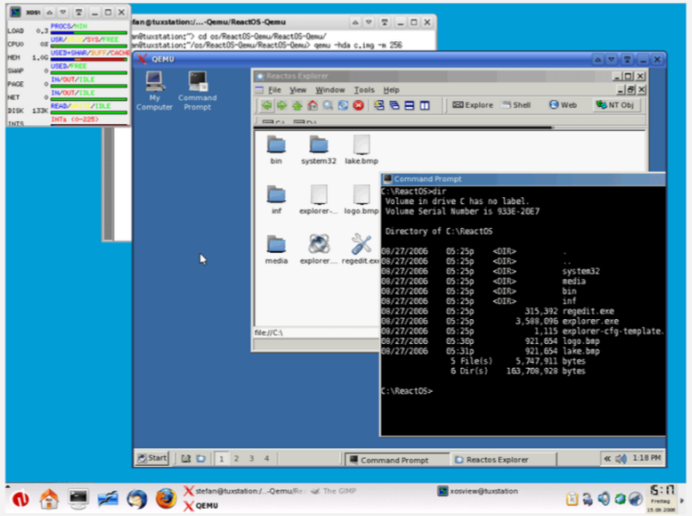
Qemu Virtual Machine Escape

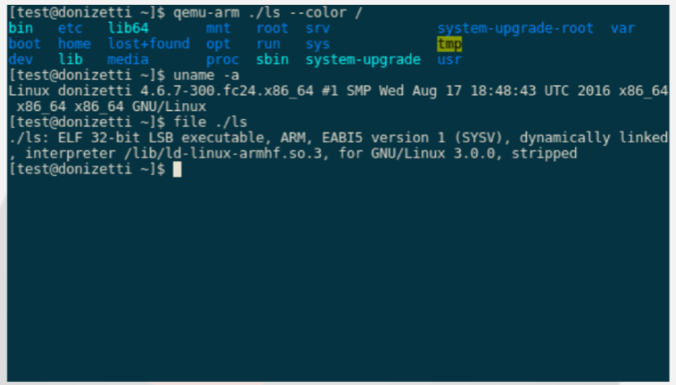
Atum

What is qemu?



