KVM Security Improvements

Andrew Honig
KVM Forum 2014

Motivation

- Tech Lead on Cloud Security for Google
    - Google Compute Engine - lots of untrusted users running whatever they want inside VMs on Google infrastructure.
    - VMs are all on KVM (https://cloud.google.com/compute/docs/faq)
- 9 CVEs in KVM (2 VM escapes)
- 6 CVEs in VMware (3 VM escapes)

KVM Vulnerability Types (non exhaustive list)

1. Guest Execution Escape
2. Guest reads of other guest data
3. Guest DoS of Host
4. Ring3-Ring0 privilege escalation (host-host or guest-guest)
5. Ring 3 DoS (host or guest)

KVM Vulnerability Types (non exhaustive list)

1. Guest Execution Escape
2. Guest reads of other guest data
3. Guest DoS of Host
4. Ring3-Ring0 privilege escalation (host-host or guest-guest)
5. Ring 3 DoS (host or guest)

Security Strategy

- Code review
- Security testing/fuzzing
- Attack Surface Reduction
- x86 only focus

CVE-2013-1796: Time MSR

- Out of bounds write to an atomic page

```
shared_kaddr = kmap_atomic(vcpu->time_page, KM_USER0);
memcpy(shared_kaddr + vcpu->time_offset, &vcpu->hv_clock,
    sizeof(vcpu->hv_clock));
```

- KVM checks starting offset of request not the entire length.
- Guest causes 30 byte write past end of page.

CVE-2013-1798: IOAPIC

- Nearly arbitrary host memory read

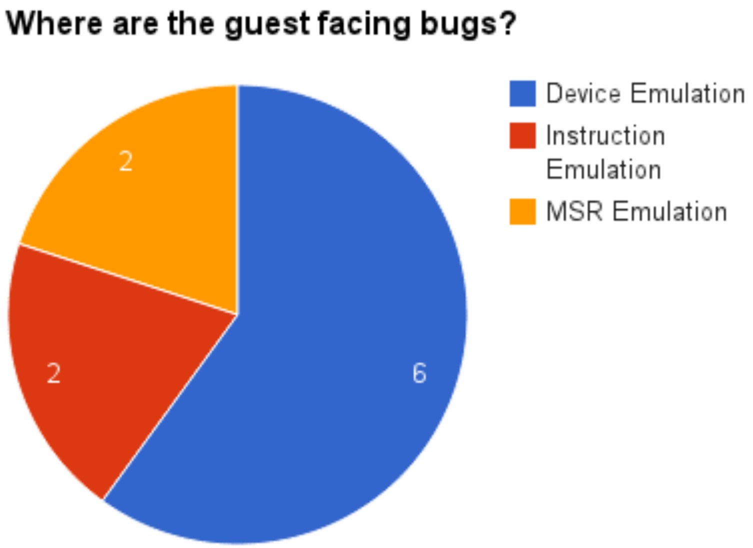
```
u32 redir_index = (ioapic->ioregsel - 0x10) >> 1;
u64 redir_content;
ASSERT(redir_index < IOAPIC_NUM_PINS);

redir_content = ioapic->redirtbl[redir_index].bits;
```

- Code uses ASSERT to verify valid index
- Assert compiles out in non-debug builds
- Guest reads arbitrary host memory

CVE-2014-0049: Instruction emulator

- Improper emulation of `pusha`
- Occurs when guest does pusha and stack starts in non-existent or mmio memory but finished in regular ram
- Allows guest to overwrite emulation data structures and leads to crash.
- VM Escape confirmed possible (although a bit racy)

