setup_ubuntu-host_qemu-vm_x86-64-kernel

Setup: Ubuntu host, QEMU vm, x86-64 kernel

These are the instructions on how to fuzz the x86-64 kernel in a QEMU with Ubuntu on the host machine and Debian Stretch in the QEMU instances.

GCC

While you may use GCC that is available from your distro, it's preferable to get the lastest one from this list. Download and unpack into $GCC , and you should have GCC binaries in $GCC/bin/

```
$ ls $GCC/bin/
cpp     gcc-ranlib  x86_64-pc-linux-gnu-gcc       x86_64-pc-linux-gnu-gcc-ranlib
gcc     gcov        x86_64-pc-linux-gnu-gcc-9.0.0
gcc-ar  gcov-dump   x86_64-pc-linux-gnu-gcc-ar
gcc-nm  gcov-tool   x86_64-pc-linux-gnu-gcc-nm
```

Kernel

Checkout Linux kernel source:

```
git clone https://github.com/torvalds/linux.git $KERNEL
```

Generate default configs:

```
cd $KERNEL
make CC="$GCC/bin/gcc" defconfig
make CC="$GCC/bin/gcc" kvmconfig
```

Now we need to enable some config options required for syzkaller.
Edit .config file manually and enable:

```
CONFIG_KCOV=y
CONFIG_DEBUG_INFO=y
CONFIG_KASAN=y
CONFIG_KASAN_INLINE=y
```

You may also need the following for a recent linux image:

```
CONFIG_CONFIGFS_FS=y
CONFIG_SECURITYFS=y
```

You might also want to enable some other kernel configs as described here.

Since enabling these options results in more sub options being available, we need to regenerate config:

```
make CC="$GCC/bin/gcc" olddefconfig
```

Build the kernel:

```
make CC="$GCC/bin/gcc" -j64
```

Now you should have vmlinux (kernel binary) and bzImage (packed kernel image):

```
$ ls $KERNEL/vmlinux
$KERNEL/vmlinux
$ ls $KERNEL/arch/x86/boot/bzImage
```

```
$KERNEL/arch/x86/boot/bzImage
```

Image

Install debootstrap:

```
sudo apt-get install debootstrap
```

To create a Debian Stretch Linux image with the minimal set of required packages do:

```
cd $IMAGE/
wget https://raw.githubusercontent.com/google/syzkaller/master/tools/create-image.sh -O create-image.sh
chmod +x create-image.sh
./create-image.sh
```

The result should be `$IMAGE/stretch.img` disk image.

If you would like to generate an image with Debian Wheezy, instead of Stretch, do:

```
./create-image.sh --distribution wheezy
```

Sometimes it's useful to have some additional packages and tools available in the VM even though they are not required to run syzkaller. To install a set of tools we find useful do (feel free to edit the list of tools in the script):

```
./create-image.sh --feature full
```

To install perf (not required to run syzkaller; requires `$KERNEL` to point to the kernel sources):

```
./create-image.sh --add-perf
```

For additional options of create-image.sh , please refer to ./create-image.sh -h

QEMU

Install QEMU :

```
sudo apt-get install qemu-system-x86
```

Make sure the kernel boots and sshd starts:

```
qemu-system-x86_64 \
  -kernel $KERNEL/arch/x86/boot/bzImage \
  -append "console=ttyS0 root=/dev/sda earlyprintk=serial"\
  -hda $IMAGE/stretch.img \
  -net user,hostfwd=tcp::10021-:22 -net nic \
  -enable-kvm \
  -nographic \
  -m 2G \
  -smp 2 \
  -pidfile vm.pid \
  2>&1 | tee vm.log
```

```
early console in setup code
early console in extract_kernel
input_data: 0x0000000005d9e276
input_len: 0x0000000001da5af3
output: 0x0000000001000000
output_len: 0x00000000058799f8
kernel_total_size: 0x0000000006b63000

Decompressing Linux... Parsing ELF... done.
Booting the kernel.
[    0.000000] Linux version 4.12.0-rc3+ ...
```

```
[    0.000000] Command line: console=ttyS0 root=/dev/sda debug earlyprintk=serial
...
[ ok ] Starting enhanced syslogd: rsyslogd.
[ ok ] Starting periodic command scheduler: cron.
[ ok ] Starting OpenBSD Secure Shell server: sshd.
```

After that you should be able to ssh to QEMU instance in another terminal:

```
ssh -i $IMAGE/stretch.id_rsa -p 10021 -o "StrictHostKeyChecking no" root@localhost
```

If this fails with "too many tries", ssh may be passing default keys before the one explicitly passed with `-i` . Append option `-o "IdentitiesOnly yes"` .

To kill the running QEMU instance:

```
kill $(cat vm.pid)
```

syzkaller

Build syzkaller as described here. Then create a manager config like the following, replacing the environment variables `$GOPATH` , `$KERNEL` and `$IMAGE` with their actual values.

```
{
    "target": "linux/amd64",
    "http": "127.0.0.1:56741",
    "workdir": "$GOPATH/src/github.com/google/syzkaller/workdir",
    "kernel_obj": "$KERNEL",
    "image": "$IMAGE/stretch.img",
    "sshkey": "$IMAGE/stretch.id_rsa",
    "syzkaller": "$GOPATH/src/github.com/google/syzkaller",
    "procs": 8,
    "type": "qemu",
```

```
    "vm": {
        "count": 4,
        "kernel": "$KERNEL/arch/x86/boot/bzImage",
        "cpu": 2,
        "mem": 2048
    }
}
```

Run syzkaller manager:

```
mkdir workdir
./bin/syz-manager -config=my.cfg
```

Now syzkaller should be running, you can check manager status with your web browser at 127.0.0.1:56741 .

If you get issues after syz-manager starts, consider running it with the -debug flag.
Also see this page for troubleshooting tips.