Fuzzing Linux with Xen
Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.
# whoami

Xen Project

./DRAKVUF®
Black-box Binary Analysis System

LibVMI
Virtual Machine Introspection
Fast, Portable, Simple
We fuzzed the device-facing input points of several Linux drivers

We built new tools to get it done

We open-sourced them

We found a bunch of bugs & fixed them

Showing you how it was done
feedback fuzzers

Not “just” about feeding random input to your target

Feedback fuzzers monitor the target execution

Collect execution log (aka. coverage)

Compare execution from run-to-run

New code discovered? Focus on that input!
feedback fuzzers

Need determinism

If your target code behaves differently between executions not due to the fuzzer, the feedback will be just noise
Add determinism to kernel-code execution

Reset vCPU registers and free copied pages for better speed
vm forking speed

VM fork creation time:

\[ \sim 745 \, \mu s \sim= 1300 \, \text{VM/s} \]

VM fork reset time:

\[ \sim 111 \, \mu s \sim= 9000 \, \text{reset/s} \]

Measured on i5-8350U
Xen VM introspection is natively supported

- Read/write/translate guest memory

Pause vCPU & get notification in dom0 of various hw events

- CPUID
- Int3
- MTF
- EPT faults
- register writes (mov-to-cr, MSRs)
- etc.
Upstreamed in Xen 4.15

• In collaboration with CERT.pl & Citrix

Use Intel® Processor Trace to record VM execution

• CPU itself stores info in memory about execution, low overhead
• Stores only enough detail to allow for post-execution reconstruction

Can record execution of the whole VM
fuzzing on xen

Parent VM → Magic CPUID → Fork → Sink VM → Add 0xCC → panic() → kasan_report() → ubsan_prologue() → ...

Fork

Fuzz VM

Reset

Decode IPT log
Report coverage

Analyze

Catch a breakpoint
Report crash
Catch Magic CPUID
Report no-crash
Neither: timeout

Run

Read input from AFL
Write it to VM's mem

Fuzz

Write it to VM's mem
Disk /dev/loop5: 90.6 MiB, 94986240 bytes, 185520 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/loop6: 64.79 MiB, 67915776 bytes, 132648 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/loop7: 65.1 MiB, 68259840 bytes, 133320 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/xvda: 40 GB, 42949672960 bytes, 83886080 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x17cc52b9

Device Boot Start  End  Sectors Size Id Type
/dev/xvda1  0 1058623 1048576 512M b hfs+ FAT32
/dev/xvda2 1052670 83884031 82831362 39.5G 5 Extended
/dev/xvda5 1052672 83884031 82831360 39.5G 83 Linux

root@t2:/shared/xhci_demo# cat ubuntu-20.04.cfg
arch = "x86_64"
name = "ubuntu-20.04"
maxmem = 1024
memory = 1024
vcpuus = 1
maxcpuus = 1
nomigrate = 1
boot = "cd"
type = "hvm"
serial = "pty"

vif = ['brdige=xenbr0,mac=00:07:5B:8C:00:00']
disk = ['phy:/dev/t2vg/ubuntu-20.04,xvda', ""

vmtrace_buf_kb=65536
device_model_args = [
"-device", "gpmu-xhci,id=gpmu-xhci,p3=8",
"-drive", "file=/tmp/usbstick,if=none,id=usbstick,format=raw",
"-device", "usb-storage,bus=gpmu-xhci,0,drive=usbstick,port=2",
]

vga="none"

root@t2:/shared/xhci_demo# kfx --setup --domain ubuntu-20.04 --magic-mark 0x13371300 -c
Init vml, init_events: 1 init_paging 0 domain ubuntu-20.04 domid 0 json (null)
Waiting for harness start (cpuid with leaf 0x13371300)
Got start cpuid callback with leaf: 0x13371300 subleaf: 0x10
target buffer & size: 0xffff888938ad940 16
Parent is ready

root@t2:/shared/xhci_demo# xl list
Name  ID   Mem VCPUs State Time(s)
Domain-0  0  8192  8  r------ 2472.7
ubuntu-20.04  12  1024  1  --p---- 31.4
Disk /dev/loop5: 90.6 MiB, 94986240 bytes, 185520 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/loop6: 64.79 MiB, 67915776 bytes, 132648 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/loop7: 65.1 MiB, 68259840 bytes, 133320 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/xvda: 40 GB, 42949672960 bytes, 83886080 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x17cc52b9

Device Boot Start End Sectors Size Id Type
/dev/xvda1 1053623 1084576 512M b hfs+ FAT32
/dev/xvda2 1052678 83884031 82831363 39.5G 5 Extended
/dev/xvda5 1052672 83884031 82831360 39.5G 83 Linux

root@t2:/shared/xhci_demo# cat ubuntu-20.04.cfg
arch = 'x86_64'
name = "ubuntu-20.04"
maxmem = 1024
memory = 1024
vcpus = 1
numcpus = 1
nomigrate = 1
boot = "cd"
type = "hvm"
selaci = "pty"

vif = ["bridge=xenbr0,mac=00:07:88:8c:00:00"]
disk = ["/dev/t2vg/ubuntu-20.04,xvda,w"]

vmtrace_buf_kb=65536
device_model_args = ["-device", "qemu-xhci,id=qemu-xhci, p=8", 
"-device", "file=/tmp/usbstick,if=none, id=usbstick, format=raw", 
"-device", "usb-storage, bus=qemu-xhci,0, drive=usbstick, port=2", 
]

vga="none"
vnc=

root@t2:/shared/xhci_demo# kfx --setup --domain ubuntu-20.04 --magic-mark 0x13371300
Waiting for harness start (cpuid with leaf 0x13371300)
Got start cpuid callback with leaf: 0x13371300 subleaf: 0x10
target buffer & size: 0xffff888033ad9d0 16
Parent is ready
root@t2:/shared/xhci_demo# mkdir input
root@t2:/shared/xhci_demo# mkdir output
root@t2:/shared/xhci_demo# rm -d domid 12 --read 0xffff888033ad9d0 --file input/seed --limit 16
Init vmt, init_events: 0 initpaging 0 domain ubuntu-20.04 domid 0 json (null)
Read operation success: 16 bytes from 0xffff888033ad9d0
root@t2:/shared/xhci_demo#
Disk /dev/loop5: 90.6 MiB, 94986240 bytes, 185520 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/loop6: 64.79 MiB, 67915776 bytes, 132648 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/loop7: 65.1 MiB, 68259840 bytes, 133320 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/xvda: 40 GiB, 42949672960 bytes, 838860800 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x17cc52b9

