Virtio Device Fuzzing

Dima Stepanov
Software Engineer, Yandex.Cloud Hypervisor Team
Table Of Contents

01 What is Fuzzing?
02 How We Added Fuzzing
03 Issues Found
04 Future Improvements
05 Useful Links
What is Fuzzing?
Fuzzing Technique: Part 1
black box, fuzz, test, crash

Fuzz Engine \(\rightarrow\) Input \(\rightarrow\) Target App

Status
Fuzzing Technique: Part 2

grey box, coverage guided

Fuzz Engine \[\rightarrow\] Input \[\rightarrow\] Target App

Target App \[\rightarrow\] Status

Code Path
Attack Surface
How We Added Fuzzing
Fuzzing Scheme

Fuzzer Engine → VIRTQ → VIRTIO Backend

Status
Real Fuzzing

AFL

fork()

QEMU/SPDK

Status

Unix Socket

Proxy

VIRTQ
AFL-Proxy Communication

AFL
SOCK
VIRTQ

connect()

send()

AFL SOCK
Proxy
Proxy-QEMU Communication

Proxy

QTEST

SOCK

QEMU/QTEST

BLKDEV

QMP
QEMU Status

code instrumentation

AFL

afl-clang

Code Path

VIRTIO

Code

QEMU
QEMU Status
persistent mode

AFL ⏯️ SIGSTOP 🔄 Reset Device ⏯️ QEMU
Real Fuzzing

AFL

fork()

Status

QEMU/SPDK

Unix

Socket

Proxy

VIRTQ
Input

Data?
Input

Data?

GUESTMEM
Input

Data?

→ GUESTMEM

→ VIRTBLK
Input

Data?

GUESTMEM

VIRTBLK

VIRTDESC

QEMU/SPDK
### Process Timing

- **Run time**: 13 days, 6 hrs, 22 min, 28 sec
- **Last new path**: 6 days, 23 hrs, 38 min, 45 sec
- **Last uniq crash**: none seen yet
- **Last uniq hang**: 10 days, 7 hrs, 0 min, 20 sec

### Cycle Progress

- **Now processing**: 20* (14.08%)
- **Paths timed out**: 0 (0.00%)

### Stage Progress

- **Now trying**: splice 11
- **Stage execs**: 47/48 (97.92%)
- **Total execs**: 5.61G
- **Exec speed**: 5170/sec

### Fuzzing Strategy Yields

- **Bit flips**: 19/328k, 2/328k, 2/327k
- **Byte flips**: 0/41.0k, 0/40.9k, 0/40.6k
- **Arithmetics**: 4/2.30M, 1/455k, 2/184k
- **Known ints**: 1/322k, 1/1.07M, 5/1.71M
- **Dictionary**: 0/0, 0/0, 0/1.91M
- **Havoc**: 38/2.18G, 11/3.42G
- **Trim**: 8.07%/6048, 0.00%

### Overall Results

- **Cycles done**: 173k
- **Total paths**: 142
- **Uniq crashes**: 0
- **Uniq hangs**: 1

### Map Coverage

- **Map density**: 0.12% / 0.48%
- **Count coverage**: 1.82 bits/tuple

### Findings in Depth

- **Favored paths**: 21 (14.79%)
- **New edges on**: 41 (28.87%)
- **Total crashes**: 0 (0 unique)
- **Total tmouts**: 9 (1 unique)

### Path Geometry

- **Levels**: 7
- **Pending**: 0
- **Pend fav**: 0
- **Own finds**: 86
- **Imported**: n/a
- **Stability**: 26.98%
03

Issues Found
VIRTIO assert (QEMU)

assert()

```c
def struct virtq_desc {
    le64 addr;       /* Address (guest-physical). */
    le32 len;        /* Length. */
    le16 flags;      /* The flags as indicated above. */
    le16 next;       /* Next field if flags & NEXT */
};
```

```bash
$ hexdump -C ./id0001.dump
00000000  40 e0 fd 3f 00 00 00 00                                   |..|
00000010  00 00 00 00 00 00 06 00 04 00 |..|
```

```bash
00000010  00 00 00 00 00 00 00 00
```
SIGSEGV (SPDK)

memory access violation

    struct virtq_desc {
        le64 addr;       /* Address (guest-physical). */
        le32 len;        /* Length. */
        le16 flags;      /* The flags as indicated above. */
        le16 next;       /* Next field if flags & NEXT */
    };

# hexdump -C sigsegv.dump
00000000  30 00 00 00 00 00 00 00  01 00 00 00 01 00 01 00 |..|
00000010  40 00 00 00 00 00 00 00  01 00 00 00 02 00 03 00 |..|
00000020  40 02 00 00 00 00 00 00  01 00 00 00 02 00 03 00 |..|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |..|
CVE-2019-9547 (SPDK)

endless loop, DoS

```c
struct virtq_desc {
    le64 addr; /* Address (guest-physical). */
    le32 len; /* Length. */
    le16 flags; /* The flags as indicated above. */
    le16 next; /* Next field if flags & NEXT */
};
```

```bash
# hexdump -C hang01.dump
00000000 b0 00 00 00 00 dd ff 80 00 00 00 00 01 00 01 00 [..]
00000010 40 14 00 00 f3 00 00 00 00 00 00 00 05 00 00 00 [..]
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [..]
```
04

Future Improvements
Improvement Plans

1. Fuzz virtio-net, virtio-sock, virtio-*
Improvement Plans

1. Fuzz virtio-net, virtio-sock, virtio-*
2. Fuzz PCI
Improvement Plans

1. Fuzz virtio-net, virtio-sock, virtio-*
2. Fuzz PCI
3. Enhance start corpus
Improvement Plans

1. Fuzz virtio-net, virtio-sock, virtio-*
2. Fuzz PCI
3. Enhance start corpus
4. Update Qtest
Improvement Plans

1. Fuzz virtio-net, virtio-sock, virtio-*
2. Fuzz PCI
3. Enhance start corpus
4. Update Qtest
5. Verify test cases
Useful Links
Links

1. QEMU changes (afl-fuzz branch):
   https://github.com/yandex/qemu/tree/afl-fuzz

2. AFL changes (qemu-fuzz branch):
   https://github.com/yandex/AFL/tree/qemu-fuzz
Fuzz the Code for Fun and Profit!

Dima Stepanov
Software Engineer, Yandex.Cloud Hypervisor Team
dimastep@yandex-team.ru