syzkaller

目标

使用syzkaller对openeuler 23.03进行测试

环境

操作系统	： Ubuntu 22.04
内核版本	： 6.2.2-060202-generic
QEMU版本	： 7.2.0
GCC版本	： 11.3.0
git版本	： 2.34.1

步骤

编译内核

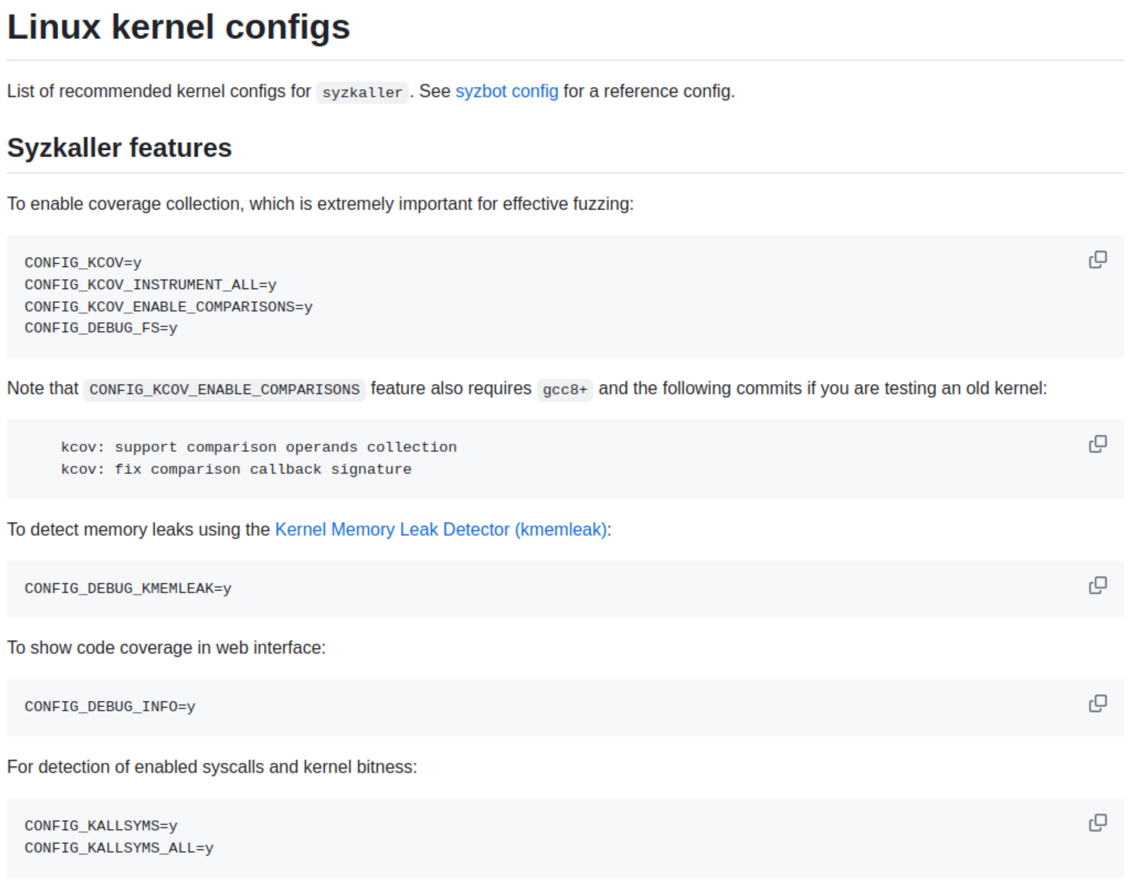
下载openeuler 23.03源码

```
git clone --depth=1 --branch v6.1.19 https://gitee.com/openeuler/kernel.git
```

配置kernel编译选项

由于syzkaller需要KCOV支持，所以需要在menuconfig中开启

主要根据syzkaller的推荐选项来配置，开启了KALLSYM，KASAN，KMSAN，DEBUGFS等选项



Linux kernel configs

List of recommended kernel configs for `syzkaller`. See syzbot config for a reference config.

Syzkaller features

To enable coverage collection, which is extremely important for effective fuzzing:

```
CONFIG_KCOV=y
CONFIG_KCOV_INSTRUMENT_ALL=y
CONFIG_KCOV_ENABLE_COMPARISONS=y
CONFIG_DEBUG_FS=y
```

Note that `CONFIG_KCOV_ENABLE_COMPARISONS` feature also requires `gcc8+` and the following commits if you are testing an old kernel:

```
    kcov: support comparison operands collection
    kcov: fix comparison callback signature
```

To detect memory leaks using the Kernel Memory Leak Detector (kmemleak):

```
CONFIG_DEBUG_KMEMLEAK=y
```

To show code coverage in web interface:

```
CONFIG_DEBUG_INFO=y
```

For detection of enabled syscalls and kernel bitness:

```
CONFIG_KALLSYMS=y
CONFIG_KALLSYMS_ALL=y
```

下载riscv-toolchain

```
git clone --depth=1 https://github.com/riscv/riscv-gnu-toolchain
```