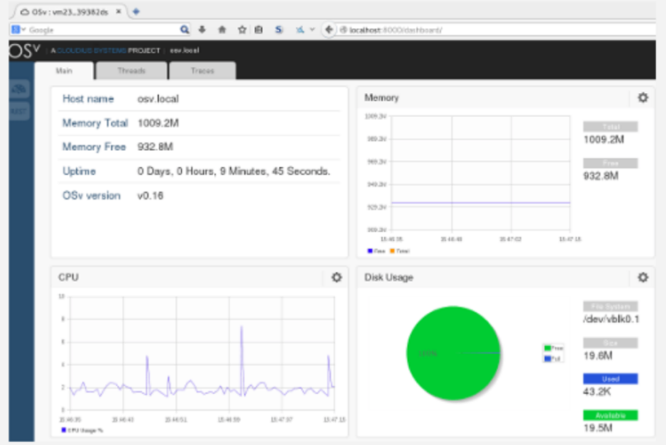
Full-system emulation

Run operating systems for any machine, on any supported architecture



User-mode emulation

Run programs for another Linux/BSD target, on any supported architecture



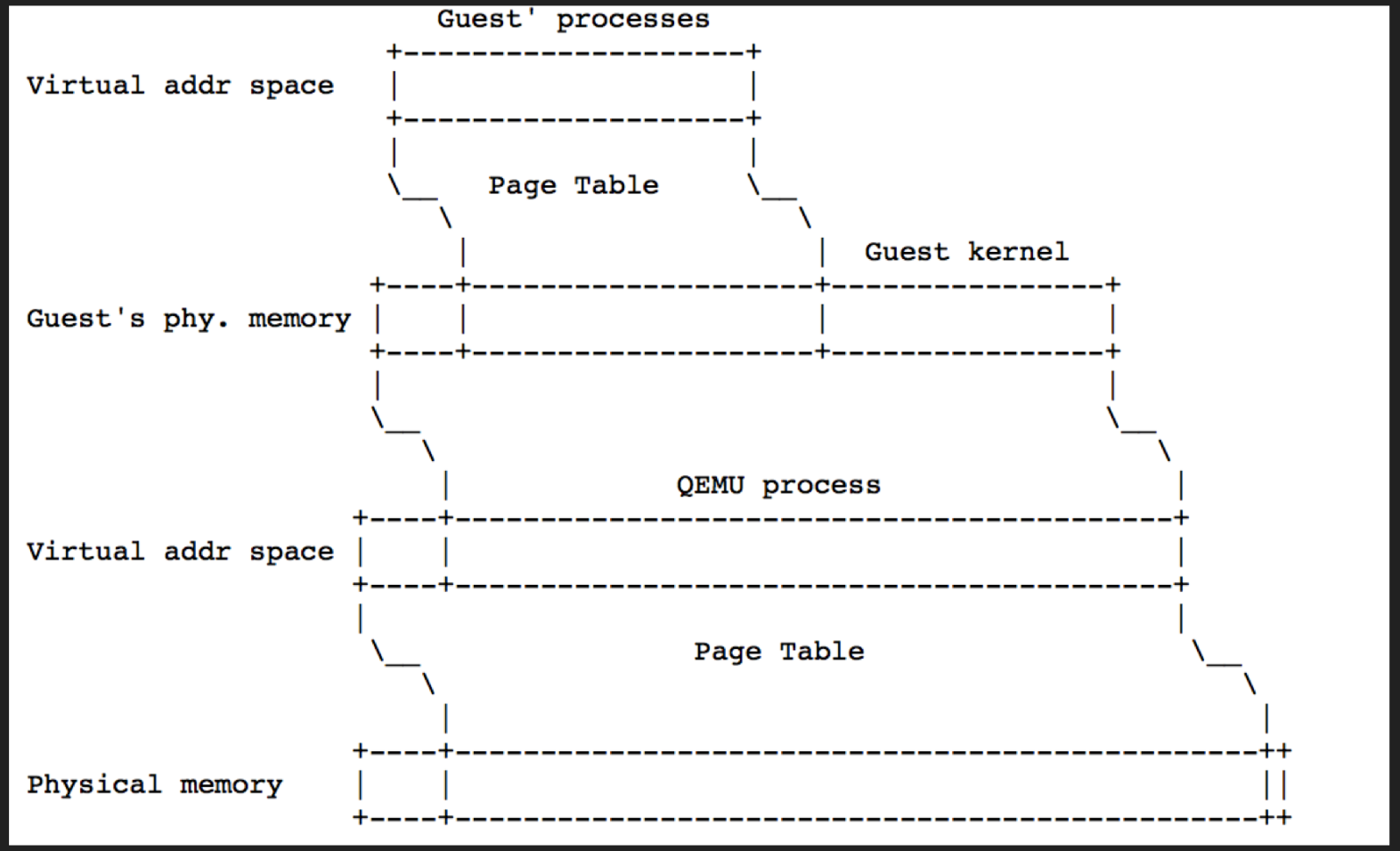
Virtualization

Run KVM and Xen virtual machines with near native performance

A free and open-source hosted hypervisor

Qemu Memory layout

- proc/[pid of qemu]/maps
- Additionally, QEMU reserves a memory region for BIOS and ROM.



