Nested Virtualization Design Session

Xen Design and Developer Summit, 11 July 2019 (https://design-sessions.xenproject.org/uid/discussion/disc_1NVcnOZyDZM1LpQbIsJm/view)

Related Presentations

- (2019) Jürgen Groß, Support of PV devices in nested Xen (https://youtube.com/watch?v=HA_teA6hV7c)
- (2019) Christopher Clark and Kelli Little, The Xen-Blanket (https://youtube.com/watch?v=i5w9sF9VerE)
- (2018) Ian Pratt, Hypervisor Security: Lessons Learned (https://youtube.com/watch?v=bNVe2y34dnM) (uXen)
- (2018) David Weston, Windows: Hardening with Hardware (https://youtube.com/watch?v=8V0wcqS22vc) (Credential Guard)

Use Cases

- Xen on Xen, some work was done for the Shim (Meltdown mitigation).
- Xen on another hypervisor, involves teaching Xen how to use enlightenments from other hypervisors.
- Qubes runs Xen on AWS bare-metal instances that use Nitro+KVM, mostly works.
- Windows Credential Guard (Hyper-V on Xen)
- Bromium Type-2 uXen in Windows and Linux guests on Xen

Issues

1. Need to be careful with features, eg. Ballooning down memory.
2. Dom0 is exposed to things that it should not see.
3. Nested virtualization works when both L0 and L1 agree, e.g Xen on Xen. When replacing Xen with another hypervisor, all falls apart.
4. Need more audit checks for what the VM can read or write, i.e. guest requirements.
5. Virtual vmentry and vmexit emulation "leaking", doesn't cope well.
6. Context switching bug fixed a while ago: doesn't understand AFR(?) loading or

whether it should do it or leave alone.

7. Missing instructions to virtualize vmexit.
8. Enlightened EPT shootdown is easy on top of the other features working.

Dependent on CPUID and MSR work

1. Auditing of changes. Can then fix virtual vmentry and vmexit, one bit at a time. Once all features are covered, it should work fine.
2. hwcaps: needed to tell the guest about the security state of the hardware.
3. Reporting CPU topology representation to guests, which is blocking core-scheduling work (presented by Juergen at Xen Summit)
4. Andrew is working on the prerequisites for the policy.

Validation of Nested Virtualization

1. First priority is correctness.
2. Second priority is performance.
3. There is a unit testing prototype which exercises vmxon, vmxoff instructions.
4. Depends on regression testing, which depends upon (a) formal security support, (b) approval of the Xen security team.
5. Other hypervisors must be tested with Xen.

Guest State

Nesting requires merge of L1 and L0 state.

1. AMD interface is much easier: it has "clean bits": if any bit is clear, must resync. Guest state is kept separately.
2. Intel guest state is kept in an opaque blob in memory, with special instructions to access it. Memory layout in RAM is unknown, behavior changes with microcode updates and there are 150 pages of relevant Intel manuals.
3. Bromium does some fun stuff to track guest state in software, poisoning RAM and then inspecting it, which is still faster than Intel's hardware-based VMCS shadowing. L1 hypervisor (Type-2 uXen): https://github.com/openxt/uxen
4. Viridian emulates the AMD way, i.e. Microsoft has Intel bits formatted in an AMD-like structure, then L0 translates the AMD structure into Intel's opaque blob.

Secure Variable Storage

1. Need an agreed sane way for multiple hypervisors to handle it, eg. a pair of ioports, translation from VMX, guest handles the interrupts via a standard ioport interception to secondary emulator: tiny.
2. Easy case: ioports + memory page for data.
3. Citrix XenServer has a closed-source implementation (varstored?)

Interface for nested PV devices

PV driver support currently involves grants and interrupts.

Requirements:

1. Should Xen's ABI include hypercall nesting level?
2. Each layer of nesting must apply access control decisions to the operation invoked by its guest.
3. Brownfield: if Xen and other L1 hypervisors must be compatible with existing Xen bare-metal deployments, the L0 hypervisor must continue to support grants, events and xenstore.
4. Greenfield: if the L0 hypervisor can be optimized for nesting, then PV driver mechanisms other than grants, events and xenstore could be considered.

Live migration with PCI graphics (has been implemented on AWS):

- need to make it look the same, regardless of nesting level
- 1 or more interrupts
- 1 or shared pages of RAM
- share xenstore
- virtio guest physical address space DMA: done right
- *need to get rid of domid* as the endpoint identifier

Access Control:

Marek: use virtio?
David: do whatever you like in L1
Juergen: new "nested hypercall", to pass downwards an opaque payload
David: how does access control work with that approach?

Christopher: xenblanket RFC series implements support for one level of nesting. Its implementation below the hypercall interface demonstrates access control logic that is required at each nesting level.