Device  Boot   Start  End   Sectors  Size Id Type
/dev/xvda1  2048   1058623 1048576  512M b  hda FAT32
/dev/xvda2  1052670  83884031 82831362 39.5G 5  Extended
/dev/xvda5  1052672  83884031 82831360 39.5G  83 Linux

root@t2:/shared/xhci_demo# kfx --setup --domain ubuntu-20.04 --magic-mark 0x13371300 -c
root@t2:/shared/xhci_demo# xl list
Name       ID    Mem VCPUs State    Time(s)   vga
Domain-0   0      8192  8       r-----   2472.7
ubuntu-20.04 12   1824   1       --p-----  31.4
root@t2:/shared/xhci_demo# afl-fuzz -i input -o output --kfx --domain 12 --json kernel.json --address 0xffff8800338ad9d0 --input @input-limit 16 --ptcov
there is more to fuzzing

- analysis
- fuzzing
- triaging
Memory made accessible to a device

• Facilitates fast I/O operations
• IOMMU can restrict device to access only DMA pages

Finding where Linux reads from DMA isn’t trivial

• __iomem, be{16/32/64}_to_cpu, le{16/32/64}_to_cpu, dma_addr_t
• Cross-reference with ftrace output
let's just read the spec

event = xhci->event_ring->dequeue;
+
   harness_start(0x13371300, event, sizeof(*event));
+
   /* Does the HC or OS own the TRB? */
   if ((le32_to_cpu(event->event_cmd.flags) & TRB_CYCLE) !=
       xhci->event_ring->cycle_state)
   +
   {
      harness_stop();
      return 0;
   }

trace_xhci_handle_event(xhci->event_ring, &event->generic);

/* to make sure a watchdog timer didn't mark the host as non-responsive. */
if (xhci->xhc_state & XHCI_STATE_DYING) {
    harness_stop();
    xhci_dbg(xhci, "xHCI host dying, returning from "
            "event handler.\n");
    return 0;
}

/* Update SW event ring dequeue pointer */
inc_deq(xhci, xhci->event_ring);

harness_stop();

/* Are there more items on the event ring? Caller will call us again to*
  * check. */
diff --git a/drivers/usb/host/xhci.h b/drivers/usb/host/xhci.h
index ebb359ebb261..77be2863cfd9 100644
--- a/drivers/usb/host/xhci.h
+++ b/drivers/usb/host/xhci.h
@@ -2750,4 +2750,22 @@ static inline const char *xhci_decode_ep_context(u32 info, u32 info2, u64 deq,
     return str;
 }

+static inline void harness_start(unsigned int magic, void *a, size_t s)
+{
+    asm volatile ("cpuid"
+                  : "=a" (magic), "=c" (magic), "=S" (magic)
+                  : "a" (magic), "c" (s), "S" (a)
+                  : "bx", "dx");
+
+
+static inline void harness_stop(void)
+{
+    unsigned int tmp;
+    asm volatile ("cpuid"
+                  : "=a" (tmp)
+                  : "a" (0x13371337)
+                  : "bx", "cx", "dx");
+}
Can’t just “log in and gather logs”

• No network
• No disk
• No console

The dmesg buffer of Linux is in RAM

• We can carve it out!
It’s been shipping with Xen since 2009

Minimal gdb bridge

Build Linux with:

```bash
CONFIG_DEBUG_INFO=y
CONFIG_GDB_SCRIPTS=y
CONFIG_FRAME_POINTER=y
CONFIG_UNWINDER_FRAME_POINTER=y
CONFIG_RANDOMIZE_BASE=n
```
Testing aborted by user