Attack surface of QEMU

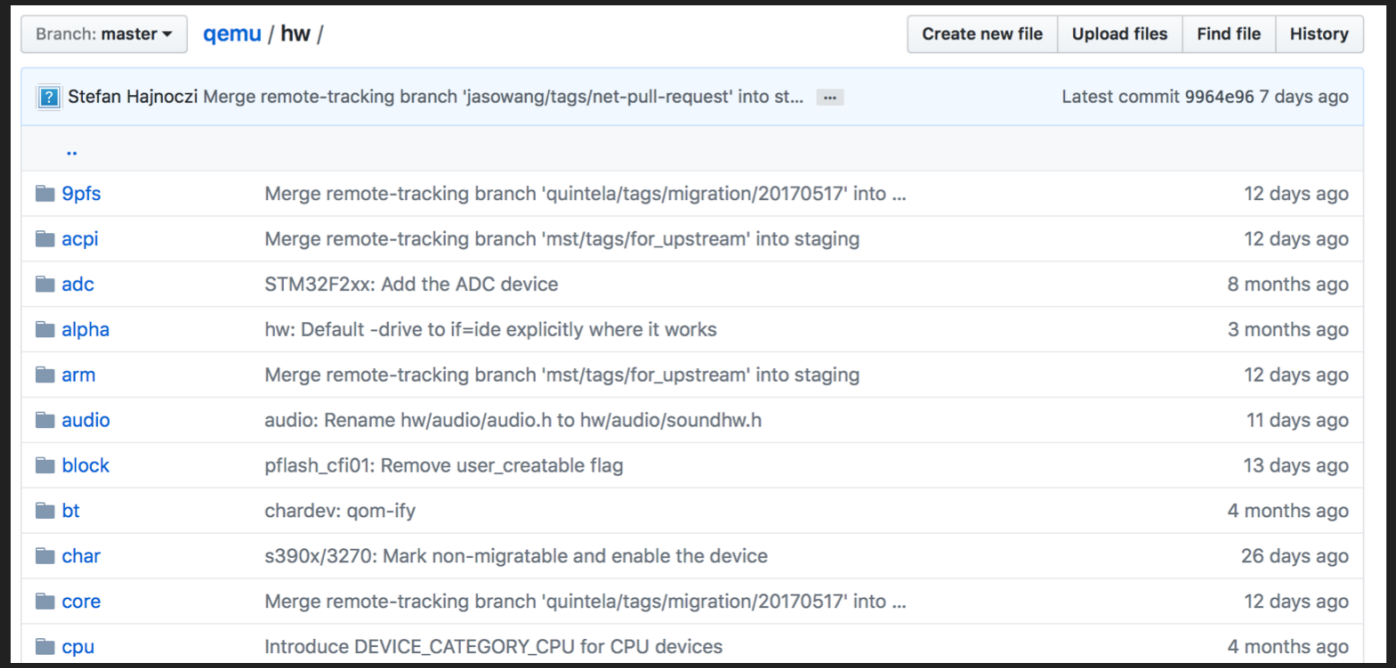
- Codes
  - Run at host
  - Communicate with guest
- Especially for Virtual Devices

```
➜  ~ qemu-system-x86_64 -device help
Controller/Bridge/Hub devices:
name "i82801b11-bridge", bus PCI
name "ioh3420", bus PCI, desc "Intel IOH device id 3420 PCIE Root Port"
name "pci-bridge", bus PCI, desc "Standard PCI Bridge"
name "pci-bridge-seat", bus PCI, desc "Standard PCI Bridge (multiseat)"
name "pcie-root-port", bus PCI, desc "PCI Express Root Port"
name "pxb", bus PCI, desc "PCI Expander Bridge"
name "pxb-pcie", bus PCI, desc "PCI Express Expander Bridge"
name "usb-hub", bus usb-bus
name "vfio-pci-igd-lpc-bridge", bus PCI, desc "VFIO dummy ISA/LPC bridge for IGD
 assignment"
name "x3130-upstream", bus PCI, desc "TI X3130 Upstream Port of PCI Express Swit
ch"
name "xio3130-downstream", bus PCI, desc "TI X3130 Downstream Port of PCI Expres
s Switch"

USB devices:
name "ich9-usb-ehci1", bus PCI
name "ich9-usb-ehci2", bus PCI
name "ich9-usb-uhci1", bus PCI
name "ich9-usb-uhci2", bus PCI
name "ich9-usb-uhci3", bus PCI
name "ich9-usb-uhci4", bus PCI
name "ich9-usb-uhci5", bus PCI
name "ich9-usb-uhci6", bus PCI
```

Select a target virtual device

- List available virtual devices:
  - qemu-system-x86_64 -device help
- Select a device as target
- Find source code:
  - qemu/hw/
- example:
  - qemu/hw/block/floppy.c
  - qemu/hw/net/pcnet.c



Branch: master ▾ qemu / hw / Create new file Upload files Find file History

Stefan Hajnoczi Merge remote-tracking branch 'jasowang/tags/net-pull-request' into st... Latest commit 9964e96 7 days ago

..

