OpenBSD vmm/vmd Update

Mike Larkin

bhyvecon 2019
20 Mar 2019 – Tokyo, Japan

Agenda

- Where we were a year ago
- Current status
- Future plans
- Q&A

One Year Ago ...

- Reasonably complete support for OpenBSD and Linux guests
- amd64 and i386 host support
- SVM/VMX support
- Scaffolding and tools to support the above
  - vmd(8)/vmctl(8)

This Past Year ...

- Adding new/core features
  - Disk snapshotting
  - Template VMs
- Security Improvements
  - Removing lazy FPU support
  - L1TF mitigation
- Platform improvements
  - Bug fixing / paying down technical debt

This Past Year (cont'd) ...

- Community involvement
  - Commercial deployments of vmm hosting providers
  - Usage of vmm(4) without vmd(8) for other use cases

2018 vmm(4) Improvements

- Platform improvements
- Correctness improvements
- Performance/stability improvements
- Security improvements

- Some of these improvements impart new functionality, some are bug fixes

2018 vmm(4) Platform Improvements

- Platform improvements
  - Instruction emulation improved
  - Support added for qemu fw_cfg interface
  - Support guest OS %drX registers
  - Platform support for PXE boot
  - Implement missing PIC functionality

2018 vmm(4) Platform Improvements

- Instruction emulation fixes/improvement
  - RDTSCP – Incorrect implementation broke SmartOS boot
  - MONITOR/MONITORX – Broke booting Linux on Ryzen hosts
- QEMU fw_cfg interface support
  - Allows passing boot parameters from SeaBIOS into the VM

2018 vmm(4) Platform Improvements

- Support for guest %drX registers
  - Allows hardware breakpoint usage inside guest VM
  - (OpenBSD doesn’t use these itself, was a subject of a security vulnerability affecting other OSes last year)

2018 vmm(4) Platform Improvements

- Platform support for PXE boot
  - Implemented after last EuroBSDcon
  - Requires iPXE extension ROM image
  - Can be handled for OpenBSD guests differently (discussed later)
- Implemented missing PIC functionality
  - Basically bug fixes

2018 vmm(4) Correctness Improvements

- Correctness improvements
  - Many fixes in CPUID emulation
  - Add support for older CPUs without XSAVE
  - Handle certain SMM-related MSRs properly

2018 vmm(4) Correctness Improvements

- CPUID improvements
  - Handle misreported large leaf function #s
  - Proper topology reporting
  - Handle bizarre “rex extended CPUID” instruction used in TempleOS
  - Properly report physical address limits for the host CPU
    - Allows VMs with much larger memory

2018 vmm(4) Correctness Improvements

- Support CPUs without XSAVE
  - Older CPUs don’t have this
- Handle reserved SMM-related MSRs
  - SDM reference guide says these should #GP on use (previously ignored, or returned 0)

2018 vmm(4) Performance Improvements

- We improved the SVM situation significantly last year …
  - Interrupt window handling was totally broken before (fixed)
  - RFLAGS.IF handling was totally broken before (fixed)
  - Each exit would lock/unlock the kernel lock up to 4 times during exit processing before (now zero)

2018 vmm(4) Performance Improvements

- #UD on VMX instructions
  - “Inspired” by a KVM bug
  - Previously, guest usermode program could crash the VM since these instructions exit **before** checking CPL
    - We would terminate the VM before …
- #GP on invalid %cr0 / %cr4 bits
  - Previously terminated the guest

2018 vmm(4) Performance Improvements

- Many of these improvements replaced “terminate the guest” with functionality appropriate for the case
  - The “terminate the guest” on anything unexpected was a remnant from early development
  - We can start to relax these conditions now

2018 vmm(4) Security Improvements

- Removed lazy FPU handling as part of the larger OS-wide effort
- And of course there was L1TF last August...