[24]

```
++ Testing aborted by user ++

[24] We're done here. Have a nice day!
```

```
root@t2:/shared/xhci_demo# kfx --setup --domain ubuntu-20.04 --magic-mark 0x13373100 -c
Init vmi, init_events: 1 init_paging 0 domain ubuntu-20.04 domid 0 json (null)
Waiting for harness start (cpuid with leaf 0x13373100)
Got start cpuid callback with leaf: 0x13373100 subleaf: 0x10
Target buffer & size: 0x000080338ad9d0 16
Parent is ready
Fork VMS created: 12 -> 15 -> 16
Init vmi, init_events: 0 init_paging 0 domain (null) domid 12 json (null)
```

```
root@t2:/shared/xhci_demo# cat ubuntu-20.04.cfg
arch = x86_64
name = ubuntu-20.04
maxmni = 1024
memory = 1024
vcpu = 1
maxcpus = 1
nomigrate = 1
boot = "cd"
type = hwmm
serial = "pty"

vif = ["bridge=xenbr0,mac=00:07:58:BC:00:00"]
disk = ["/dev/2vg/ubuntu-20.04.xvdw.w"]

vmtrace_buf_kb=65536

device_model_args = [
  "-device", "qemu-xhci,id=qemu-xhci,p3=8",
  "-device", "filep=tpm2bustick,if=none,Id=usbstick,format=raw",
  "-device", "usb-storage,bus=qemu-xhci.0,drive=usbstick,port=2",
]
```

```
vgae="none"
```

```
vnc=

root@t2:/shared/xhci_demo# kfx --setup --domain ubuntu-20.04 --magic-mark 0x13373100 -c
Init vmi, init_events: 1 init_paging 0 domain ubuntu-20.04 domid 0 json (null)
Waiting for harness start (cpuid with leaf 0x13373100)
Got start cpuid callback with leaf: 0x13373100 subleaf: 0x10
Target buffer & size: 0x000080338ad9d0 16
Parent is ready
Fork VMS created: 12 -> 15 -> 16
Init vmi, init_events: 0 init_paging 0 domain (null) domid 12 json (null)
Read operation success: 16 bytes from 0xffffffff38ad9d0
```

```
root@t2:/shared/xhci_demo# ```
+++ Testing aborted by user +++

*[!] We're done here. Have a nice day!

root@t2:/shared/xhci_demo# kfx --domain 12 --json kernel.json --address 0xffff880338ad9d0 --output put/output/default/crashes/id\0000000, sig\06, src\00031+000016, time\06699, op\splice, ret

##### START #######
Init vml, int_events: 0 init_paging 0 domain (null) domid 12 json (null)
Parent is ready
Fork VMs created: 12 -> 15 -> 16
Init vml, int_events: 0 init_paging 1 domain (null) domid 15 json kernel.json
Creating sink list from built-in information listed in sink.h
Failed to find address for sink ksan_report in the JSON
Setting breakpoint on sink usbn_prologue 0xffff882224498 -> 0x22224498
Setting breakpoint on sink ksan_report 0xffff8813ff3f0 -> 0x13ff3f0
Failed to find address for sink page_fault in the JSON
Setting breakpoint on sink oops_begin 0xffff882408ac8 -> 0x2408ac8
Setting breakpoint on sink panic 0xffff8821ee4b4 -> 0x21ee4b4
Init vml, int_events: 1 init_paging 1 domain (null) domid 16 json (null)
Setup trace
Setup trace finished
VM Fork is ready for fuzzing
Starting fuzzer on 16
Running in standalone mode
Get 16 bytes of input from output/default/crashes/id\0000000, sig\06, src\00031+000016, time\06699,
qsplice, ret=4
Got input size 15
Starting trace from 0xffff8880d8739a.
Writing 15 bytes of input to 0xffff880338ad9d0 1
Starting fuzz loop
[TRACER int3] RTPS: 0xffff882224498
Sink USBN_prologue! Tracer counter: 0. Crash: 1.
Stopping fuzz loop.
Result: crash.
Closing tracer

##### DONE #######

root@t2:/shared/xhci_demo# cat ubuntu-20.04.cfg
arch = "x86_64"
name = "ubuntu-20.04"
maxmem = 1024
memory = 1024
cpus = 1
maxcpus = 1
nomigrate = 1
boot = "cd"
type = "hvm"
serial = "pty"
vif = ["bridge=exbr0,mac=00:07:5B:BC:00:00"]
disk = ["/phys/dev/t2vg/ubuntu-20.04,xvda,w"]
vmtrace_buf_kb=65536

device_model_args = 
"-device", "qemu-xhci, id=qemu-xhci, p=8",
"-drive", "file=/tmp/usbstick, if=none, id=usbstick, format=raw",
"-device", "usb-storage, bus=qemu-xhci.0,drive=usbstick, port=2",
]
vga="none"

/virt

root@t2:/shared/xhci_demo# kfx --setup --domain ubuntu-20.04 --magic-mark 0x13371300 -c
Init vml, int_events: 1 init_paging 0 domain ubuntu-20.04 0 domid 0 json (null)
Waiting for harness start (cpuid with leaf 0x13371300)
Got cpuid callback with leaf: 0x13371300 subleaf: 0x10
Target buffer & size: 0xffff880338ad9d0 16
Parent is ready

root@t2:/shared/xhci_demo# mkdir input
root@t2:/shared/xhci_demo# mkdir output
root@t2:/shared/xhci_demo# rmven --domid 12 --read 0xffff880338ad9d0 --file input/seed --limit t 16
Init vml, int_events: 0 init_paging 1 domain (null) domid 12 json (null)
Read operation success: 16 bytes from 0xffff880338ad9d0

root@t2:/shared/xhci_demo#
Closing tracer

########### DONE ###########

root@t2:/shared/xhcl_demo# kfx --domain 12 --json kernel.json --address 0xffff880338ad9d0 --trace output/default/crashes/id:0000080,slg:06,src:008031+008016,time:6669,op:splice,rep:4 --input-limit 16 --nocov --debug --sink u8s8_epilogue --keep

########### START ############

Init vmi, init_events: 0 init_paging 0 domain (null) domid 12 json (null)
Parent is ready
Fork VMs created: 12 -> 17 -> 18
Init vmi, init_events: 0 init_paging 1 domain (null) domid 17 json kernel.json
Setting breakpoint on sink u8s8_epilogue 0xffff882224582 -> 0x2224582
Sinks are ready
Init vmi, init_events: 1 init_paging 1 domain (null) domid 18 json (null)
Setup trace
Setup trace finished
VM Fork is ready for fuzzing
Starting fuzzer on 18
Starting fuzz loop

Get 16 bytes of input from output/default/crashes/id:0000080,slg:06,src:008031+008016,time:6669,op:splice,rep:4
Got input size 15
Starting trace from 0xffff8881d8739a.
Writing 15 bytes of input to 0xffff888338ad9d0
Starting fuzz loop
[TRACER int3] RIP: 0xffff882224582
$Sink u8s8_epilogue! Tracer counter: 0. Crash: 1.
Stopping fuzz loop.
Result: crash
Closing tracer

########### DONE ###########

root@t2:/shared/xhcl_demo# xl list

<table>
<thead>
<tr>
<th>Name</th>
<th>ID</th>
<th>Mem</th>
<th>VCPUs</th>
<th>State</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain-9</td>
<td>0</td>
<td>8192</td>
<td>8</td>
<td>r-----</td>
<td>2499.5</td>
</tr>
<tr>