9pfs	Merge remote-tracking branch 'quintela/tags/migration/20170517' into ...	12 days ago
acpi	Merge remote-tracking branch 'mst/tags/for_upstream' into staging	12 days ago
adc	STM32F2xx: Add the ADC device	8 months ago
alpha	hw: Default -drive to if=ide explicitly where it works	3 months ago
arm	Merge remote-tracking branch 'mst/tags/for_upstream' into staging	12 days ago
audio	audio: Rename hw/audio/audio.h to hw/audio/soundhw.h	11 days ago
block	pflash_cfi01: Remove user_creatable flag	13 days ago
bt	chardev: qom-ify	4 months ago
char	s390x/3270: Mark non-migratable and enable the device	26 days ago
core	Merge remote-tracking branch 'quintela/tags/migration/20170517' into ...	12 days ago
cpu	Introduce DEVICE_CATEGORY_CPU for CPU devices	4 months ago

Communicate with virtual devices

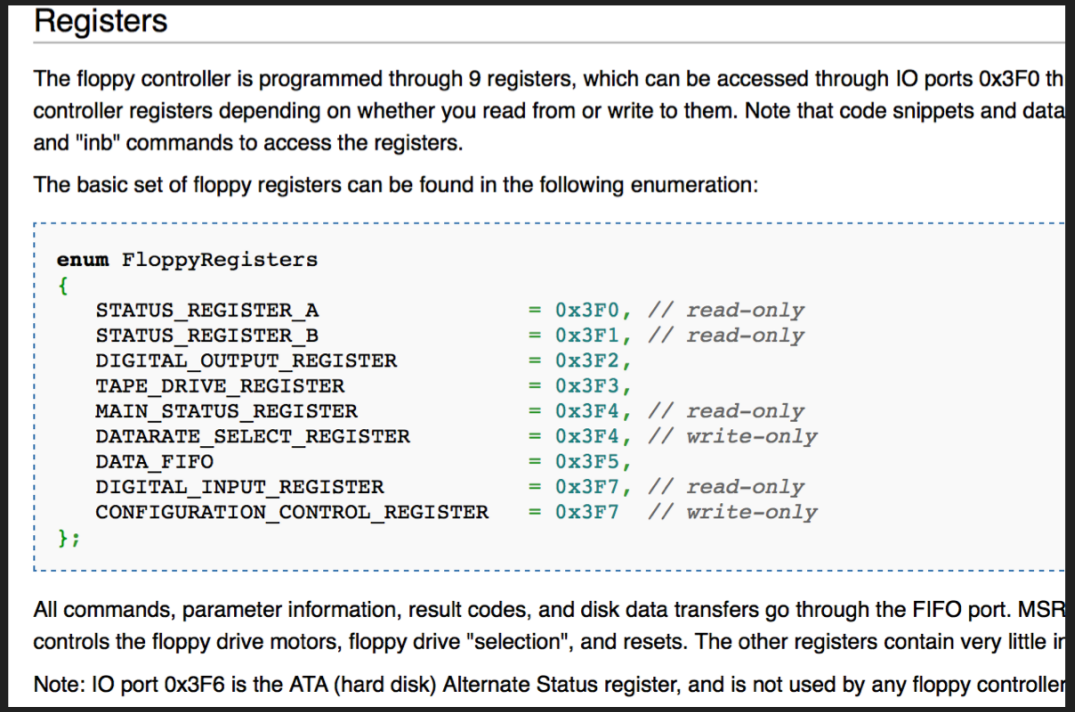
- Find the IO port && IO MEM of Virtual Device
  - lspci -nnv
  - lshw
  - lspci && cat /proc/[ioport,iomem] | **grep xx:xx:xx**
  - modinfo driver_kernel_mod
  - Read source code OR document
- Communicate by IO port && IO MEM
  - Use In[b/w/l]&&out[b/w/l] from sys/io.h
  - Use ioremap at kernel module

```
00:0f.0 VGA compatible controller [0300]: VMware SVGA II Adapter [15ad:0405] (pr
og-if 00 [VGA controller])
        Subsystem: VMware SVGA II Adapter [15ad:0405]
        Flags: bus master, medium devsel, latency 64, IRQ 16
        I/O ports at 1070 [size=16]
        Memory at e8000000 (32-bit, prefetchable) [size=128M]
        Memory at fe000000 (32-bit, non-prefetchable) [size=8M]
        [virtual] Expansion ROM at c0000000 [disabled] [size=32K]
        Capabilities: <access denied>
        Kernel driver in use: vmwgfx
        Kernel modules: vmwgfx
```

```c
#include <sys/io.h>
#define FIFO 0x3f5
int main() {
        int i;
        iopl(3);
        outb(0x8e,0x3f5); /* READ ID */
        for (i=0;i<10000000;i++)
                outb(0x42,0x3f5); /* push */
 }
```