安装依赖

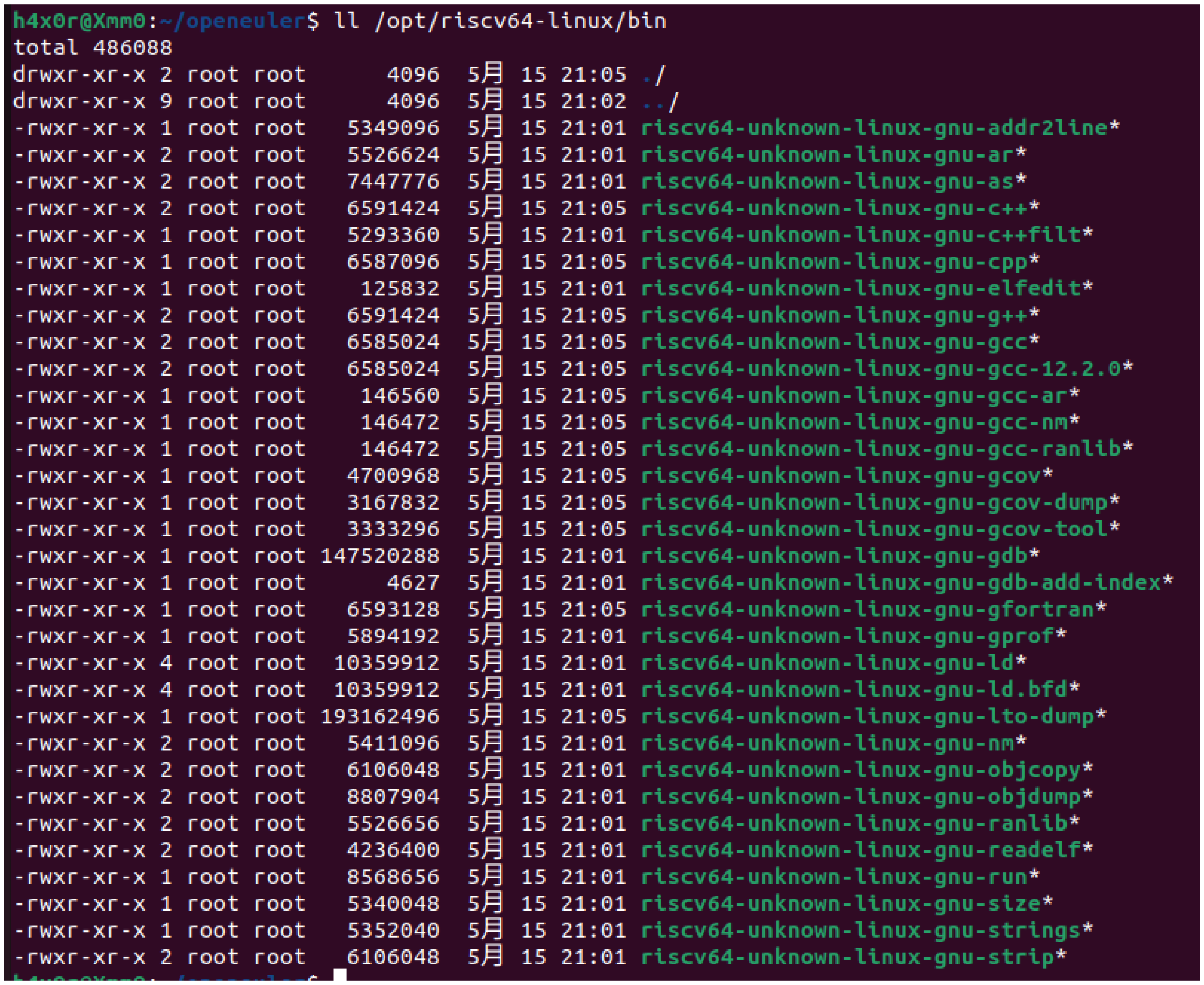
```
sudo apt-get install autoconf automake autotools-dev curl python3 libmpc-dev
libmpfr-dev libgmp-dev gawk build-essential bison flex texinfo gperf libtool
patchutils bc zlib1g-dev libexpat-dev ninja-build
```

make

```
sudo mkdir -p /opt/riscv64-linux/
./configure --prefix=/opt/riscv64-linux
sudo make -j `nproc`
```

设置环境变量

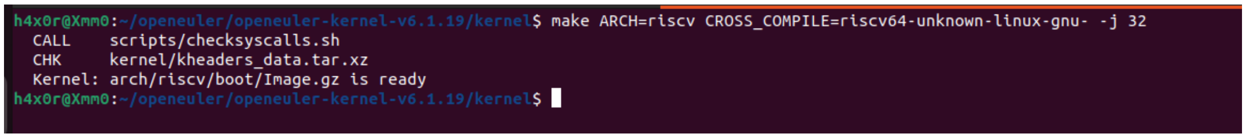
```
echo 'export PATH=$PATH:/opt/riscv64-linux/bin/' >> ~/.bashrc
```



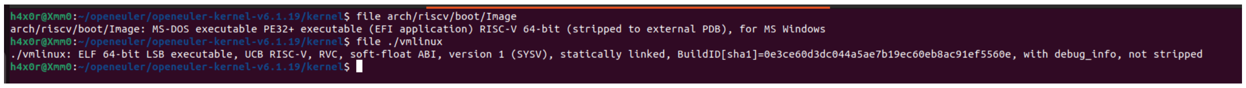
```
h4x0r@Xmm0:~/openeuler$ ll /opt/riscv64-linux/bin
total 486088
drwxr-xr-x 2 root root      4096 5月 15 21:05 ./
drwxr-xr-x 9 root root      4096 5月 15 21:02 ../
-rwxr-xr-x 1 root root   5349096 5月 15 21:01 riscv64-unknown-linux-gnu-addr2line*
-rwxr-xr-x 2 root root   5526624 5月 15 21:01 riscv64-unknown-linux-gnu-ar*
-rwxr-xr-x 2 root root   7447776 5月 15 21:01 riscv64-unknown-linux-gnu-as*
-rwxr-xr-x 2 root root   6591424 5月 15 21:05 riscv64-unknown-linux-gnu-c++*
-rwxr-xr-x 1 root root   5293360 5月 15 21:01 riscv64-unknown-linux-gnu-c++filt*
-rwxr-xr-x 1 root root   6587096 5月 15 21:05 riscv64-unknown-linux-gnu-cpp*
-rwxr-xr-x 1 root root    125832 5月 15 21:01 riscv64-unknown-linux-gnu-elfedit*
-rwxr-xr-x 2 root root   6591424 5月 15 21:05 riscv64-unknown-linux-gnu-g++*
-rwxr-xr-x 2 root root   6585024 5月 15 21:05 riscv64-unknown-linux-gnu-gcc*
-rwxr-xr-x 2 root root   6585024 5月 15 21:05 riscv64-unknown-linux-gnu-gcc-12.2.0*
-rwxr-xr-x 1 root root    146560 5月 15 21:05 riscv64-unknown-linux-gnu-gcc-ar*
-rwxr-xr-x 1 root root    146472 5月 15 21:05 riscv64-unknown-linux-gnu-gcc-nm*
-rwxr-xr-x 1 root root    146472 5月 15 21:05 riscv64-unknown-linux-gnu-gcc-ranlib*
-rwxr-xr-x 1 root root   4700968 5月 15 21:05 riscv64-unknown-linux-gnu-gcov*
-rwxr-xr-x 1 root root   3167832 5月 15 21:05 riscv64-unknown-linux-gnu-gcov-dump*
-rwxr-xr-x 1 root root   3333296 5月 15 21:05 riscv64-unknown-linux-gnu-gcov-tool*
-rwxr-xr-x 1 root root 147520288 5月 15 21:01 riscv64-unknown-linux-gnu-gdb*
-rwxr-xr-x 1 root root      4627 5月 15 21:01 riscv64-unknown-linux-gnu-gdb-add-index*
-rwxr-xr-x 1 root root   6593128 5月 15 21:05 riscv64-unknown-linux-gnu-gfortran*
-rwxr-xr-x 1 root root   5894192 5月 15 21:01 riscv64-unknown-linux-gnu-gprof*
-rwxr-xr-x 4 root root  10359912 5月 15 21:01 riscv64-unknown-linux-gnu-ld*
-rwxr-xr-x 4 root root  10359912 5月 15 21:01 riscv64-unknown-linux-gnu-ld.bfd*
-rwxr-xr-x 1 root root 193162496 5月 15 21:05 riscv64-unknown-linux-gnu-lto-dump*
-rwxr-xr-x 2 root root   5411096 5月 15 21:01 riscv64-unknown-linux-gnu-nm*
-rwxr-xr-x 2 root root   6106048 5月 15 21:01 riscv64-unknown-linux-gnu-objcopy*
-rwxr-xr-x 2 root root   8807904 5月 15 21:01 riscv64-unknown-linux-gnu-objdump*
-rwxr-xr-x 2 root root   5526656 5月 15 21:01 riscv64-unknown-linux-gnu-ranlib*
-rwxr-xr-x 2 root root   4236400 5月 15 21:01 riscv64-unknown-linux-gnu-readelf*
-rwxr-xr-x 1 root root   8568656 5月 15 21:01 riscv64-unknown-linux-gnu-run*
-rwxr-xr-x 1 root root   5340048 5月 15 21:01 riscv64-unknown-linux-gnu-size*
-rwxr-xr-x 1 root root   5352040 5月 15 21:01 riscv64-unknown-linux-gnu-strings*
-rwxr-xr-x 2 root root   6106048 5月 15 21:01 riscv64-unknown-linux-gnu-strip*
```

