CMPE-283 Assignment-1 Report

Write a Linux kernel module to discover VMX feature

Purpose

Learn how to write the Linux kernel module to discover the Intel CPU VMX feature.
Note: single member group.

Detail Recipe

1. Setup the Linux environment

a. For this experiment, Intel CPU is required.
b. We are using Google Cloud Platform, follow GCP tutorial to create a new VM instance, key configurations of the VM is shown as below.

Config	Info
CPU	Intel Broadwell, 1 vCPU
Linux Kernel Version	4.15.0-1027-gcp
Linux Release Version	Ubuntu 16.04

c. Configure the VM so that we can scp file between our host and the VM.
   i. Ref: https://cloud.google.com/compute/docs/instances/transfer-files
   ii. Enable OS login on VM instance.
   iii. Assign user account the roles in IAM.
   iv. Use gcloud SDK from host to do SSH/SCP.

2. Enable VMX virtualization features in the Linux

a. For GCP, this requires us to enable nested virtualization. The main idea here is to add the VMX license to a normal image.
b. Ref: https://cloud.google.com/compute/docs/instances/enable-nested-virtualization-vm-instances
c. This step is mandatory, otherwise when the kernel module tries to dump MSRs, it would report error as below.

```
CMPE 283 Assignment 1 Module Start
unchecked MSR access error: RDMSR from 0x480 at rIP: 0xffffffffbdc6d1ba (native_read_msr+0xa/0x30)
Call Trace:
 detect_vmx_features+0x14/0x1e0 [cmpe283_1]
 init_module+0x1a/0x20 [cmpe283_1]
 do_one_initcall+0x52/0x1a4
 ? __vunmap+0x81/0xb0
 ? _cond_resched+0x19/0x40
 ? kmem_cache_alloc_trace+0x14e/0x1b0
 ? do_init_module+0x27/0x209
 do_init_module+0x5f/0x209
 load_module+0x191e/0x1f10
 ? ima_post_read_file+0x96/0xa0
 SYSC_finit_module+0xfc/0x120
 ? SYSC_finit_module+0xfc/0x120
 SyS_finit_module+0xe/0x10
 do_syscall_64+0x7b/0x150
 entry_SYSCALL_64_after_hwframe+0x42/0xb7
RIP: 0033:0x7f0805b85839
RSP: 002b:00007ffed5e740c8 EFLAGS: 00000246 ORIG_RAX: 0000000000000139
RAX: ffffffffffffffda RBX: 000055c6d1cba7a0 RCX: 00007f0805b85839
RDX: 0000000000000000 RSI: 000055c6d1718d2e RDI: 0000000000000003
RBP: 000055c6d1718d2e R08: 0000000000000000 R09: 00007f0805e58000
R10: 0000000000000003 R11: 0000000000000246 R12: 0000000000000000
R13: 000055c6d1cba760 R14: 0000000000000000 R15: 0000000000000000
```

d. Verify the nested VM has been enabled, the following cmd should return 1.

```
# grep -cw vmx /proc/cpuinfo
```

3. Download and build the Linux kernel source code

a. This step is optional for assignment 1, I verified that we can build the Linux kernel in any path of a Linux environment, and insert the module to the current kernel to verify its functionality. Since assignment 2 will require kernel compilation and build, it would be better to compile it right now so as to save time since till then only the modified modules will need to be re-compiled.

b. Clone the Linux repo:

```
# git clone https://github.com/torvalds/linux.git
# cd linux
```

c. Copy your kernel config file to the current directory. You'll find it in /boot, and the name will vary but it will general start with "config". The new name should be ".config":

```
# copy   /boot/config-4.15.0-29-generic     .config
```

d. `# make oldconfig` (and answer 'y' or default answer to each question)

e. `# sudo bash`

f. `# make && make modules && make install && make modules_install`

g. One trick for the above step, increase vCPU number and use `make -j#` to accelerate the compile, then revert to single vCPU to save budget.

h. reboot, and select the new kernel during the boot process

i. For the following steps we will work in the linux/ folder as created in the above steps.

j. Get information of what Linux commit we are working on:

```
# git log
commit a215ce8f0e00c2d707080236f1aafec337371043
Merge: 2d28e01 cffaaf0
Author: Linus Torvalds <torvalds@linux-foundation.org>
Date: Fri Mar 1 09:13:04 2019 -0800
```

k. Create separate folder for our module

```
# mkdir cmpe283
# cd cmpe283
```

l. SCP the given skeleton source code and Makefile to the VM via gcloud SDK from the host

```
# gcloud compute scp cmpe283-1.c ubuntu16-nested-vm:~/linux/cmpe283
# gcloud compute scp Makefile ubuntu16-nested-vm:~/linux/cmpe283
```

4. Create the new kernel module

a. Before touching any code, it would be better to get some background knowledge regarding kernel module programming. Ref to: https://linux.die.net/lkmpg/index.html

b. Modify the code to add the following functionalities:

i. Ref to SDM to add data structures for proc-based control field, secondary proc-based control field, exit control field, entry control field.

ii. Determine if CPU support VMX true controls

iii. Based on the above result, decide which MSR fields to reference.

iv. For the secondary proc-based control MSR, there is only one choice.

v. Expose VMX features by calling the “report_capability()” helper.

5. Load and verify the new kernel module

a. Make the module source code

```
# make
```

b. Insert the module into current Linux kernel

```
# sudo insmod cmpe283-1.ko
```

c. Check system message to verify our functionality

```
# dmesg
```

d. Remove the module from the kernel

```
# sudo rmmod cmpe283_1
```

e. Repeat the above procedure until you reach expected result.