Auditing && Fuzzing

- Reading Device Document
- Reading Source code
- Auditing OR Fuzzing as you like
- For example:
  - http://wiki.osdev.org/Floppy_Disk_Controller
  - https://github.com/qemu/qemu/blob/master/hw/block/fdc.c



Registers

The floppy controller is programmed through 9 registers, which can be accessed through IO ports 0x3F0 th
controller registers depending on whether you read from or write to them. Note that code snippets and data
and "inb" commands to access the registers.

The basic set of floppy registers can be found in the following enumeration:

```
enum FloppyRegisters
{
   STATUS_REGISTER_A                = 0x3F0, // read-only
   STATUS_REGISTER_B                = 0x3F1, // read-only
   DIGITAL_OUTPUT_REGISTER          = 0x3F2,
   TAPE_DRIVE_REGISTER              = 0x3F3,
   MAIN_STATUS_REGISTER             = 0x3F4, // read-only
   DATARATE_SELECT_REGISTER         = 0x3F4, // write-only
   DATA_FIFO                        = 0x3F5,
   DIGITAL_INPUT_REGISTER           = 0x3F7, // read-only
   CONFIGURATION_CONTROL_REGISTER   = 0x3F7  // write-only
};
```

All commands, parameter information, result codes, and disk data transfers go through the FIFO port. MSR
controls the floppy drive motors, floppy drive "selection", and resets. The other registers contain very little ir

Note: IO port 0x3F6 is the ATA (hard disk) Alternate Status register, and is not used by any floppy controller

Exploit

- Leak address layout of Qemu process
  - Libc base, .text base, guest phyaddr base etc
  - phyaddr is sometime necessary because we are communicate with virtual device
- Control flow hijack
  - Function table/call back(for example irq)
- RESET NX bit && run shellcode
  - Maybe you should clear MADV_DONTFORK flag to get a shell

Example1: venom(CVE-2015-3456)

- Find Document(OSDev, googling)
- READ Document, Know what device do
- Read diff, find vuneribility (heap overflow)
- Read Source Code
- Craft POC
- http://blogs.360.cn/blog/venom-毒液漏洞分析（qemu-kvm-cve-2015-3456）/