2018 vmm(4) Security Improvements

- L1TF primer
  - Allows read of data in L1 cache
  - EPT addresses are treated as physical addresses (!)
  - Basically means a guest can read data out of L1 that likely was placed there while running in VMX root mode

2018 vmm(4) Security Improvements

- L1TF entry semantics (now)
  - Flush L1 cache
  - Enter guest
  - …
- How do you flush L1?
  - And is it only L1D or is there L1I → L1D leakage too?

2018 vmm(4) Security Improvements

- New microcode has “flush L1” command MSR
- What if you don’t have the new microcode?
  - Read a bunch of junk, hopefully fill all of L1D what you read
  - What about the cachelines you touch after that, but before the entry (guest CPU registers)?
  - And what about L1I, anyway?

2018 vmm(4) Security Improvements

- Our L1TF ‘junk’ data consists of 64KB of ‘0xcc’, ***just in case*** there is L1D → L1I leakage
  - Of course nobody who knows has said anything

2018 vmm(4) Security Improvements

- Maxime from NetBSD also reported a bug in our handling of xsetbv arguments
- Thanks Maxime!

2018 vmd(8)/vmctl(8) Improvements

- Most of the more impactful improvements came in vmd(8) and vmctl(8)
  - Qcow2 disk support
  - Disk snapshots
  - Template VMs
  - More user friendly vmctl(8) options

2018 vmctl(8)/vmd(8) Improvements

- Qcow2 disk support
  - Supported in “standalone” or “base + snapshot” mode
  - Integrated into vmctl(8) and vmd(8)
- Old “raw” format still supported
  - Both modes “sparse” but qcow2 is “lazy allocated” (image grows over time)

2018 vmd(8)/vmctl(8) Improvements

- Qcow2 (cont'd)
  - vmctl(8) can create qcow2 disks:

```
-kadath- ~> vmctl create foo.raw -s 10g
vmctl: raw imagefile created
-kadath- ~> vmctl create foo.qcow2 -s 10g
vmctl: qcow2 imagefile created
-kadath- ~> ls -la foo.*
-rw-------  1 mlarkin  wheel       262144 Mar 18 21:30 foo.qcow2
-rw-------  1 mlarkin  wheel  10737418240 Mar 18 21:30 foo.raw
```

2018 vmd(8)/vmctl(8) Improvements

- Qcow2 (cont'd)
  - vmctl(8) can convert disks:

```
-kadath- ~> vmctl create foo2.raw -i foo.qcow2
vmctl: raw imagefile created
-kadath- ~> ls -la foo*
-rw-------  1 mlarkin  wheel       262144 Mar 18 21:30 foo.qcow2
-rw-------  1 mlarkin  wheel  10737418240 Mar 18 21:30 foo.raw
-rw-------  1 mlarkin  wheel  10737418240 Mar 18 21:33 foo2.raw
```

2018 vmd(8)/vmctl(8) Improvements

- Qcow2 (cont'd)
  - Sparseness is preserved:

```
-kadath- ~> du -h foo*
192K    foo.qcow2
192K    foo.raw
192K    foo2.raw
```

2018 vmd(8)/vmctl(8) Improvements

- Qcow2 (cont'd)
  - Base image + snapshot:

```
-kadath- ~> vmctl create derived.qcow2 -s 10G -b foo.qcow2
vmctl: qcow2 imagefile created
-kadath- ~> ls -la *qcow2
-rw-------  1 mlarkin  wheel  262144 Mar 18 21:37 derived.qcow2
-rw-------  1 mlarkin  wheel  262144 Mar 18 21:30 foo.qcow2
```

2018 vmd(8)/vmctl(8) Improvements

- Qcow2 (cont'd)
  - Base image + snapshot accumulates all disk changes in snapshot disk
  - Rollback?
    - rm derived.qcow2
    - Restore previous derived.qcow2, restart VM
  - It would be nice to have rollback/rollforward be a new vmctl option (any takers?)