编译内核

```
make ARCH=riscv CROSS_COMPILE=riscv64-unknown-linux-gnu- -j 32
```



```
h4x0r@Xmm0:~/openeuler/openeuler-kernel-v6.1.19/kernel$ make ARCH=riscv CROSS_COMPILE=riscv64-unknown-linux-gnu- -j 32
  CALL    scripts/checksyscalls.sh
  CHK     kernel/kheaders_data.tar.xz
  Kernel: arch/riscv/boot/Image.gz is ready
h4x0r@Xmm0:~/openeuler/openeuler-kernel-v6.1.19/kernel$ 
```

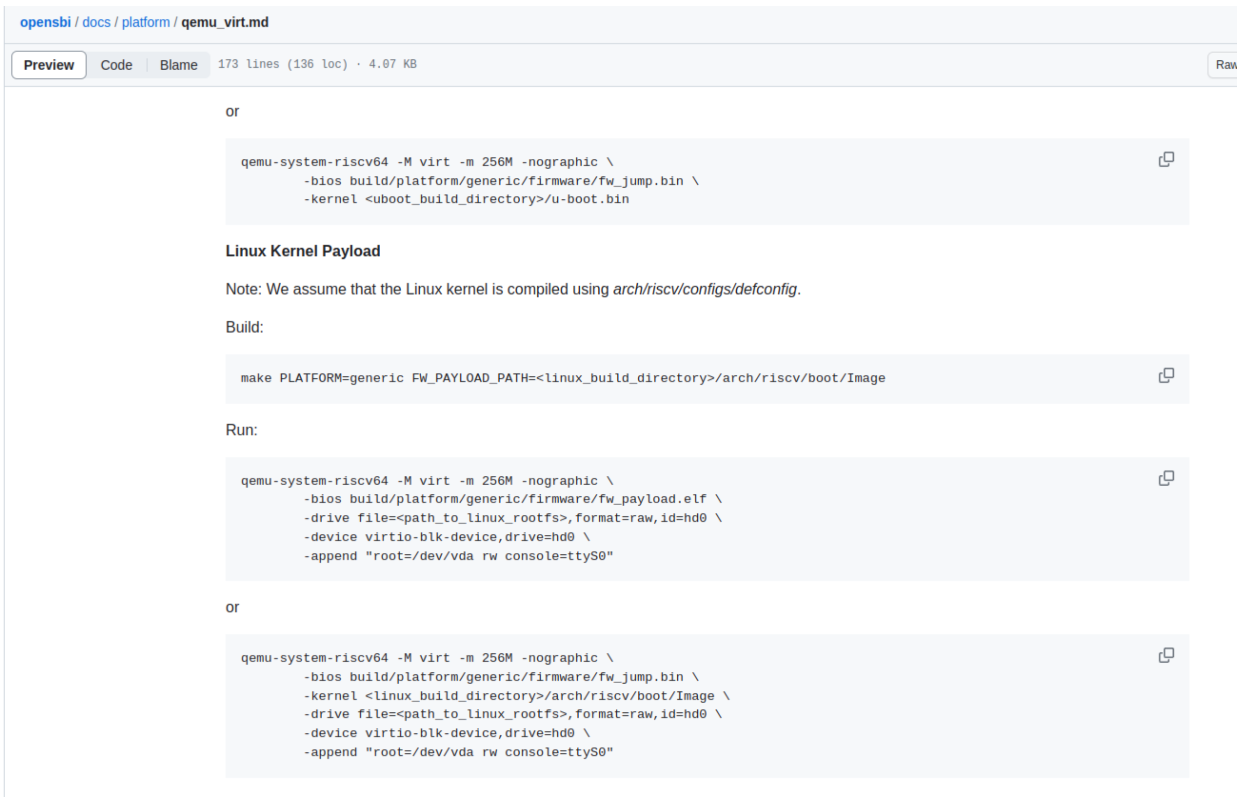


```
h4x0r@Xmm0:~/openeuler/openeuler-kernel-v6.1.19/kernel$ file arch/riscv/boot/Image
arch/riscv/boot/Image: MS-DOS executable PE32+ executable (EFI application) RISC-V 64-bit (stripped to external PDB), for MS Windows
h4x0r@Xmm0:~/openeuler/openeuler-kernel-v6.1.19/kernel$ file ./vmlinux
./vmlinux: ELF 64-bit LSB executable, UCB RISC-V, RVC, soft-float ABI, version 1 (SYSV), statically linked, BuildID[sha1]=0e3ce60d3dc044a5ae7b19ec60eb8ac91ef5560e, with debug_info, not stripped
h4x0r@Xmm0:~/openeuler/openeuler-kernel-v6.1.19/kernel$ 
```

下载并编译OpenSBI

```
git clone --depth=1 https://github.com/riscv-software-src/opensbi.git
```

按照此文档的说明，将linux内核作为payload进行编译



opensbi / docs / platform / qemu_virt.md

Preview | Code | Blame 173 lines (136 loc) · 4.07 KB

or

```
qemu-system-riscv64 -M virt -m 256M -nographic \
        -bios build/platform/generic/firmware/fw_jump.bin \
        -kernel <uboot_build_directory>/u-boot.bin
```

Linux Kernel Payload

Note: We assume that the Linux kernel is compiled using *arch/riscv/configs/defconfig*.

