A case stufy of QEMU and AddressSanitizer

Dongli Zhang ⋮ 8-10 minutes

Introduction

The memory address sanitizer, AddressSanitizer, is used to detect memory bugs, e.g., use-after-free, out-of-bounds access, double-free or memory leak. This article provides a case study about AddressSanitizer in QEMU in userspace.

In this article, we demonstrate how we detect a QEMU memory leak bug with AddressSanitizer, using an OCI VM instance as the KVM hypervisor. The fix for which has already been accepted by QEMU.

Step 1. Create an OCI VM

Create an OCI instance with Oracle Linux 7. We use the VM.Standard2.4 shape in this article.

Step 2. Configure yum

Once your instance is created, enable the yum repository **ol7_developer_EPEL** in order to install the necessary prerequisite packages.

```
$ sudo yum-config-manager --enable ol7_developer_EPEL

$ cat /etc/yum.repos.d/oracle-epel-ol7.repo
[ol7_developer_EPEL]
name=Oracle Linux $releasever EPEL Packages for Development ($basearch)
baseurl=https://yum$ociregion.$ocidomain/repo/OracleLinux/OL7/developer_EPEL/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
```

Now install the required packages:

```
$ sudo yum install -y targetcli ninja-build glib2-devel pixman-devel \
zlib-devel devtoolset-10 devtoolset-10-libasan-devel
```

Step 3. Configure gcc

In order to use gcc-10 we need to enable the devtoolset-10 software collection:

```
$ scl enable devtoolset-10 bash
```

Please double check that the gcc version is **10.2.1**.

```
$ gcc -v
... ...
gcc version 10.2.1 20210130 (Red Hat 10.2.1-11.1.0.1) (GCC)
```

To ensure this software collection is enabled permanently across logins, add the above `scl` command to */etc/profile*.

Step 4. Build QEMU

We use qemu-7.1.0 as an example.

The option **-j8** is provided to indicate we want to run 8 jobs in parallel during the build. This number usually reflects the number of CPUs available, which in our case is 8. If for instance you had 16 CPUs available, you could specify **-j16**.

Copied to Clipboard

Error: Could not Copy

Copied to Clipboard

Error: Could not Copy

```
$ cd /home/opc
$ wget https://download.qemu.org/qemu-7.1.0.tar.xz
$ tar xvf qemu-7.1.0.tar.xz
$ cd qemu-7.1.0

$ mkdir build
$ cd build

$ ../configure --target-list=x86_64-softmmu --enable-sanitizers --enable-debug
$ make -j8 > /dev/null
```

The QEMU binary is located at:

Copied to Clipboard

Error: Could not Copy

Copied to Clipboard

Error: Could not Copy

```
/home/opc/qemu-7.1.0/build/x86_64-softmmu/qemu-system-x86_64
```

Step 5. Create vhost-scsi target

Copied to Clipboard

Error: Could not Copy

Copied to Clipboard

Error: Could not Copy

```
$ dd if=/dev/zero of=/home/opc/storage.raw bs=1M count=128
$ sudo targetcli /backstores/fileio create name=storage file_or_dev=/home/opc/storage.raw
$ sudo targetcli /vhost create naa.1123451234512345
$ sudo targetcli /vhost/naa.1123451234512345/tpg1/luns create /backstores/fileio/storage
```

The below output from targetcli illustrates the vhost-scsi target configuration:

Copied to Clipboard

Error: Could not Copy

Copied to Clipboard

Error: Could not Copy

```
$ sudo targetcli ls
o- / ...........................................................................................................
[...]
  o- backstores
```

```
.......................................................................................................
[...]
  | o- block
.................................................................................................... [Storage
Objects: 0]
  | o- fileio
................................................................................................... [Storage
Objects: 1]
  | | o- storage ........................................................... [/home/opc/storage.raw
(128.0MiB) write-back activated]
  | |   o- alua
..................................................................................................... [ALUA
Groups: 1]
  | |     o- default_tg_pt_gp ........................................................................ [ALUA
state: Active/optimized]
  | o- pscsi
.................................................................................................... [Storage
Objects: 0]
  | o- ramdisk
.................................................................................................. [Storage
Objects: 0]
  o- iscsi
.......................................................................................................
[Targets: 0]
  o- loopback
....................................................................................................
[Targets: 0]
  o- vhost
.......................................................................................................
[Targets: 1]
    o- naa.1123451234512345
................................................................................................ [TPGs: 1]
      o- tpg1 ..............................................................................
[naa.5001405ba9aa6ea7, no-gen-acls]
        o- acls
.....................................................................................................
[ACLs: 0]
        o- luns
.....................................................................................................
[LUNs: 1]
          o- lun0 ....................................................... [fileio/storage
(/home/opc/storage.raw) (default_tg_pt_gp)]
```

Step 6. Create a QEMU instance with vhost-scsi-pci

Since the objective is to demonstrate the memory leak at **vhost-scsi**, a boot image is not required.

Copied to Clipboard

Error: Could not Copy

Copied to Clipboard

Error: Could not Copy

```
$ sudo /home/opc/qemu-7.1.0/build/x86_64-softmmu/qemu-system-x86_64 \
  -m 4G -smp 2 -machine pc,accel=kvm -vnc :0 \
  -device vhost-scsi-pci,wwpn=naa.1123451234512345,num_queues=2
```

Step 7. Kill the QEMU instance

After a while, we kill the QEMU instance with **Ctrl + c**. You will notice that AddressSanitizer reports the memory leak as illustrated below:

Copied to Clipboard

Error: Could not Copy

Copied to Clipboard

Error: Could not Copy

```
=================================================================
==23486==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 40 byte(s) in 1 object(s) allocated from:
    #0 0x7f21f525b917 in __interceptor_calloc (/lib64/libasan.so.6+0xb4917)
    #1 0x7f21f44327b5 in g_malloc0 (/lib64/libglib-2.0.so.0+0x517b5)
    #2 0x55bbb48f8fed in vhost_scsi_start ../hw/scsi/vhost-scsi.c:91
    #3 0x55bbb48f91c1 in vhost_scsi_set_status ../hw/scsi/vhost-scsi.c:130
    #4 0x55bbb49641a7 in virtio_set_status ../hw/virtio/virtio.c:1997
    #5 0x55bbb44e4df7 in virtio_pci_common_write ../hw/virtio/virtio-pci.c:1299
    #6 0x55bbb49fd21d in memory_region_write_accessor ../softmmu/memory.c:492
    #7 0x55bbb49fd57f in access_with_adjusted_size ../softmmu/memory.c:554
    #8 0x55bbb4a053e9 in memory_region_dispatch_write ../softmmu/memory.c:1514
    #9 0x55bbb4a26a0a in flatview_write_continue ../softmmu/physmem.c:2825
    #10 0x55bbb4a26d5d in flatview_write ../softmmu/physmem.c:2867
    #11 0x55bbb4a2766c in address_space_write ../softmmu/physmem.c:2963
    #12 0x55bbb4a27724 in address_space_rw ../softmmu/physmem.c:2973
    #13 0x55bbb4b71b78 in kvm_cpu_exec ../accel/kvm/kvm-all.c:2954
    #14 0x55bbb4b79d3a in kvm_vcpu_thread_fn ../accel/kvm/kvm-accel-ops.c:49
    #15 0x55bbb4fa6131 in qemu_thread_start ../util/qemu-thread-posix.c:504
    #16 0x7f21f3643ea4 in start_thread (/lib64/libpthread.so.0+0x7ea4)

SUMMARY: AddressSanitizer: 40 byte(s) leaked in 1 allocation(s).
```

The fix for this bug has been accepted by upstream QEMU:

- vhost-scsi: fix memleak of vsc->inflight
  https://gitlab.com/qemu-project/qemu/-/commit/aba0d042b1c1be38818cec16af3f34e9e9e2aed2

Summary

In this article, we have demonstrated how we detect a QEMU memory leak bug with AddressSanitizer on an OCI VM instance. The version used in this article is qemu-7.1.0. The issue is not reproducible any longer as the bugfix has been applied.

Dongli Zhang