2018 vmd(8)/vmctl(8) Improvements

- vmctl(8) new command options for easier VM management
  - vmctl start -B xxx
    - Set boot device (OpenBSD guests)
    - Used for autoinstalling guest VMs via network (vmctl start -B net …)
  - vmctl stop -a
    - Stop all VMs (used for shutdown scripts)

2018 vmd(8)/vmctl(8) Improvements

- vmctl(8) new command options for easier VM management
  - vmctl stop -f
    - Force kill (terminate) a VM
    - Don’t wait for vmmci(4)

2018 vmd(8)/vmctl(8) Improvements

- Template VMs
  - vmctl start -t
  - Allows for quick and easy “cloning” of VM settings

```
-t name     Use an existing VM with the specified name as a
            template to create a new VM instance.  The instance
            will inherit settings from the parent VM, except for
            exclusive options such as disk, interface lladdr, or
            interface names.
```

2018 vmm(4)/vmd(8) Misc Improvements

- We finally retired i386 hosts
  - It served its purpose during early development
  - Found a lot of bugs
  - Wasn’t really worth maintaining anymore
- Of course i386 guests still work

2019 Goals

- We did pretty well reducing the bug count in 2018
  - But there are still many
- Solicit community involvement
  - Glad to have lots of new faces at the vmm table
- SMP is likely my personal #1 goal
  - We’ve done just about everything else interesting

New Ideas For vmm(4)

- Underjack update
- Nested virtualization update

New Ideas For vmm(4)

- Last year I talked about the underjack approach
  - Putting vmm(4) underneath the host
  - Run host as a VM itself
  - Allows XO (execute only) memory in the host
- XO memory is one defence against ROP attacks
  - Go see Todd Mortimer’s talk about RETGUARD this week for another defence!

New Ideas For vmm(4)

- Underjack (cont'd)
  - Kernel is working (was completed after last year's BhyveCon)
  - How do you handle running VMs in vmm(4) when the host machine itself is a VM?

New Ideas For vmm(4)

- Host/root partition approach
  - Host treated as VM until launching a new (child) VM in vmm(4) via vmctl(8)
  - Temporarily exit host VM
  - Enter guest context as usual
  - Re-enter host VM context after exit
  - Repeat ad nauseum
- This approach treats the host and guest VMs as peers of each other
  - Difficult to support nested XO memory

New Ideas For vmm(4)

- Nested VMX approach
  - Never leave VMX mode
  - Host VM launches VMs of its own
    - Host VM becomes nested hypervisor
  - Can more easily accomplish nested XO
- The first approach is easier to code
- The second approach allows for arbitrary levels of nesting

New Ideas For vmm(4)

- Nested VMX approach status
  - Does “emulated” VMCS (no VMCS shadowing)
    - Slow
- May decide at some point to switch approaches
  - Security improvement (XO memory) vs functionality (nested VMs) decision

New Ideas For vmm(4)

- Nested VMX update
  - Boots OpenBSD/vmm(4) and Linux/KVM guests
  - Needs to be redone to use shadow VMCS
    - Tons of VMCS traffic
    - Lots of issues for 32 bit hypervisor hosts if not done (HI/LO VMCS fields handled separately)
    - Maybe we don’t care

New Ideas For vmm(4)

- pvclock(4)
  - Paravirtualized clock
  - Modeled after KVM’s PV clock interface
  - Should hopefully help time skews and high CPU usage for applications doing lots of gettimeofday() or equivalent

Community Involvement

- I’d like to take a few minutes to point out a few things going on in the community …

- OpenBSD.amsterdam
  - Hosted vmm(4) VMs
  - Part of the hosting fee is donated to the OpenBSD foundation

Community Involvement

- OpenBSD.amsterdam (cont'd)
  - 238 VMs deployed since last year, across 7 servers
- BhyveCon referral/discount code
  - 'BhyveCon' (5 EUR discount)

Community Involvement

- Solo5
  - Sandboxed environment for running unikernels
  - Support added for using vmm(4) as a backend hypervisor
- Would love to see more integrations like this

Questions?

- Any questions?

Thank You

Mike Larkin
mlarkin@openbsd.org
@mlarkin2012