net.ifnames=0 nokaslr" -drive
file=/home/zju/script/stretch2.img,format=raw -net user,host=10.0.2.10,hostfwd=tcp:127.0.0.1:10021-:22 -net
nic,model=e1000 -enable-kvm -nographic
```

poc.c is attached(run in qemu).

```
gcc poc.c -static -o poc -lpthread
```

------------[Credits]------------

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

------------[Backtrace]------------

```
KASAN: null-ptr-deref in range [0x0000000000000030-0x0000000000000037]
CPU: 0 PID: 453 Comm: syz-executor425 Not tainted 5.17.0 #3
Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS 1.10.2-1ubuntu1~cloud0 04/01/2014
RIP: 0010:kvm_dirty_ring_push+0x10c/0x2e0 arch/x86/kvm/../../../virt/kvm/dirty_ring.c:159
Code: 0f 8e 8e 01 00 00 48 b8 00 00 00 00 00 fc ff df 41 83 ec 01 44 23 65 00 49 c1 e4 04 4c 01 e3 48 8d 7b
04 48 89 fa 48 c1 ea 03 <0f> b6 14 02 48 89 f8 83 e0 07 83 c0 03 38 d0 7c 08 84 d2 0f 85 47
RSP: 0018:ffff88800812fb88 EFLAGS: 00010207
RAX: dffffc0000000000 RBX: 0000000000000030 RCX: ffffffffa5a929d4
RDX: 0000000000000006 RSI: 0000000000000000 RDI: 0000000000000034
RBP: ffff888004d12118 R08: 0000000000000001 R09: fffffbfff5104469
R10: 0000000000000000 R11: fffffbfff5104468 R12: 0000000000000030
R13: 0000000000000000 R14: 0000000000000000 R15: ffff888004d10dd0
FS:  0000000000868880(0000) GS:ffff88806d200000(0000) knlGS:0000000000000000
CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
CR2: 0000000020ffe010 CR3: 0000000005ba4002 CR4: 00000000003726f0
DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
DR3: 0000000000000000 DR6: 00000000fffe0ff0 DR7: 0000000000000400
Call Trace:
 <TASK>
 mark_page_dirty_in_slot+0x192/0x270 arch/x86/kvm/../../../virt/kvm/kvm_main.c:3171
 kvm_steal_time_set_preempted arch/x86/kvm/x86.c:4600 [inline]
 kvm_arch_vcpu_put+0x34e/0x5b0 arch/x86/kvm/x86.c:4618
 vcpu_put+0x1b/0x70 arch/x86/kvm/../../../virt/kvm/kvm_main.c:211
 vmx_free_vcpu+0xcb/0x130 arch/x86/kvm/vmx/vmx.c:6985
 kvm_arch_vcpu_destroy+0x76/0x290 arch/x86/kvm/x86.c:11219
 kvm_vcpu_destroy arch/x86/kvm/../../../virt/kvm/kvm_main.c:441 [inline]
 kvm_destroy_vcpus+0x119/0x280 arch/x86/kvm/../../../virt/kvm/kvm_main.c:460
 kvm_free_vcpus arch/x86/kvm/x86.c:11659 [inline]
 kvm_arch_destroy_vm+0x22a/0x380 arch/x86/kvm/x86.c:11769
 kvm_destroy_vm arch/x86/kvm/../../../virt/kvm/kvm_main.c:1217 [inline]
 kvm_put_kvm+0x3ff/0x900 arch/x86/kvm/../../../virt/kvm/kvm_main.c:1250
 kvm_vcpu_release+0x4d/0x70 arch/x86/kvm/../../../virt/kvm/kvm_main.c:3668
 __fput+0x21b/0x940 fs/file_table.c:317
 task_work_run+0xde/0x180 kernel/task_work.c:164
 tracehook_notify_resume include/linux/tracehook.h:188 [inline]
 exit_to_user_mode_loop kernel/entry/common.c:175 [inline]
 exit_to_user_mode_prepare+0x14d/0x150 kernel/entry/common.c:207
 __syscall_exit_to_user_mode_work kernel/entry/common.c:289 [inline]
 syscall_exit_to_user_mode+0x1d/0x40 kernel/entry/common.c:300
 do_syscall_64+0x48/0x90 arch/x86/entry/common.c:86
 entry_SYSCALL_64_after_hwframe+0x44/0xae
------------[ Patch ]------------
We try to do a patch, which can not make the poc trigger this flaw.
diff --git a/virt/kvm/dirty_ring.c b/virt/kvm/dirty_ring.c.patch
index 222ecc8..38f1b66 100644
--- a/virt/kvm/dirty_ring.c
+++ b/virt/kvm/dirty_ring.c.patch
@@ -154,6 +154,8 @@ void kvm_dirty_ring_push(struct kvm_dirty_ring *ring, u32 slot, u64 offset)
        /* It should never get full */
        WARN_ON_ONCE(kvm_dirty_ring_full(ring));

+       if (!ring->dirty_gfns)
+               return;
        entry = &ring->dirty_gfns[ring->dirty_index & (ring->size - 1)];

        entry->slot = slot;


------------[ Cut here ]------------
C repro and kernel config are attached.
Best regards.
    Yongkang Jia of Zhejiang University
```

Content of type "text/html" **skipped**

View attachment "poc.c" **of type** "text/plain" **(6534 bytes)**

Download attachment "config" **of type** "application/octet-stream" **(130642 bytes)**