<td>ubuntu-20.04</td>
<td>12</td>
<td>1023</td>
<td>1</td>
<td>p-----</td>
<td>31.4</td>
</tr>
<tr>
<td>ubuntu-11</td>
<td>17</td>
<td>0</td>
<td>1</td>
<td>p-----</td>
<td>0.0</td>
</tr>
<tr>
<td>(null)</td>
<td>18</td>
<td>0</td>
<td>1</td>
<td>p-----</td>
<td>0.0</td>
</tr>
</tbody>
</table>

root@t2:/shared/xhcl_demo# gdbx -a 18 64 4567

Listening on port 4567

arch = 'x86_64'
name = 'ubuntu-20.04'
maxmem = 1024
memory = 1024
vcpu = 1
maxvcpu = 1
nomigrate = 1
boot = "cd"
type = "hwv"
serial = "pty"
vif = ["bridge=xenbr0,mac=00:08:53:8B:8C:00:04",
"disk = ["phy:/dev/t2vg/ubuntu-20.04,xvda,w",

vmtrace_buf_kb=65536

device_model_args = [
  "device", "qemu-xhci,id=qemu-xhci,p=3", "drive", "file=tmp/usbdisk,if=none,id=usbdisk,format=raw", 
  "device", "usb-storage,bus=qemu-xhci.0,drive=usbdisk,dir=2",
]

vga = "none"

root@t2:/shared/xhcl_demo# kfx --setup --domain ubuntu-20.04 --magic-mark 0x13371300 -c

Init vmi, init_events: 1 init_paging 0 domain ubuntu-20.04 domid 0 json (null)
Waiting for harness start (cpuid with leaf 0x13371300)
Got start cpuid callback with leaf: 0x13371300 subleaf: 0x10
Target buffer & size: 0xffff880338ad9d0 16
Parent is ready

root@t2:/shared/xhcl_demo# mkdir input

root@t2:/shared/xhcl_demo# mkdir output

root@t2:/shared/xhcl_demo# mkdir --domain 12 --read 0xffff880338ad9d0 --file input/seed --link t 16

Init vmi, init_events: 0 init_paging 1 domain (null) domid 12 json (null)
Read operation success: 16 bytes from 0xffff880338ad9d0

root@t2:/shared/xhcl_demo# cd /linux/

root@t2:/shared/linux# gdb vm Linux -ex 'target remote:4567' -ex 'lx-dmesg' -ex 'd'
5.172237: lp: driver loaded but no devices found
5.191441: pdev: user-space parallel port driver
5.196170: systemd[1]: Starting udev Coldplug All Devices...
5.227720: EXT4-fs (xvd5): re-mounted.Opts: errors=remount-ro
5.235619: systemd[1]: Starting Uncomplicated firewall...
5.296859: systemd[1]: Started Journal Service.
5.459965: Adding 1918356k swap on /swapfile. Priority: -2 extents: 5 across:1951124k SFS
5.547991: systemd-journald[282]: Received client request to flush runtime journal.
7.955898: crypto: max_cpu_glen set to 1060
7.611235: AVX2 version of gcc_enc/doc engaged.
7.611238: AES CTR mode by8 optimization enabled
8.868795: Input: Xen Virtual Keyboard as /devices/virtual/input/inputs
8.874412: Input: Xen Virtual Pointer as /devices/virtual/input/inputs
19.559953: audit: type=1400 audit=1623688469.128:2: apparmor:STATUS operation="profile_load" profile="unconfined" path="/dev/ndoe/probe" pid=364 comm="apparmor_parser"
19.559969: audit: type=1400 audit=1623688469.128:3: apparmor:STATUS operation="profile_load" profile="unconfined" path="/dev/ndoe/probe/kmod" pid=364 comm="apparmor_parser"
19.602633: audit: type=1400 audit=1623688469.168:5: apparmor:STATUS operation="profile_load" profile="unconfined" name="/usr/sbin/cupsd" pid=365 comm="apparmor_parser"
19.602646: audit: type=1400 audit=1623688469.168:6: apparmor:STATUS operation="profile_load" profile="unconfined" name="/usr/sbin/cupsd" third_party" pid=365 comm="apparmor_parser"
19.626879: audit: type=1400 audit=1623688469.192:7: apparmor:STATUS operation="profile_load" profile="unconfined" name="/sbin/lp" pid=368 comm="apparmor_parser"
19.636608: audit: type=1400 audit=1623688469.204:8: apparmor:STATUS operation="profile_load" profile="unconfined" name="/sbin/lp" pid=368 comm="apparmor_parser"
19.656092: audit: type=1400 audit=1623688469.224:9: apparmor:STATUS operation="profile_load" profile="unconfined" name="/sbin/cupsd" pid=370 comm="apparmor_parser"
19.669014: audit: type=1400 audit=1623688469.236:10: apparmor:STATUS operation="profile_load" profile="unconfined" name="/bin/unprint" pid=370 comm="apparmor_parser"
19.669931: audit: type=1400 audit=1623688469.236:11: apparmor:STATUS operation="profile_load" profile="unconfined" name="man_filter" pid=371 comm="apparmor_parser"
19.691824: audit: type=1400 audit=1623688469.236:12: apparmor:STATUS operation="profile_load" profile="unconfined" name="man_filter" pid=371 comm="apparmor_parser"
129.841886: index is out of range for type 'xhci_virt_ep [31]'
triaging beyond the basics

Gathering the dmesg log has been sufficient to triage most bugs

Not all of them

Sink may get triggered by code far away from the driver
static int igb_clean_rx_irq(struct igb_q_vector *q_vector, const int budget) {
    struct igb_ring *rx_ring = q_vector->rx_ring;
    struct sk_buff *skb = rx_ring->skb;
    unsigned int total_bytes = 0, total_packets = 0;
    u16 cleaned_count = igb_desc_unused(rx_ring);

    while (likely(total_packets < budget)) {
        union e1000_adv_rx_desc *rx_desc;
        struct igb_rx_buffer *rx_buffer;
        unsigned int size;

        harness_stop();

        /* return some buffers to hardware, one at a time is too slow */
        if (cleaned_count >= IGB_RX_BUFFER_WRITE) {
            igb_alloc_rx_buffers(rx_ring, cleaned_count);
            cleaned_count = 0;
        }

        rx_desc = IGB_RX_DESC(rx_ring, rx_ring->next_to_clean);

        harness_start(0x13371500, rx_desc, sizeof(*rx_desc));
    }
}
BUG: KASAN: null-ptr-deref in gro_pull_from_frag0+0x98/0x1f0
Read of size 28 at addr 0000000000000000 by task rcu_sched/11

CPU: 0 PID: 11 Comm: rcu_sched Not tainted 5.4.91kfx-next-2+ #7
Hardware name: Xen HVM domU, BIOS 4.14.1 12/21/2020
Call Trace:
<IRQ>
dump_stack+0x96/0xc7
__kasan_report.cold+0x65/0x92
? gro_pull_from_frag0+0x98/0x1f0
kasan_report+0x14/0x20
check_memory_region+0x129/0x1b0
memcpy+0x24/0x50
gro_pull_from_frag0+0x98/0x1f0
dev_gro_receive+0xd2/0xd10
? __asan_loadN+0xf/0x20
? skb_gro_reset_offset+0x8f/0x140
napi_gro_receive+0x56/0x1e0
igb_poll+0xc48/0x1eb0
net_rx_action+0x29c/0x700
? napi_busy_loop+0x3d0/0x3d0
...
Bug triggers in net/core/dev.c, NOT in IGB itself:

```
static void gro_pull_from_frag0(struct sk_buff *skb, int grow) {
    struct skb_shared_info *pinfo = skb_shinfo(skb);
    BUG_ON(skb->end - skb->tail < grow);
    memcpy(skb_tail_pointer(skb), NAPI_GRO_CB(skb)->frag0, grow);
}
```
root@ccg-crfw-bld11:/home/cros/demo# xl list

Name          ID    Mem  VCPUs  State    Time(s)
Domain-0      0      24000 12    r------    5688.7
ubuntu-2004   37     4830   1    -p------    24.7
(null)         38     0     1    -p------    0.0
(null)         39     0     1    ------     410.8

root@ccg-crfw-bld11:/home/cros/demo# kfx --domid 37^C
root@ccg-crfw-bld11:/home/cros/demo# cat setup.log
Init: vml, init_events: 1 init_pageing 0 domain (null) domid 37 json (null)
Waiting for harness start (cpuid with leaf 0x13371500)
Got start cpuid callback with leaf: 0x13371500 subleaf: 0x10
Target buffer & size: 0xffff8880b7c13000 16
Parent is ready

root@ccg-crfw-bld11:/home/cros/demo# kfx --domid 37 --json kfx-2.json --address 0xffff8880b7c13000 --input output/default/crashes/id\:000000\,sig\:06\,src\:000000\,time\:209
\,op\:havoc\,rep\:8 --input-limit 16 --nocov --debug > debug.log
Init vml, init_events: 0 init_paging 0 domain (null) domid 37 json (null)
Fork VMs created: 37 -> 58 -> 59
Init vml, init_events: 0 init_paging 1 domain (null) domid 58 json kfx:2.json
Creating sink list from built-in information listed in slink.h
Setting breakpoint on slink ubsan_prologue 0xffffffff82502babc -> 0x2502cbc
Setting breakpoint on slink page_fault 0xffffffff82601250 -> 0x2601250
Setting breakpoint on slink oops_begin 0xffffffff816606560 -> 0x16606560
Setting breakpoint on slink panic 0xffffffff824d2049 -> 0x24d2049
Sinks are ready
Init vml, init_events: 1 init_paging 1 domain (null) domid 59 json (null)
Setup trace
Setup trace finished
VM Fork is ready for fuzzing
Starting fuzzer on 59
Running in standalone mode
Get 16 bytes of input from output/default/crashes/ld:000000, sig:86, src:000000, time:2090, op:havoc, rep:8
Got input size 16
Starting trace from 0xffffffff81e07f58.
Writing 16 bytes of input to 0xffffffff80800b7c3000
Starting fuzz loop
[TRACER cpuid] RIP: 0xffffffff81e08669
CPUID leaf 13371501
[TRACER cpuid] RIP: 0xffffffff81e08672
CPUID leaf ffff8888
[TRACER cpuid] RIP: 0xffffffff81e082c8
CPUID leaf 13371502
[TRACER cpuid] RIP: 0xffffffff81e082d0
CPUID leaf ffff8888
[TRACER cpuid] RIP: 0xffffffff81e08306
CPUID leaf 13371503
[TRACER cpuid] RIP: 0xffffffff81e0830f
CPUID leaf ffff8888
[TRACER int3] RIP: 0xffffffff8149e050
Stopping fuzz loop.
Result: crash
Closing tracer
########### DONE ###############
root@ccg-crfw-bld11:/home/cros/demo# forkvm 37
Fork VM id: 60
root@ccg-crfw-bld11:/home/cros/demo# rwmen --domid ^C
root@ccg-crfw-bld11:/home/cros/demo# cat setup.log
Init vmi, init_events: 1 init_paging 0 domain (null) domid 37 json (null)
Waiting for harness start (cpuid with leaf 0x13371500)
Got start cpuid callback with leaf: 0x13371500 subleaf: 0x10
Target buffer & size: 0xffff8880b7c13000 16
Parent is ready
root@ccg-crfw-bld11:/home/cros/demo# rwmen --domid 60 --write 0xffff8880b7c13000 --input output/default/crashes/id\:000000\,sig\:06\,src\:000000\,time\:2090\,op\:havoc\,rep\:8 --limit 16
rwmen: unrecognized option '--input'
Usage:
  --domid <domid>
  --read <address>
  --write <address>
  --file <input/output file>
  --limit <input/output limit>
Optional:
  --pageable <address> (-1 for physical memory access)
root@ccg-crfw-bld11:/home/cros/demo# rwmen --domid 60 --write 0xffff8880b7c13000 --file output/default/crashes/id\:000000\,sig\:06\,src\:000000\,time\:2090\,op\:havoc\,rep\:8 --limit 16
Init vmi, init_events: 0 init_paging 1 domain (null) domid 60 json (null)
Write operation success: 16 bytes to 0xffff8880b7c13000
root@ccg-crfw-bld11:/home/cros/demo#
Init vmi, init_events: 0 init_paging 0 domain (null) domid 37 json (null)
Parent is ready
Fork VMs created: 37 -> 58 -> 59
Init vmi, init_events: 0 init_paging 1 domain (null) domid 58 json kfx-2.json
Creating sink list from built-in information listed in sink.h
Setting breakpoint on sink ubsan_prologue 0xffffffff82502bcb -> 0x2502bcb
Setting breakpoint on sink kasan_report 0xffffffff8149e050 -> 0x149e050
Setting breakpoint on sink page_Fault 0xffffffff82691250 -> 0x2691250
Setting breakpoint on sink c cops__begin 0xffffffff81605050 -> 0x1605050
Setting breakpoint on sink panic 0xffffffff824d2049 -> 0x24d2049
Sinks are ready
Init vmi, init_events: 1 init_paging 1 domain (null) domid 59 json (null)
Setup trace
Setup trace finished
VM Fork is ready for fuzzing
Starting fuzzer on 59
Running in standalone mode
Get 16 bytes of input from output/default/crashes/id:000000,slg:06,src:000000,time:2090,op:havoc,rep:8
Get input size 16
Starting trace from 0xffffffff801e07f58.
Writing 16 bytes of input to 0xffffffff8080b7c13000
Starting fuzz loop:
[TRACER cpu6] RIP: 0xffffffff81e08069
CPUID leaf 13371501
[TRACER cpu6] RIP: 0xffffffff81e08072
CPUID leaf ffff8880
[TRACER cpu6] RIP: 0xffffffff81e082c0
CPUID leaf 13371502
[TRACER cpu6] RIP: 0xffffffff81e082d0
CPUID leaf ffff8880
[TRACER cpu6] RIP: 0xffffffff81e08306
CPUID leaf 13371503
[TRACER cpu6] RIP: 0xffffffff81e0830f
CPUID leaf ffff8880
[TRACER int3] RIP: 0xffffffff8149e050
Stopping fuzz loop.
Result: crash
Closing tracer

DONE
cat stepper.log | awk '{ print $2 }' | addr2line -e kfx-2.vmlinux -f -p -a > stepper-decoded.log
0xffffffff8149f277: memory_is_poisoned_2_4_8 at /home/cros/chrome-kernel/mm/kasan/generic.c:70
0xffffffff8149f27b: memory_is_poisoned_2_4_8 at /home/cros/chrome-kernel/mm/kasan/generic.c:70
0xffffffff8149f27e: memory_is_poisoned_2_4_8 at /home/cros/chrome-kernel/mm/kasan/generic.c:70
0xffffffff8149f281: memory_is_poisoned_2_4_8 at /home/cros/chrome-kernel/mm/kasan/generic.c:70
0xffffffff8149f285: memory_is_poisoned_2_4_8 at /home/cros/chrome-kernel/mm/kasan/generic.c:70
0xffffffff8149f287: memory_is_poisoned_1 at /home/cros/chrome-kernel/mm/kasan/generic.c:51
0xffffffff8149f291: kasan_mm_to_shadow at /home/cros/chrome-kernel/mm/include/linux/kasan.h:28
0xffffffff8149f295: memory_is_poisoned_1 at /home/cros/chrome-kernel/mm/kasan/generic.c:51
0xffffffff8149f299: memory_is_poisoned_1 at /home/cros/chrome-kernel/mm/kasan/generic.c:53
0xffffffff8149f29b: memory_is_poisoned_1 at /home/cros/chrome-kernel/mm/kasan/generic.c:53
0xffffffff8149f2ab: __asan_load8 at /home/cros/chrome-kernel/mm/kasan/generic.c:251
0xffffffff8149f2b9: __asan_load8 at /home/cros/chrome-kernel/mm/kasan/generic.c:251
0xffffffff822122d8: gro_pull_from_frag8 at /home/cros/chrome-kernel/net/core/dev.c:5422 (discriminator 2)
0xffffffff822122e5: memcpy at /home/cros/chrome-kernel/mm/include/linux/string.h:403 (discriminator 2)
0xffffffff822122e9: skb_tail_pointer at /home/cros/chrome-kernel/mm/include/linux/skbuff.h:2167 (discriminator 2)
0xffffffff822122ec: memcpy at /home/cros/chrome-kernel/mm/include/linux/string.h:403 (discriminator 2)
0xffffffff822122f0: skb_tail_pointer at /home/cros/chrome-kernel/mm/include/linux/skbuff.h:2167 (discriminator 2)
0xffffffff822122f3: memcpy at /home/cros/chrome-kernel/mm/include/linux/string.h:403 (discriminator 2)
0xffffffff8149f740: memcpy at /home/cros/chrome-kernel/mm/kasan/common.c:121
0xffffffff8149f741: memcpy at /home/cros/chrome-kernel/mm/kasan/common.c:121
0xffffffff8149f744: memcpy at /home/cros/chrome-kernel/mm/kasan/common.c:121
0xffffffff8149f746: memcpy at /home/cros/chrome-kernel/mm/kasan/common.c:121
0xffffffff8149f749: memcpy at /home/cros/chrome-kernel/mm/kasan/common.c:121
0xffffffff8149f74a: memcpy at /home/cros/chrome-kernel/mm/kasan/common.c:121
0xffffffff8149f74c: memcpy at /home/cros/chrome-kernel/mm/kasan/common.c:121
0xffffffff8149f74d: memcpy at /home/cros/chrome-kernel/mm/kasan/common.c:121
0xffffffff8149f74e: memcpy at /home/cros/chrome-kernel/mm/kasan/common.c:121
0xffffffff8149f750: memcpy at /home/cros/chrome-kernel/mm/kasan/common.c:121
0xffffffff8149f751: memcpy at /home/cros/chrome-kernel/mm/kasan/common.c:121
0xffffffff8149f755: memcpy at /home/cros/chrome-kernel/mm/kasan/common.c:121
0xffffffff8149f757: memcpy at /home/cros/chrome-kernel/mm/kasan/common.c:121
0xffffffff8149f75a: memcpy at /home/cros/chrome-kernel/mm/kasan/common.c:121
0xffffffff8149f75c: memcpy at /home/cros/chrome-kernel/mm/kasan/common.c:121
0xffffffff8149f530: check_memory_region_inline at /home/cros/chrome-kernel/mm/kasan/generic.c:172
0xffffffff8149f533: check_memory_region_inline at /home/cros/chrome-kernel/mm/kasan/generic.c:172
0xffffffff8149f539: check_memory_region_inline at /home/cros/chrome-kernel/mm/kasan/generic.c:175
0xffffffff8149f543: check_memory_region at /home/cros/chrome-kernel/mm/kasan/generic.c:190
0xffffffff8149f544: check_memory_region at /home/cros/chrome-kernel/mm/kasan/generic.c:190
0xffffffff8149f547: check_memory_region at /home/cros/chrome-kernel/mm/kasan/generic.c:190
0xffffffff8149f54a: check_memory_region at /home/cros/chrome-kernel/mm/kasan/generic.c:190
0xffffffff8149f54c: check_memory_region at /home/cros/chrome-kernel/mm/kasan/generic.c:190
### Output from `addr2line -e kfx-2.vm` to `nano stepper-decoded.log`