Build:

```
make PLATFORM=generic FW_PAYLOAD_PATH=<linux_build_directory>/arch/riscv/boot/Image
```

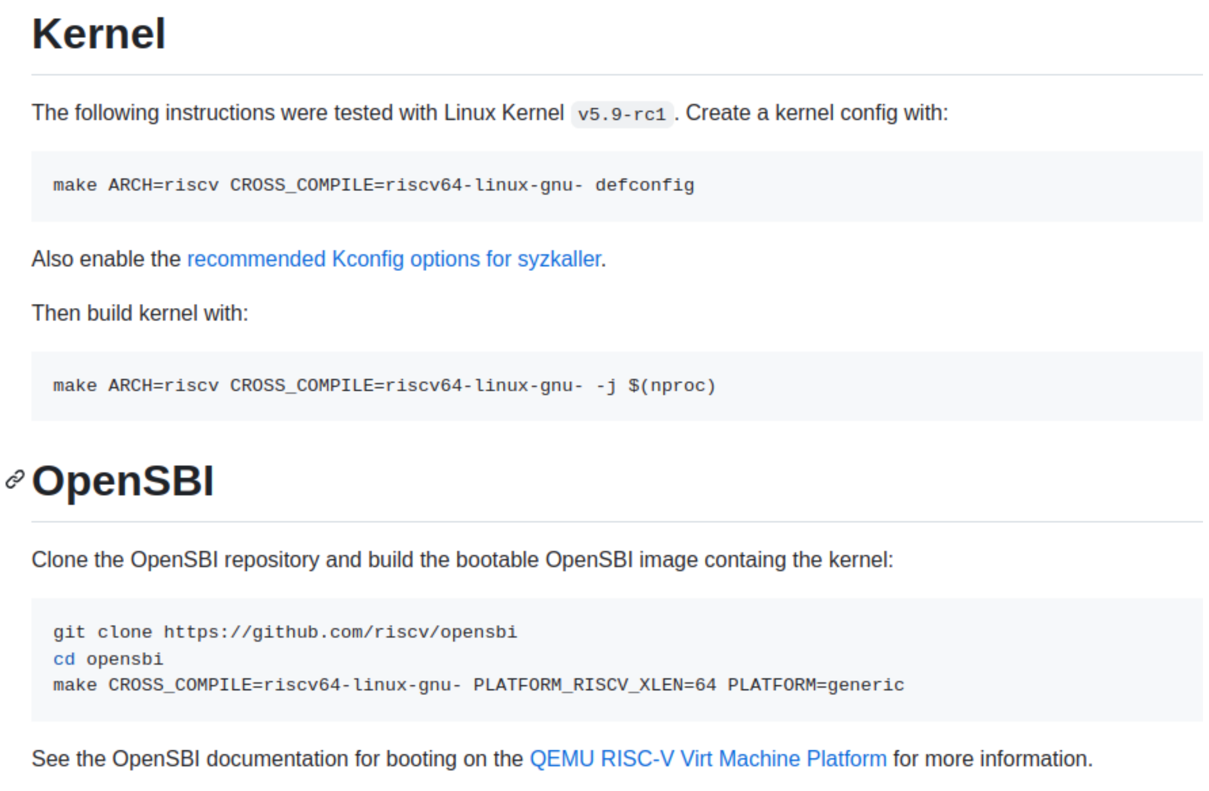
Run:

```
qemu-system-riscv64 -M virt -m 256M -nographic \
        -bios build/platform/generic/firmware/fw_payload.elf \
        -drive file=<path_to_linux_rootfs>,format=raw,id=hd0 \
        -device virtio-blk-device,drive=hd0 \
        -append "root=/dev/vda rw console=ttyS0"
```

or

```
qemu-system-riscv64 -M virt -m 256M -nographic \
        -bios build/platform/generic/firmware/fw_jump.bin \
        -kernel <linux_build_directory>/arch/riscv/boot/Image \
        -drive file=<path_to_linux_rootfs>,format=raw,id=hd0 \
        -device virtio-blk-device,drive=hd0 \
        -append "root=/dev/vda rw console=ttyS0"
```

同时参考syzkaller的文档添加了CROSS_COMPILE环境变量



Kernel

The following instructions were tested with Linux Kernel `v5.9-rc1`. Create a kernel config with:

```
make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu- defconfig
```

Also enable the recommended Kconfig options for syzkaller.

Then build kernel with:

```
make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu- -j $(nproc)
```

