MORPHUZZ: Bending (Input) Space to Fuzz Virtual Devices

Alexander Bulekov, *Boston University and Red Hat;* Bandan Das and Stefan Hajnoczi, *Red Hat;* Manuel Egele, *Boston University*

https://www.usenix.org/conference/usenixsecurity22/presentation/bulekov

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium is sponsored by USENIX.

A Artifact Appendix

A.1 Abstract

Our artifacts facilitate building and running Morphuzz for the QEMU and Bhyve hypervisors. These are the two implementations of Morphuzz described in our paper. We packaged the fuzzers in two VMs (one for fuzzing each hypervisor). Use an Intel/AMD x86-64 Linux machine to run these VMs.

A.2 Artifact check-list (meta-information)

- **Compilation:** clang (included)
- **Run-time environment:** Linux
- **Hardware:** Intel/AMD x86-64 Machine
- **Output:** Crashes
- **How much disk space required (approximately)?:** 20 GB
- **How much time is needed to prepare workflow (approximately)?:** 1-2 Hours
- **Publicly available?:** Yes
- **Code licenses (if publicly available)?:** GPLv2
- **Archived?: https://zenodo.org/record/5655839**

A.3 Description

A.3.1 How to access

Download the Artifacts (2 VM images) from: https://zenodo.org/record/5655839

A.3.2 Hardware dependencies

Intel or AMD x86-64 Machine with Virtualization support

A.3.3 Software dependencies

Linux, KVM, and QEMU 3.1+

A.4 QEMU Installation

Our instructions are based around VM images. The evaluation should be possible on a single bare-metal x86 machine running linux.

Please install QEMU to run these VMs:

- Debian: `apt install qemu-system-x86_64`
- Ubuntu: `apt install qemu-system-x86_64`
- Fedora: `yum install qemu-kvm`
- Other: https://www.qemu.org/download/

Additionally, to ensure that the VM takes advantage of hardware-acceleration

1. Check that you have virtualization extensions enabled in BIOS (VT-x on Intel, AMD-V/SEV on AMD).
2. Check that your user has access to `/dev/kvm`. Usually, you can add your user to the kvm group: `sudo adduser $USER kvm`. (Otherwise you may need to run qemu as root/with sudo).
3. To test out bhyve-fuzzing, you may need to enable support for nested-virtualization. Unlike the QEMU-Fuzzer, the bhyve-Fuzzer is not decoupled from the in-kernel CPU virtualization component. Thus, even though our fuzzers do not run any virtual CPU code, we will need to enable nested-virt to create the VM. This amounts to loading the KVM kernel module with a special flag. This page has instructions to do this for AMD and Intel CPUs: https://docs.fedoraproject.org/en-US/quick-docs/using-nested-virtualization-in-kvm/

Please open a terminal and "cd" into the folder containing the qcow2 VM images you downloaded. We will be building the fuzzers inside the VMs. Then, we will fuzz a virtual-device. We will observe any crashes found by the fuzzer. Finally, we will generate coverage-reports for the results.

In each VM, we provide annotated scripts that will build the fuzzer and run it for some example virtual-device.

Boot the QEMU Fuzzing VM:

```
$ qemu-system-x86_64 -machine q35 \
-accel kvm -cpu host -m 4G -smp 2 \
-hda ./morphuzz_qemu.qcow2 -vga virtio \
-device virtio-net,netdev=mynet0 \
-netdev \
user,id=mynet0,hostfwd=tcp:127.0.0.1:22222-:22
```

After a few seconds, you should be able to ssh into the VM from another terminal on your machine:

```
$ ssh -p22222 paper@localhost
Credentials:
    user: paper
    pass: artifact_eval
```

Once you are SSHed, we can proceed with building and running the fuzzer.

A.5 QEMU Evaluation

We provide annotated scripts for building, fuzzing, and providing readable qtest-reproducers:

```
$ cat build.sh        # Examine the build script...
$ ./build.sh          # Build QEMU with Morphuzz

$ cat run_example.sh  # Examine the example script
                      # for running the fuzzer
$ ./run_example.sh    # Fuzz a virtual device
# ctrl-c to stop fuzzing

$ cat reproducer.sh   # Examine the script
                      # to build a QEMU reproducer

$ ./reproduce.sh      # This will reproduce
                      # a megaraid bug

$ ./build_gcov.sh     # Build Morphuzz with
                      # GCov Support

$ ./run_gcov.sh       # Run the CORPUS collected by
                      # run_example.sh and output a
                      # coverage summary
```

Once you are ready to switch to the bhyve VM: `sudo shutdown`

A.6 bhyve Installation

Boot the bhyve Fuzzing VM:

```
$ qemu-system-x86_64 -machine q35 \
-accel kvm -cpu host -m 4G -smp 2 \
-hda ./morphuzz_bhyve.qcow2 -vga virtio \
-device virtio-net,netdev=mynet0 \
-netdev \
user,id=mynet0,hostfwd=tcp:127.0.0.1:22223-:22
```

After a few seconds, you should be able to ssh into the VM from another terminal on your machine:

```
$ ssh -p22223 paper@localhost
Credentials:
    user: paper
    pass: artifact_eval
```

A.7 bhyve Evaluation

```
$ cat build.sh          # Examine the build script...
$ sudo ./build.sh       # Build Bhyve with Morphuzz

$ cat run_example.sh    # Examine the example run script
$ sudo ./run_example.sh # Fuzz Bhyve configured with
                        # common virtual-devices
# ctrl-c to stop fuzzing

$ sudo ./reproduce_crashes.sh   # This will reproduce
                                # the crashes discovered
                                # by Morphuzz

$ sudo ./run_cov.sh     # This will output a fuzzing
                        # coverage report to /tmp/html

# Use scp to copy the coverage report to the local
# machine. View the report in a web browser.
```

Note that some of these commands require sudo. Once you are done, `sudo poweroff`

A.8 Experiment customization

The QEMU/Bhyve configurations can be customized by changing the environment variables specified in the `run_example.sh` script. These variables specify the virtual devices attached to QEMU/Bhyve.

A.9 Notes

These steps are specific to the artifacts provided in the VMs. An up-to-date version of QMorphuzz is maintained and documented at https://gitlab.com/qemu-project/qemu/[1]

Current upstream documentation for using QEMU's fuzzing infrastructure/QMorphuzz can be found at: https://gitlab.com/qemu-project/qemu/-/blob/c39deb218178d1fb814dd2138ceff4b541a03d85/docs/devel/fuzzing.rst

The main differences between the upstream version of Morphuzz, and the version described in this paper are:

- The upstream version of QMorphuzz performs PCI enumeration, prior to fuzzing, to improve fuzzing efficiency.
- The upstream version contains some device-specific fuzzers (independent of QMorphuzz), which serve mostly as examples to go along with documentation. These are removed in the artifact.
- The upstream version of QMorphuzz provides a sparse memory device which improves the efficiency of the fuzzing process.
- The upstream version of QMorphuzz includes the configurations used to fuzz QEMU on OSS-Fuzz.
- The version of QEMU in the artifacts is 5.0.0. At this time, QEMU 6.2.0 has been released. Many of the bugs that can be found by Morphuzz in the artifact VM have already been patched.
- The upstream version comes with documentation for fuzzing additional devices, and adding custom QEMU fuzzers.

[1] Stable link to the version of QEMU at the time of this writing: https://gitlab.com/qemu-project/qemu/-/tree/c39deb218178d1fb814dd2138ceff4b541a03d85