```bash
root@ccg-crfw-bld11:/home/cros/demo# cat stepper.log | awk '{ print $2 }' | addr2line -e kfx-2.vm linux -f -p -a > stepper-decoded.log
```

### `xl list`

<table>
<thead>
<tr>
<th>Name</th>
<th>ID</th>
<th>Mem</th>
<th>VCPUs</th>
<th>State</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain-0</td>
<td>0 24800</td>
<td>12</td>
<td>r----</td>
<td>5813.5</td>
<td></td>
</tr>
<tr>
<td>ubuntu-2004</td>
<td>37 4030</td>
<td>1</td>
<td>-p---</td>
<td>24.7</td>
<td></td>
</tr>
<tr>
<td>(null)</td>
<td>38 0 1</td>
<td>-p---</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(null)</td>
<td>39 0 1</td>
<td>r----</td>
<td>464.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(null)</td>
<td>60 0 1</td>
<td>-p---</td>
<td>0.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### `forkvm 37`

```bash
Fork VM id: 61
```

### `rwmem` on `domain 61`

```bash
root@ccg-crfw-bld11:/home/cros/demo# rwmem --domid 61 --write 0xffff8888b7c13000 --file output/default/crashes/id\:000000\,sig\:06\,src\:000000\,time\:2900\,op\:havoc\,rep\:
8 --limit 16
```

Write operation success: 16 bytes to 0xffff8888b7c13000

### `xl list`

<table>
<thead>
<tr>
<th>Name</th>
<th>ID</th>
<th>Mem</th>
<th>VCPUs</th>
<th>State</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain-0</td>
<td>0 24800</td>
<td>12</td>
<td>r----</td>
<td>5833.4</td>
<td></td>
</tr>
<tr>
<td>ubuntu-2004</td>
<td>37 4030</td>
<td>1</td>
<td>-p---</td>
<td>24.7</td>
<td></td>
</tr>
<tr>
<td>(null)</td>
<td>38 0 1</td>
<td>-p---</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(null)</td>
<td>39 0 1</td>
<td>-p---</td>
<td>472.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(null)</td>
<td>60 0 1</td>
<td>-p---</td>
<td>0.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

### Xen-hvctx 61

```bash
root@ccg-crfw-bld11:/home/cros/demo# xen-hvctx 61 | grep rsi
```

```
Header: magic 0x54381286, version 1
    rbp 0xffff8888c9409bf0    rsp 0x000000000000001c
```

```bash
root@ccg-crfw-bld11:/home/cros/demo# xen-hvctx 61 | grep rdi
```

```
rdi 0x0000000000000000
```

```bash
root@ccg-crfw-bld11:/home/cros/demo# xen-hvctx 61 | grep rsp
```

```
rsi 0xffff8888c9409bd8
```

```bash
root@ccg-crfw-bld11:/home/cros/demo#```
Both src and dst in the memcpy is corrupted

Anything else we can do?
ubuntu-2004 37 4030 1 -p--- 24.7
(null) 38 0 1 -p--- 0.0
(null) 39 0 1 r----- 686.3
(null) 60 0 1 -p-- 0.1
(null) 61 0 1 -p-- 0.1
(null) 62 0 1 -p-- 0.0
(null) 63 0 1 -p-- 0.0
(null) 66 0 1 -p-- 0.1

root@cg-crfw-bld11:/home/cros/demo# cat setup.log
Init vmi, init_events: 1 init_paging 0 domain (null) domid 37 json (null)
Waiting for harness start (cpuid with leaf 0x13371500)
Got start cpuid callback with leaf: 0x13371500 subleaf: 0x10
Target buffer & size: 0xffff8880b7c13000 16
Parent is ready

root@cg-crfw-bld11:/home/cros/demo# kfx --domid 37 --json kfx-2.json --address 0xffff8880b7c13000 --input input/seed --input-limit 16 --nocov --debug