OpenSBI

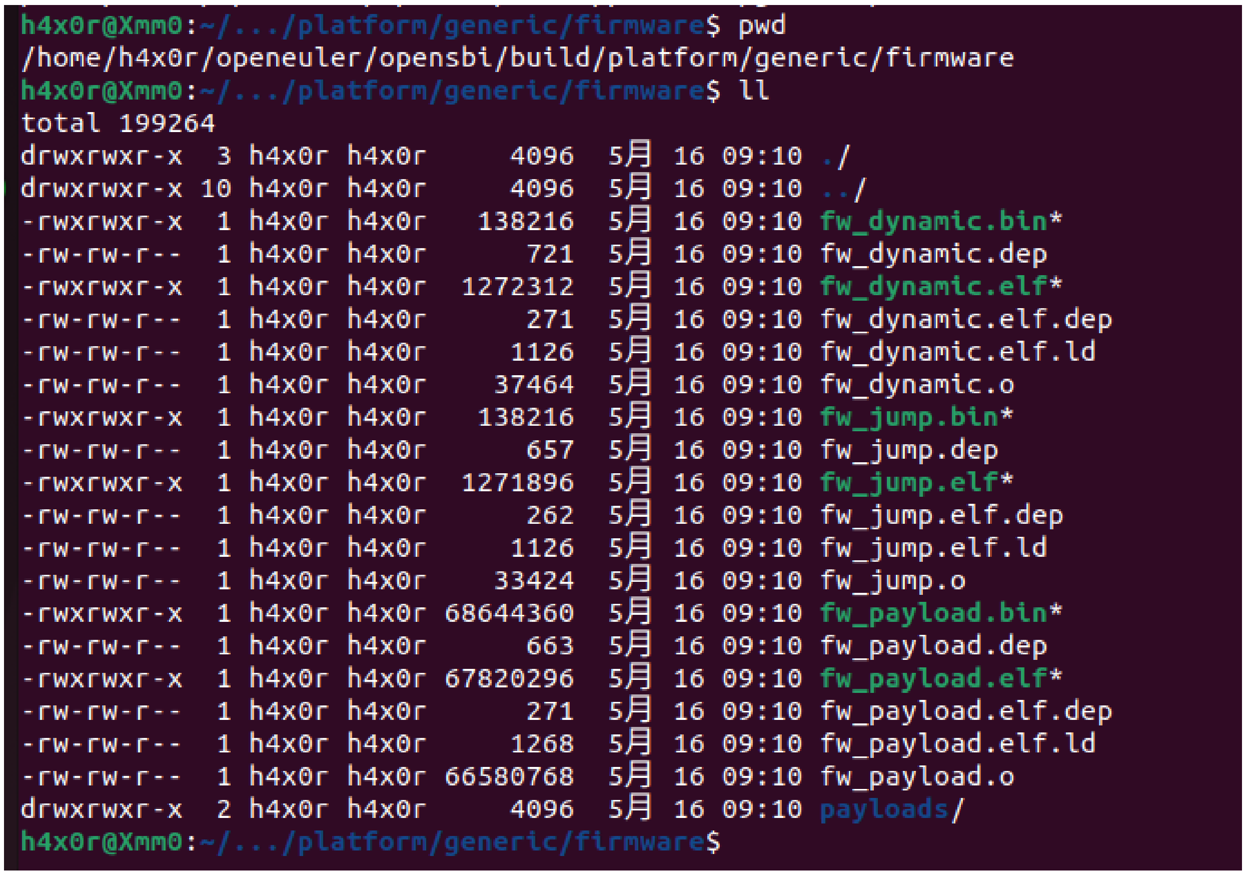
Clone the OpenSBI repository and build the bootable OpenSBI image containg the kernel:

```
git clone https://github.com/riscv/opensbi
cd opensbi
make CROSS_COMPILE=riscv64-linux-gnu- PLATFORM_RISCV_XLEN=64 PLATFORM=generic
```

See the OpenSBI documentation for booting on the QEMU RISC-V Virt Machine Platform for more information.

编译命令

```
    make CROSS_COMPILE=riscv64-unknown-linux-gnu- PLATFORM_RISCV_XLEN=64
PLATFORM=generic FW_PAYLOAD_PATH=/home/h4x0r/openeuler/openeuler-kernel-
v6.1.19/kernel/arch/riscv/boot/Image -j 32
```

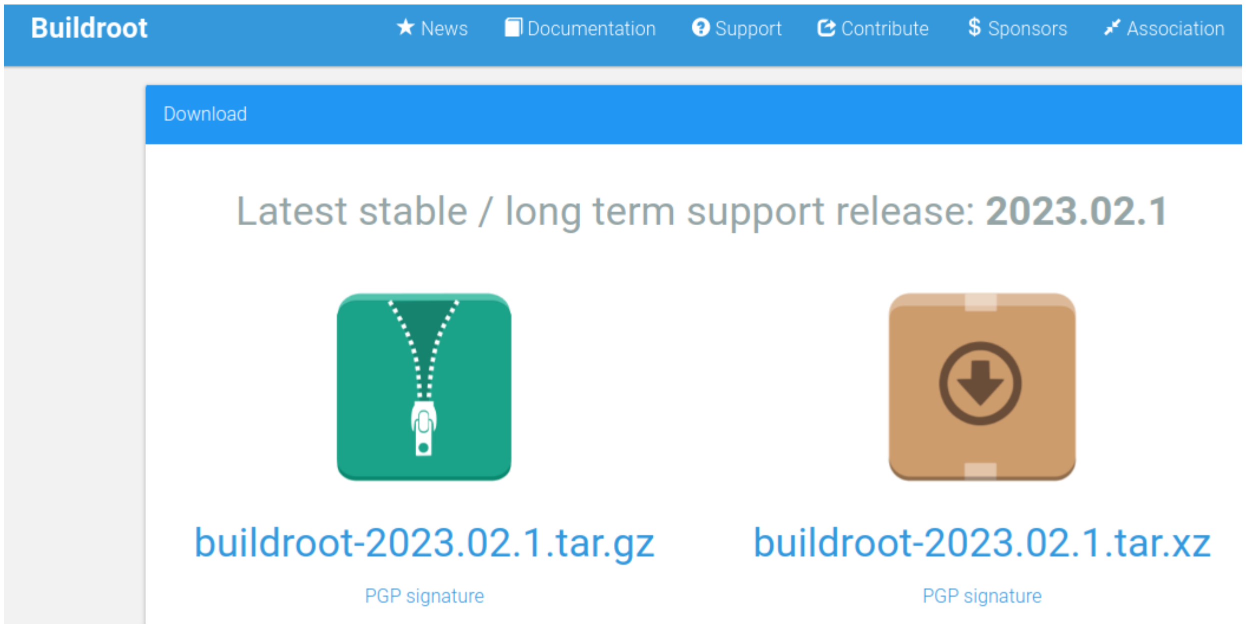


```
h4x0r@Xmm0:~/.../platform/generic/firmware$ pwd
/home/h4x0r/openeuler/opensbi/build/platform/generic/firmware
h4x0r@Xmm0:~/.../platform/generic/firmware$ ll
total 199264
drwxrwxr-x  3 h4x0r h4x0r     4096  5月 16 09:10 ./
drwxrwxr-x 10 h4x0r h4x0r     4096  5月 16 09:10 ../
-rwxrwxr-x  1 h4x0r h4x0r   138216  5月 16 09:10 fw_dynamic.bin*
-rw-rw-r--  1 h4x0r h4x0r      721  5月 16 09:10 fw_dynamic.dep
-rwxrwxr-x  1 h4x0r h4x0r  1272312  5月 16 09:10 fw_dynamic.elf*
-rw-rw-r--  1 h4x0r h4x0r      271  5月 16 09:10 fw_dynamic.elf.dep
-rw-rw-r--  1 h4x0r h4x0r     1126  5月 16 09:10 fw_dynamic.elf.ld
-rw-rw-r--  1 h4x0r h4x0r    37464  5月 16 09:10 fw_dynamic.o
-rwxrwxr-x  1 h4x0r h4x0r   138216  5月 16 09:10 fw_jump.bin*
-rw-rw-r--  1 h4x0r h4x0r      657  5月 16 09:10 fw_jump.dep
-rwxrwxr-x  1 h4x0r h4x0r  1271896  5月 16 09:10 fw_jump.elf*
-rw-rw-r--  1 h4x0r h4x0r      262  5月 16 09:10 fw_jump.elf.dep
-rw-rw-r--  1 h4x0r h4x0r     1126  5月 16 09:10 fw_jump.elf.ld
-rw-rw-r--  1 h4x0r h4x0r    33424  5月 16 09:10 fw_jump.o
-rwxrwxr-x  1 h4x0r h4x0r 68644360  5月 16 09:10 fw_payload.bin*
-rw-rw-r--  1 h4x0r h4x0r      663  5月 16 09:10 fw_payload.dep
-rwxrwxr-x  1 h4x0r h4x0r 67820296  5月 16 09:10 fw_payload.elf*
-rw-rw-r--  1 h4x0r h4x0r      271  5月 16 09:10 fw_payload.elf.dep
-rw-rw-r--  1 h4x0r h4x0r     1268  5月 16 09:10 fw_payload.elf.ld
-rw-rw-r--  1 h4x0r h4x0r 66580768  5月 16 09:10 fw_payload.o
drwxrwxr-x  2 h4x0r h4x0r     4096  5月 16 09:10 payloads/
h4x0r@Xmm0:~/.../platform/generic/firmware$
```