Attack Surface Reduction



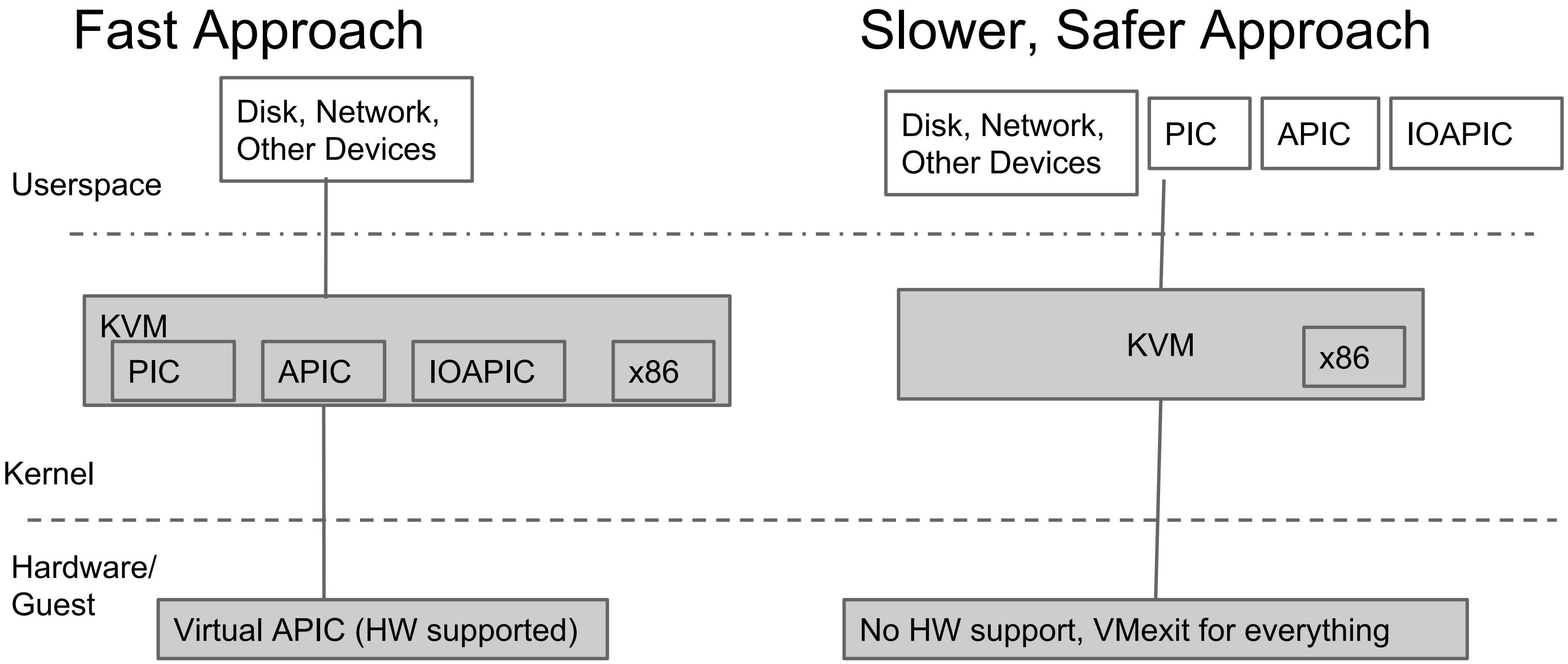
Moving attack surfaces to userspace VMM

- Vulnerability impact is greatly reduced
  - ASLR, stack canaries, AppArmor and other mitigations more common
  - VM escapes lead to userspace access only
  - DoS only affects the process of the VM, not others
- Very early in experimentation, comments, corrections, and better ideas are most welcome.