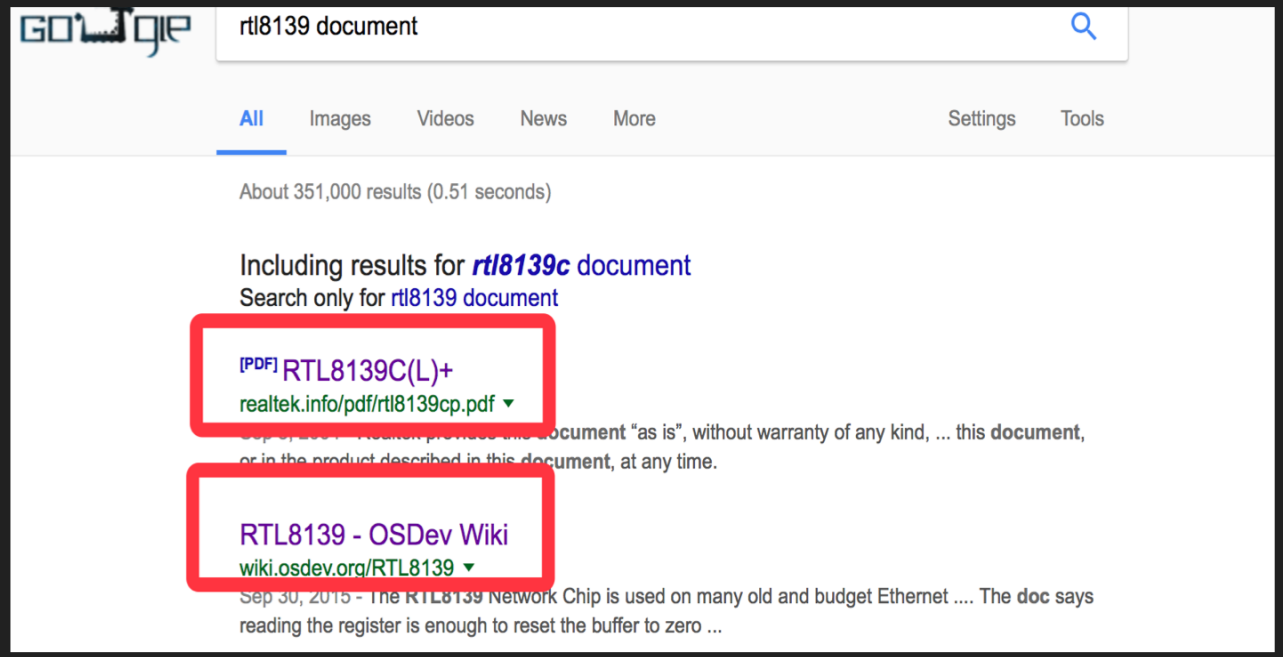
The Proper Way to issue a command

1. Read MSR (port 0x3F4).
2. Verify that RQM = 1 and DIO = 0 ((Value & 0xc0) == 0x80) -- if not, reset the controller and star
3. Send your chosen command byte to the FIFO port (port 0x3F5).
4. In a loop: loop on reading MSR until RQM = 1. Verify DIO = 0, then send the next parameter by
5. Either Execution or Result Phase begins when all parameter bytes have been sent, depending
   not perform read/write/head movement operations, skip to the Result Phase.
6. (In PIO Mode Execution phase) read MSR, verify NDMA = 1 ((Value & 0x20) == 0x20) -- if it's n
7. begin a loop:
8. Either poll MSR until RQM = 1, or wait for an IRQ6, using some waiting method.
9. In an inner loop: transfer a byte in or out of the FIFO port via a system buffer, then read MSR. F
10. if NDMA = 1, loop back to the beginning of the outer loop, unless your data buffer ran out (detec
11. Result Phase begins. If the command does not have a Result phase, it silently exits to waiting f
12. If using DMA on a read/write command, wait for a terminal IRQ6.
    1. Loop on reading MSR until RQM = 1, verify that DIO = 1.
    2. In a loop: read the next result byte from the FIFO, loop on reading MSR until RQM = 1, v
13. After reading all the expected result bytes: check them for error conditions, verify that RQM = 1

Note: implementing a failure timeout for each loop and the IRQ is pretty much required -- it is the only

Example2:CVE-2015-5165&&CVE-2015-7504

- Find Document(OS Dev Googling)
  - http://wiki.osdev.org/RTL8139
  - http://realtek.info/pdf/rtl8139cp.pdf
  - http://wiki.osdev.org/AMD_PCNET
- Read diff, find vuneribility
- Read Source code
- Craft POC
- Exploit
- http://www.phrack.org/papers/vm-escape-qemu-case-study.html



References

- http://www.phrack.org/papers/vm-escape-qemu-case-study.html
- http://blogs.360.cn/blog/venom-毒液漏洞分析（qemu-kvm-cve-2015-3456）/
- http://www.tuicool.com/articles/MzqYbia
- http://www.freebuf.com/vuls/87673.html
- http://www.powerofcommunity.net/poc2016/wei.pdf
- https://nelhage.com/talks/kvm-defcon-2011.pdf
- http://mgalgs.github.io/2015/05/16/how-to-build-a-custom-linux-kernel-for-qemu-2015-edition.html
- https://conference.hitb.org/hitbsecconf2016ams/materials/D1T1%20-%20Tang%20Qing%20Hao%20-%20Virtualization%20System%20Vulnerability%20Discovery%20Framework.pdf