创建rootfs

下载buildroot

从官网上找到最新的LTS版本压缩包



Latest stable / long term support release: **2023.02.1**

buildroot-2023.02.1.tar.gz
PGP signature

buildroot-2023.02.1.tar.xz
PGP signature

下载解压

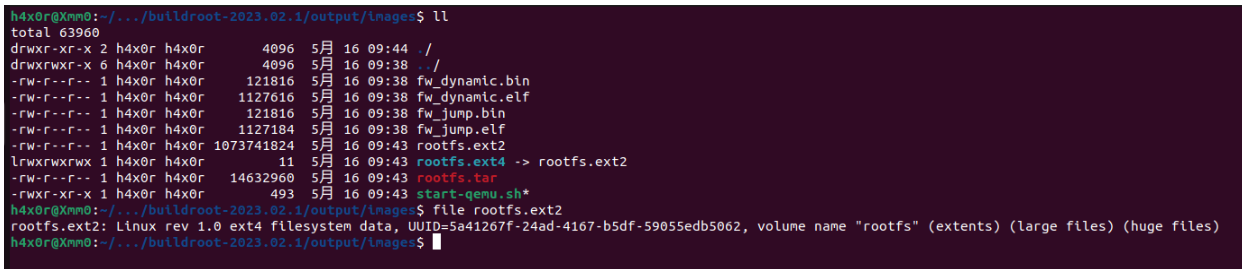
编译

```
make qemu_riscv64_virt_defconfig
make menuconfig
```

由于syzkaller需要被测试的vm通过ssh进行文件传输，所以需要根据文档对buildroot的选项进行一些修改

> Choose the following options:
>
> ```
>     Target packages
>             Networking applications
>                 [*] iproute2
>                 [*] openssh
>     Filesystem images
>                 ext2/3/4 variant - ext4
>                 exact size - 1g
> ```
>
> Unselect:
>
> ```
>     Kernel
>             Linux Kernel
> ```
>
> Run `make`.
>
> Then add the following line to `output/target/etc/fstab`:
>
> ```
> debugfs /sys/kernel/debug       debugfs defaults        0       0
> ```
>
> Then replace `output/target/etc/ssh/sshd_config` with the following contents:
>
> ```
> PermitRootLogin yes
> PasswordAuthentication yes
> PermitEmptyPasswords yes
> ```

完成后make进行打包



```
h4x0r@Xmm0:~/.../buildroot-2023.02.1/output/images$ ll
total 63960
drwxr-xr-x 2 h4x0r h4x0r       4096  5月 16 09:44 ./
drwxrwxr-x 6 h4x0r h4x0r       4096  5月 16 09:38 ../
-rw-r--r-- 1 h4x0r h4x0r     121816  5月 16 09:38 fw_dynamic.bin
-rw-r--r-- 1 h4x0r h4x0r    1127616  5月 16 09:38 fw_dynamic.elf
-rw-r--r-- 1 h4x0r h4x0r     121816  5月 16 09:38 fw_jump.bin
-rw-r--r-- 1 h4x0r h4x0r    1127184  5月 16 09:38 fw_jump.elf
-rw-r--r-- 1 h4x0r h4x0r 1073741824  5月 16 09:43 rootfs.ext2
lrwxrwxrwx 1 h4x0r h4x0r         11  5月 16 09:43 rootfs.ext4 -> rootfs.ext2
-rw-r--r-- 1 h4x0r h4x0r   14632960  5月 16 09:43 rootfs.tar
-rwxr-xr-x 1 h4x0r h4x0r        493  5月 16 09:43 start-qemu.sh*
h4x0r@Xmm0:~/.../buildroot-2023.02.1/output/images$ file rootfs.ext2
rootfs.ext2: Linux rev 1.0 ext4 filesystem data, UUID=5a41267f-24ad-4167-b5df-59055edb5062, volume name "rootfs" (extents) (large files) (huge files)
h4x0r@Xmm0:~/.../buildroot-2023.02.1/output/images$ 
```

启动运行

根据opensbi文档的运行步骤来修改openeuler官网给出的start_vm.sh

Run:

```
qemu-system-riscv64 -M virt -m 256M -nographic \
        -bios build/platform/generic/firmware/fw_payload.elf \
        -drive file=<path_to_linux_rootfs>,format=raw,id=hd0 \
        -device virtio-blk-device,drive=hd0 \
        -append "root=/dev/vda rw console=ttyS0"
```