6. Commit modification and generate diff file to submit

a. Git add files to reflect modification in our work directory to the index

```
# git add cmpe283/cmpe283-1.c
# git add cmpe283/Makefile
```

b. Git commit to stage the modification into our local repository

```
# git commit -m “Assignment #1 Finished.”
```

c. Git diff between the latest commit with the fresh starting point

```
# git diff <old commit id> <new commit id> > cmpe283-1.diff
```

Appendix:

1. Print information of our module on the GCP VM.

```
# dmesg
[16374.273943] CMPE 283 Assignment 1 Module Start
[16374.273950] Can CPU Support True VMX Feature? Y | 0xd8100011e57ed0
[16374.273952] Pinbased Controls MSR: 0x3f00000016
[16374.273953] can_set | can_clear | feature
[16374.273954]    O    |    O      | External Interrupt Exiting
[16374.273954]    O    |    O      | NMI Exiting
[16374.273955]    O    |    O      | Virtual NMIs
[16374.273956]    X    |    O      | Activate VMX Preemption Timer
[16374.273956]    X    |    O      | Process Posted Interrupts
[16374.273958] Procbased Controls MSR: 0xf7b9fffe04006172
[16374.273959] can_set | can_clear | feature
[16374.273959]    O    |    O      | Interrupt Window Exiting
[16374.273960]    O    |    O      | Use TSC Offsetting
[16374.273961]    O    |    O      | HLT Exiting
[16374.273962]    O    |    O      | INVLPG Exiting
[16374.273962]    O    |    O      | MWAIT Exiting
[16374.273963]    O    |    O      | RDPMC Exiting
[16374.273964]    O    |    O      | RDTSC Exiting
[16374.273964]    O    |    O      | CR3 Load Exiting
[16374.273965]    O    |    O      | CR3 Store Exiting
[16374.273965]    O    |    O      | CR8 Load Exiting
[16374.273966]    O    |    O      | CR8 Store Exiting
[16374.273967]    O    |    O      | Use TPR Shadow
[16374.273967]    X    |    O      | NMI Window Exiting
[16374.273968]    O    |    O      | MOV-DR Exiting
[16374.273969]    O    |    O      | Unconditional IO Exiting
[16374.273969]    O    |    O      | Use IO Bitmaps
[16374.273977]    X    |    O      | Monitor Trap Flag
[16374.273978]    O    |    O      | Use MSR Bitmaps
[16374.273979]    O    |    O      | Monitor Exiting
[16374.273979]    O    |    O      | PAUSE Exiting
[16374.273980]    O    |    O      | Activate Secondary Controls
[16374.273982] Secondary Procbased Controls MSR: 0x51df00000000
[16374.273983] can_set | can_clear | feature
[16374.273983]    O    |    O      | Virtualize APIC Access
[16374.273984]    O    |    O      | Enable EPT
[16374.273985]    O    |    O      | Descriptor-table Exiting
[16374.273985]    O    |    O      | Enable RDTSCP
```

[16374.273986] O | O | Virtualize x2APIC Mode
[16374.273986] X | O | Enable VPID
[16374.273987] O | O | WBIVD Exiting
[16374.273988] O | O | Unrestricted Guest
[16374.273989] O | O | APIC-register Virtualization
[16374.273989] X | O | Virtual-interrupt Delivery
[16374.273990] X | O | PAUSE-loop Exiting
[16374.273991] X | O | RDRAND Exiting
[16374.273991] O | O | Enable INVPCID
[16374.273992] X | O | Enable VM Functions
[16374.273992] O | O | VMCS Shadowing
[16374.273993] X | O | Enable ENCLS Exiting
[16374.273994] X | O | RDSEED Exiting
[16374.273994] X | O | Enable PML
[16374.273995] X | O | EPT-violation #VE
[16374.273995] X | O | Conceal VMC from PT
[16374.273996] X | O | Enable XSAVES or XRSTORS
[16374.273997] X | O | Mode-based Execute Control for EPT
[16374.273997] X | O | Use TSC Scaling
[16374.273998] X | O | Enable ENCLV Exiting
[16374.274000] Exit Controls MSR: 0x3fefff00036dfb
[16374.274000] can_set | can_clear | feature
[16374.274001] O | O | Save Debug Controls
[16374.274001] O | O | Host Addr-space Size
[16374.274002] X | O | Load IA32_PERF_GLOBAL_CTRL
[16374.274002] O | O | Acknowledge Interrupt on Exit
[16374.274003] O | O | Save IA32_PAT
[16374.274004] O | O | Load IA32_PAT
[16374.274004] O | O | Save IA32_EFER
[16374.274005] O | O | Load IA32_EFER
[16374.274006] X | O | Save VMX preemption Timer Value
[16374.274006] X | O | Clear IA32_BNDCFGS
[16374.274007] X | O | Conceal VMX from PT
[16374.274008] Entry Controls MSR: 0xd3ff000011fb
[16374.274009] can_set | can_clear | feature
[16374.274009] O | O | Load Debug Controls
[16374.274010] O | O | IA-32e Mode Guest
[16374.274010] X | O | Entry to SMM
[16374.274011] X | O | Deactivate Dual-monitor Treatment
[16374.274012] X | O | Load IA32_RERF_GLOBAL_CTRL
[16374.274012] O | O | Load IA32_PAT
[16374.274013] O | O | Load IA32_EFER
[16374.274013] X | O | Load IA32_BNDCFGS
[16374.274014] X | O | Conceal VMX from PT
[16488.469822] CMPE 283 Assignment 1 Module Exits