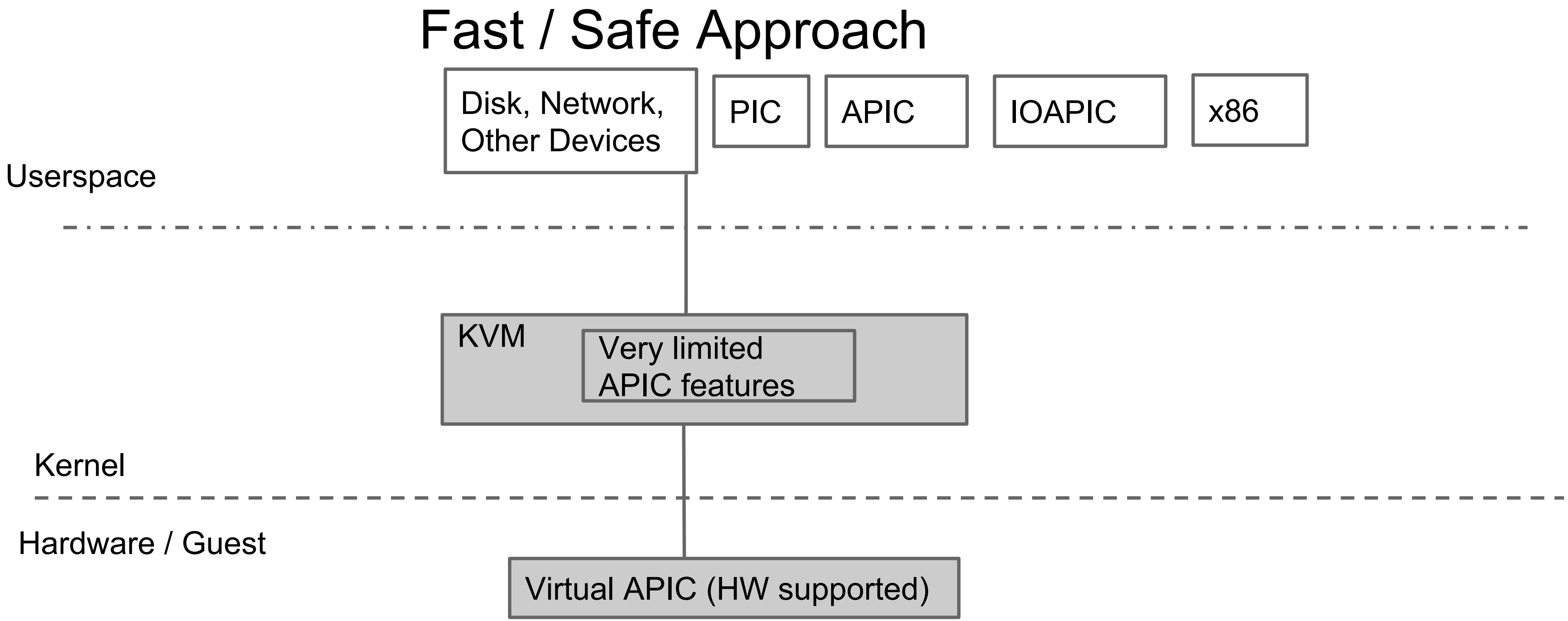
Approach

- Opt-in ways to move more functionality into userspace plus new interfaces to improve performance
- Start with all possible functionality in userspace and only cherry-pick what’s needed for performance
- Goal: >50% attack surface reduction with <.1% perf impact on macro benchmarks for modern guests on modern hardware.
- Attack surface metric:
    - Lines of code that process guest input
    - # pages of Intel SDM manual emulated

Current Options



What we're building



What must be in kernel?

- EOIs, TPR adjustments, and Self-IPIs definitely need to be in the kernel. Non self-IPIs, maybe.
- IOAPIC, PIC, and PIT are not perf critical.
- Emulator usually not perf critical.
- Some MSRs must stay in the kernel.

Experimental new ioctls/interfaces

- KVM_CREATE_IRQ_CHIP_LITE
  - Allows access to APIC page from userspace, kernel only enables apicv features
  - Kernel may need to support non-self IPIs, but via x2apic only
- KVM_SET_EOI_EXIT_BITMAP
- KVM_SET_EXIT_ON_EMULATION
- KVM_SET_MSR_EXIT_BITMAP
- Expanded kvm_run structure

Status

- Done some experimenting with KVM_CREATE_IRQ_CHIP_LITE and KVM_SET_MSR_EXIT_BITMAP with promising results
- Other ioctls in progress

Questions, Comments???