or

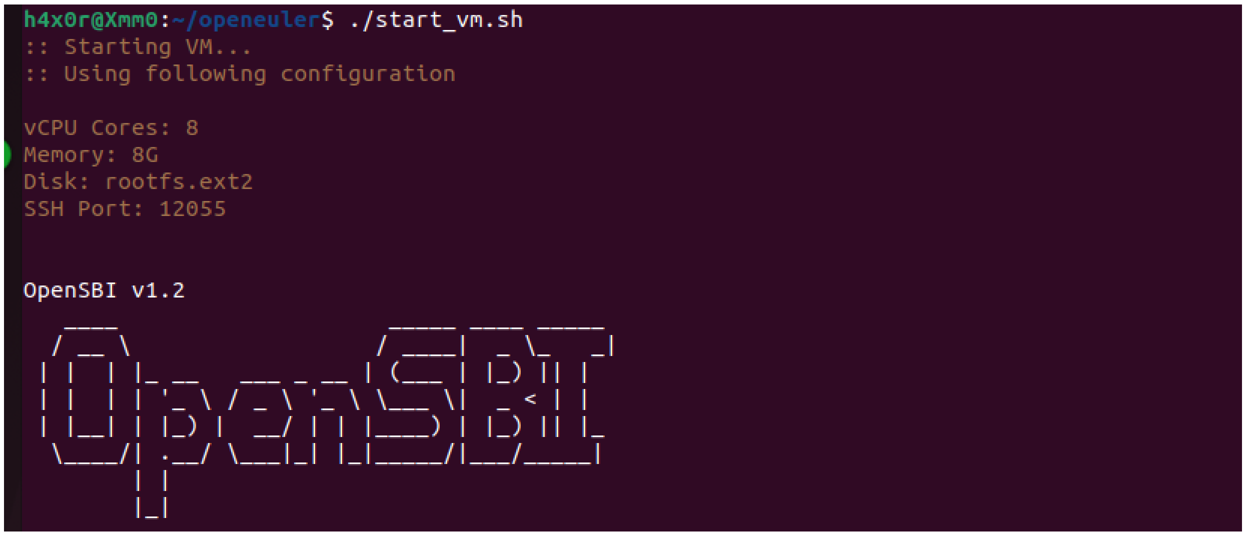
```
qemu-system-riscv64 -M virt -m 256M -nographic \
        -bios build/platform/generic/firmware/fw_jump.bin \
        -kernel <linux_build_directory>/arch/riscv/boot/Image \
        -drive file=<path_to_linux_rootfs>,format=raw,id=hd0 \
        -device virtio-blk-device,drive=hd0 \
        -append "root=/dev/vda rw console=ttyS0"
```

```bash
#!/usr/bin/env bash

RESTORE=$(echo -en '\001\033[0m\002')
YELLOW=$(echo -en '\001\033[00;33m\002')

## Configuration
vcpu=8
memory=8
memory_append=`expr $memory \* 1024`
drive="$(ls *.ext2)"
fw="fw_jump.bin"
ssh_port=12055

cmd="/usr/local/bin/qemu-riscv64/bin/qemu-system-riscv64 \
  -nographic -machine virt \
  -smp "$vcpu" -m "$memory"G \
  -bios "$fw" \
  -kernel ./Image \
  -drive file="$drive",format=raw,id=hd0 \
  -object rng-random,filename=/dev/urandom,id=rng0 \
  -device virtio-rng-device,rng=rng0 \
  -device virtio-blk-device,drive=hd0 \
  -device virtio-net-device,netdev=usernet \
  -append 'root=/dev/vda rw console=ttyS0' \
  -netdev user,id=usernet,hostfwd=tcp::"$ssh_port"-:22 \
  -device qemu-xhci -usb -device usb-kbd -device usb-tablet"

echo ${YELLOW}:: Starting VM...${RESTORE}
echo ${YELLOW}:: Using following configuration${RESTORE}
echo ""
echo ${YELLOW}vCPU Cores: "$vcpu"${RESTORE}
echo ${YELLOW}Memory: "$memory"G${RESTORE}
echo ${YELLOW}Disk: "$drive"${RESTORE}
echo ${YELLOW}SSH Port: "$ssh_port"${RESTORE}
echo ""

sleep 1
```

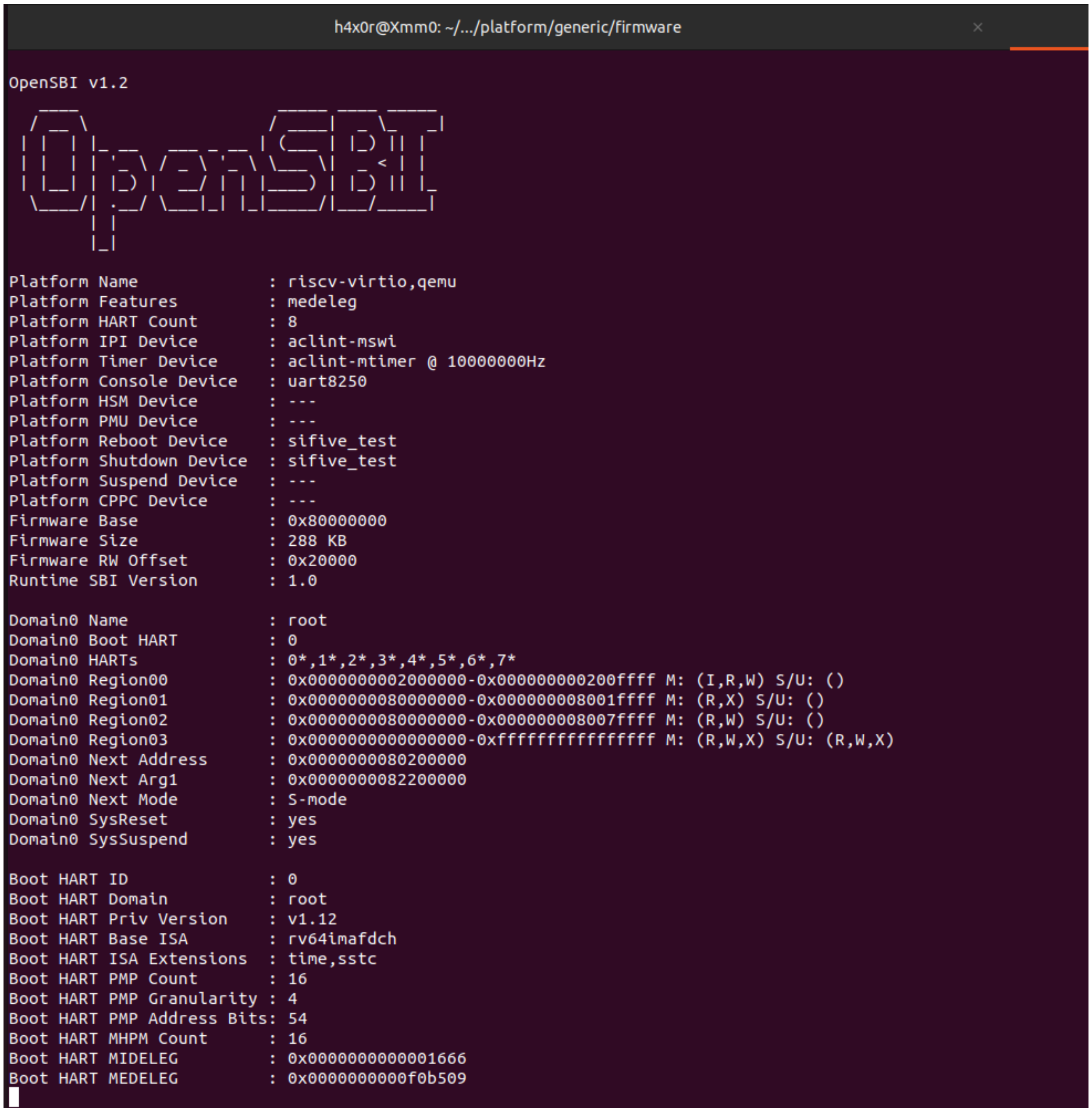
```
eval $cmd
```

将opensbi中编译得到的fw_jump.bin和buildroot得到的rootfs.ext2以及编译出来的内核镜像复制过来

运行脚本



```
h4x0r@Xmm0:~/openeuler$ ./start_vm.sh
:: Starting VM...
:: Using following configuration

vCPU Cores: 8
Memory: 8G
Disk: rootfs.ext2
SSH Port: 12055


OpenSBI v1.2
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|
```