# START ###################################################################
Init vmi, init_events: 0 init_paging 0 domain (null) domid 37 json (null)
Parent is ready
Fork VMs created: 37 -> 67 -> 68
Init vmi, init_events: 0 init_paging 1 domain (null) domid 67 json kfx-2.json
Creating sink list from built-in information listed in sink.h
Setting breakpoint on sink ubsan_prologue 0xfffff825922cb -> 0x25822bc
Setting breakpoint on sink kasan_report 0xfffff81a9e0850 -> 0x149e050
Setting breakpoint on sink pageFault 0xfffff82601250 -> 0x2601250
Setting breakpoint on sink oops_begin 0xfffff81060560 -> 0x1060560
Setting breakpoint on sink panic 0xfffff824d2049 -> 0x24d2049
Sinks are ready
Init vmi, init_events: 1 init_paging 1 domain (null) domid 68 json (null)
Setup trace
Setup trace finished
VM Fork is ready for fuzzing
Starting fuzzer on 68
Running in standalone mode
Get 16 bytes of input from input/seed
Got input size 16
Starting trace from 0xfffff81e07f58.
Writing 16 bytes of Input to 0xffff8880b7c13000
Starting fuzz loop
[TRACER cpuid] RIP: 0xfffff81e08069
CPUID leaf 13371501
[TRACER cpuid] RIP: 0xfffff81e08072
CPUID leaf 1fff8880
Init vm. init_events: 0 init_paging 1 domain (null) domid 67 json kfx-2.json
Creating sink list from built-in information listed in sink.h
Setting breakpoint on sink ubsan_prologue 0xfffffffff82502bcb -> 0x2502bcb
Setting breakpoint on sink kasan_report 0xfffffffff8149e050 -> 0x149e050
Setting breakpoint on sink page_fault 0xfffffffff82601250 -> 0x2601250
Setting breakpoint on sink oops_begin 0xfffffffff81660560 -> 0x1660560
Setting breakpoint on sink panic 0xfffffffff824d2649 -> 0x24d2649
Sinks are ready
Init vm. init_events: 1 init_paging 1 domain (null) domid 68 json (null)
Setup trace
Setup trace finished
VM Fork is ready for fuzzing
Starting fuzzer on 68
Running in standalone mode
Got 16 bytes of input from input/seed
Got input size 16
Starting trace from 0xfffffffff81e07f58.
Writing 16 bytes of input to 0xfffffffff8880b7c13000
Starting fuzz loop
[TRACER cpuid] RIP: 0xfffffffff81e0869
CPUID leaf 13371501
[TRACER cpuid] RIP: 0xfffffffff81e0872
CPUID leaf ffff8880
[TRACER cpuid] RIP: 0xfffffffff81e082c8
CPUID leaf 13371502
[TRACER cpuid] RIP: 0xfffffffff81e082d0
CPUID leaf ffff8880
[TRACER cpuid] RIP: 0xfffffffff81e08306
CPUID leaf 13371503
[TRACER cpuid] RIP: 0xfffffffff81e0830f
CPUID leaf ffff8880
[TRACER cpuid] RIP: 0xfffffffff81e07f08
CPUID leaf 13371337
Harness signal on finish
Stopping fuzz loop.
Result: no crash
Closing tracer

########### DONE ###############
root@ccg-crfw-bld11:/home/cros/demo# forkmv 37
Fork VM id: 69
root@ccg-crfw-bld11:/home/cros/demo# stepper --domid 69 --limit 1000000 --stop-on-address 0xfffffffff81e07f03 > normal.log
root@ccg-crfw-bld11:/home/cros/demo# cat normal.log | awk '{ print $2 }' | addr2line -e kfx-2.vmunix -f -p -a > normal-decoded.log
root@ccg-crfw-bld11:/home/cros/demo# diff stepper-decoded.log normal-decoded.log | less
Buggy code:

```c
/* pull timestamp out of packet data */
if (igb_test_staterr(rx_desc, E1000_RXDADV_STAT_TSIP)) {
    igb_ptp_rx_pktstamp(rx_ring->q_vector, skb->data, skb);
    __skb_pull(skb, IGB_TS_HDR_LEN);
}
```

Fixed code:

```c
/* pull timestamp out of packet data */
if (igb_test_staterr(rx_desc, E1000_RXDADV_STAT_TSIP)) {
    if (!igb_ptp_rx_pktstamp(rx_ring->q_vector, skb->data, skb))
        __skb_pull(skb, IGB_TS_HDR_LEN);
}
```
can you spot the bug?

case TRB_RESET_DEV:
    /* SLOT_ID field in reset device cmd completion event TRB is 0.
     * Use the SLOT_ID from the command TRB instead (xhci 4.6.11)
     */
    slot_id = TRB_TO_SLOT_ID(
        le32_to_cpu(cmd_trb->generic.field[3]));
    xhci_handle_cmd_reset_dev(xhci, slot_id, event);
    break;

static void xhci_handle_cmd_reset_dev(struct xhci_hcd *xhci, int slot_id,
                                        struct xhci_event_cmd *event)
{
    struct xhci_virt_device *vdev;
    struct xhci_slot_ctx *slot_ctx;

    vdev = xhci->devs[slot_id];
    slot_ctx = xhci_get_slot_ctx(xhci, vdev->out_ctx);
    trace_xhci_handle_cmd_reset_dev(slot_ctx);

    xhci_dbg(xhci, "Completed reset device command.\n");
    if (!xhci->devs[slot_id])
        xhci_warn(xhci, "Reset device command completion 
                    "for disabled slot %u\n", slot_id);
}
what can go wrong here?

```c
static void handle_cmd_completion(struct xhci_hcd *xhci, 
    struct xhci_event_cmd *event) 
{
    int slot_id = TRB_TO_SLOT_ID(le32_to_cpu(event->flags));

    cmd_type = TRB_FIELD_TO_TYPE(le32_to_cpu(cmd_trb->generic.field[3]));
    switch (cmd_type) {

    case TRB_ADDR_DEV:
        xhci_handle_cmd_addr_dev(xhci, slot_id);
        break;

    static void xhci_handle_cmd_addr_dev(struct xhci_hcd *xhci, int slot_id) 
    {
        struct xhci_virt_device *vdev;
        struct xhci_slot_ctx *slot_ctx;

        vdev = xhci->devs[slot_id];
        slot_ctx = xhci_get_slot_ctx(xhci, vdev->out_ctx);
        trace_xhci_handle_cmd_addr_dev(slot_ctx);
    }
```
what we found & fixed

• 9 NULL-pointer derefs
• 3 array index out-of-bounds
• 2 infinite loops in IRQ handler
• 2 user-memory accesses
doublefetch detection

Remove R/W EPT permission of DMA page

- Byte-granularity
- Look for PF with read access at the same page & offset 2x
- Report crash to AFL

We thought it would be rare

- Happens all over the place
- Happens because of different bits used of the same byte
- Doesn’t make sense to stop execution at the first detection
mission accomplished!!

Or is it?

- Did we discover all DMA input points?
- We got bogged down by documenting all the bugs
- Staring at the code is only productive up to a point

Let’s do better!
Standalone EPT fault monitoring

Linux allocates DMA pages through its internal DMA API

- `dma_alloc_attrs()`
- Hook function entry & ret with int3

Remove EPT permission on-the-fly

- Log all code-sites that read from any DMA!
Ubuntu
Advanced options for Ubuntu
Memory test (memtest86+)
Memory test (memtest86+, serial console 115200)
Zephyr Kernel

Use the ^ and v keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands
before booting or 'c' for a command-line.
The highlighted entry will be executed automatically in 1s.
Ubuntu 20.04.3 LTS ubuntu tty50

root@1262:/shared/hcl demon# cat screenlog.0 | grep 'r-' | awk '{ print $4 }' | sort -u | addr2line -e ./linux/vmlinux -f -p -c

ubuntu login:
Started System Logging Service.

Started LSB: Record successful boot for GRUB.

Started GRUB: automatic crash report generation.

Started WPA supplicant.

Started Dispatcher daemon for systemd-networkd.

Finished GRUB: failed boot detection.

Available DNS/DNS-SD Stack.

Started Login Service.

Started Network Manager.

Started Authorization Manager.

Started Switcheroo Control Proxy service.

Reached target Network.

Started Modem Manager.

Starting Network Manager Wait Online...

Started Make remote CUPS printers available locally.

Starting OpenSSH service...

Starting OpenBSD Secure Shell server...

Starting Hostname Service...

Starting Permit User Sessions...

Started Unattended Upgrades Shutdown.

Finished Permit User Sessions.

Started OpenBSD Secure Shell server.

Accounts Service.

Started Modem Manager.

Started Disk Manager.

Starting GNOME Display Manager...

Starting Hold until boot process finishes up...

Finished Hold until boot process finishes up.

Finished Hostname Service.

Starting Network Manager Script Dispatcher Service...

Started Serial Getty on hvc0.

Started Serial Getty on ttyS0.

Reached target Login Prompts.

Started Set console scheme.

Finished Network Manager Wait Online.

Finished Set console scheme.

Started Network Manager Script Dispatcher Service.

Created /etc/system-getty alice.

Reached target Network is Online.

Started Tool to automatic nit kernel crash signatures...

Started Tool to automatic nit kernel crash signatures...

Reached target Multi-User System.
still some problems

Sometimes the data isn’t used where DMA is accessed

• Where is it used and is it safe?

¯\_(ツ)_/¯

We don’t want to go back to reading the source

• Very hard to follow data-flow by eye
• Error-prone, manual, annoying
full-vm taint analysis

Track tainted data propagation in the kernel

- Record kernel execution using VMtrace
- Replay recorded instruction stream in Triton DBI’s taint engine
- Check where RIP gets tainted

Can find code-locations affected by DMA input anywhere
Fork VM id: 92

Name  ID  Men VCPUs  State  Time(s)
Domain-0  0   8192   8  r-----  96534.3
ubuntu-20.04  86   0   1  --p---  11.9
(null)  92   0   1  --p---  0.0

Usage:
Step 1:
--domid <domid>
--save-state <file>

Step 2:
--domid <domid>
--load-state <file>
--taint <address[size]> (can be specified multiple times)
--pt <file>
--json <file>
--skip-userspace (optional)

Saving state

Tainted reg: rdx: 0
Tainted reg: rdi: 0
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rip: ffffffff02de01e
Tainted reg: rdi: 0
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rip: ffffffff02de05e
Tainted reg: rdx: 0
Tainted reg: rdi: 1
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rip: ffffffff02de041
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rip: ffffffff02de083
Tainted reg: rdx: 0
Tainted reg: rdi: 0
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rip: ffffffff02de023
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rip: ffffffff02de05a
Tainted reg: rdx: 0
Tainted reg: rdi: 1
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rip: ffffffff02de041
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rip: ffffffff02de083
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rax: 0
Tainted reg: rdi: 0
Tainted reg: rip: ffffffff02de023

---

<table>
<thead>
<tr>
<th>Name</th>
<th>ID</th>
<th>Mem VCPUs</th>
<th>State</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain-0</td>
<td>0</td>
<td>0</td>
<td>8192</td>
<td>0</td>
</tr>
<tr>
<td>ubuntu-20.04</td>
<td>86</td>
<td>0</td>
<td>1</td>
<td>11.9</td>
</tr>
<tr>
<td>(null)</td>
<td>91</td>
<td>0</td>
<td>1</td>
<td>0.0</td>
</tr>
</tbody>
</table>

---

```
root@t2:~/shared/vntaint # xl list
```

```
Name  ID  Mem VCPUs  State  Time(s)  
Domain-0  0  8192  0       96524.2
ubuntu-20.04  86  0  1  11.9
(null)  91  0  1  0.0
```

---

```
root@t2:~/shared/vntaint # xl unpause 91
```

```
libxl: error: libxl_xshelp.c:201:libxl__xs_read_mandatory: xenstore read failed: '/libxl/91/type': No such file or directory
libxl: warning: libxl_dom.c:ss:libxl_domain_type: unable to get domain type for domid=91, assuming HVM
```

---

```
root@t2:~/shared/vntaint # .vntaint --help
```

usage:
Step 1:
```
  ...-domain <domid>
  ...-save-state <file>
```

Step 2:
```
  ...-domain <domid>
  ...-load-state <file>
  ...-taint <address[size]> (can be specified multiple times)
  ...-pt <file>
  ...-json <file>
  ...-skip-userspace (optional)
```

---

```
root@t2:~/shared/vntaint # .vntaint --save-state state.log --domain 86
```

Saving state

```
root@t2:~/shared/vntaint # xl unpause 92
```

```
libxl: error: libxl_xshelp.c:201:libxl__xs_read_mandatory: xenstore read failed: '/libxl/92/type': No such file or directory
libxl: warning: libxl_dom.c:ss:libxl_domain_type: unable to get domain type for domid=92, assuming HVM
```

---

```
root@t2:~/shared/vntaint # xl list
```

```
Name  ID  Mem VCPUs  State  Time(s)  
Domain-0  0  8192  0       96538.0
ubuntu-20.04  86  0  1  11.9
(null)  92  30  1  1
```

---

```
root@t2:~/shared/vntaint # .vntaint --domain 86 --load-state state.log --taint 0xffffffff02e00100:0x1000 --pt /vntrace/log --json /shared/errno.5.4.0-48.json > taint.log
```

---
Code released as open-source (MIT)

https://github.com/intel/vmtaint
Getting some targets working in a Xen VM is difficult

No problem!

- VM forks need only CPU-state and memory
- Take snapshot on QEMU/KVM/Simics
- Load it up on Xen
- Fuzz away!
what’s next

Automation

Capture system state using Intel® DCI

• Fuzz BIOS/SMM

• Fuzz bare-metal systems

Sandsifter ring0

Nested virt-support
things we didn’t cover here

Fuzzing other OS’s
Fuzzing Xen
Fuzzing userspace
Fuzzing black-box binaries
Fuzzing malware
Questions? Comments?
@tklengyel

Thanks goes to a whole lot of people who contributed & made this work possible!

Intel IAGS & CCG, Google ChromeOS team, Citrix, CERT.pl & the wider open-source security community releasing tools for everyone's benefit

https://github.com/intel/kernel-fuzzer-for-xen-project
https://github.com/intel/vmtaint