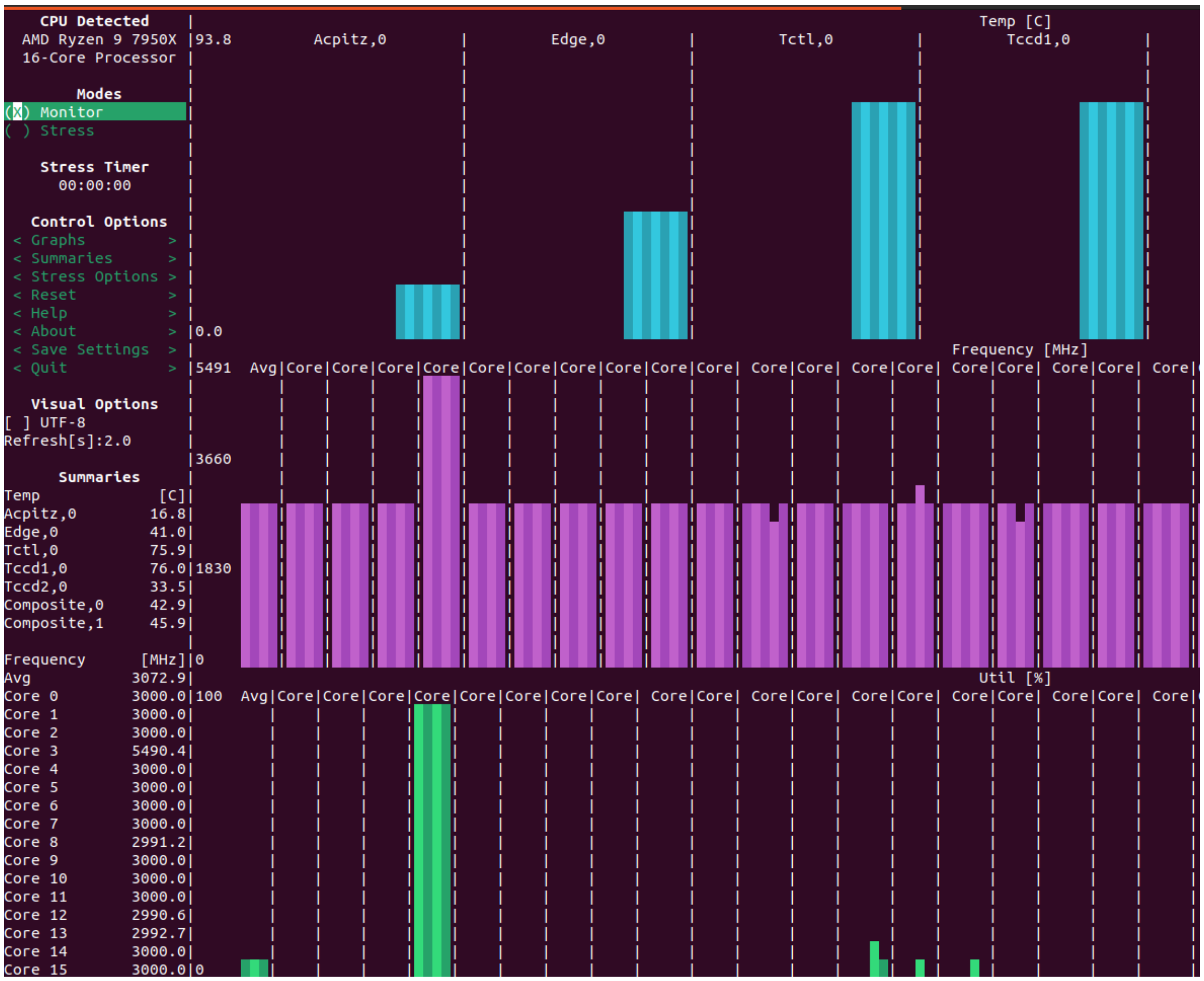


h4x0r@Xmm0: ~/.../platform/generic/firmware

```
OpenSBI v1.2
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

Platform Name             : riscv-virtio,qemu
Platform Features         : medeleg
Platform HART Count       : 8
Platform IPI Device       : aclint-mswi
Platform Timer Device     : aclint-mtimer @ 10000000Hz
Platform Console Device   : uart8250
Platform HSM Device       : ---
Platform PMU Device       : ---
Platform Reboot Device    : sifive_test
Platform Shutdown Device  : sifive_test
Platform Suspend Device   : ---
Platform CPPC Device      : ---
Firmware Base             : 0x80000000
Firmware Size             : 288 KB
Firmware RW Offset        : 0x20000
Runtime SBI Version       : 1.0

Domain0 Name              : root
Domain0 Boot HART         : 0
Domain0 HARTs             : 0*,1*,2*,3*,4*,5*,6*,7*
Domain0 Region00          : 0x0000000002000000-0x000000000200ffff M: (I,R,W) S/U: ()
Domain0 Region01          : 0x0000000080000000-0x000000008001ffff M: (R,X) S/U: ()
Domain0 Region02          : 0x0000000080000000-0x000000008007ffff M: (R,W) S/U: ()
Domain0 Region03          : 0x0000000000000000-0xffffffffffffffff M: (R,W,X) S/U: (R,W,X)
Domain0 Next Address      : 0x0000000080200000
Domain0 Next Arg1         : 0x0000000082200000
Domain0 Next Mode         : S-mode
Domain0 SysReset          : yes
Domain0 SysSuspend        : yes

Boot HART ID              : 0
Boot HART Domain          : root
Boot HART Priv Version    : v1.12
Boot HART Base ISA        : rv64imafdch
Boot HART ISA Extensions  : time,sstc
Boot HART PMP Count       : 16
Boot HART PMP Granularity : 4
Boot HART PMP Address Bits: 54
Boot HART MHPM Count      : 16
Boot HART MIDELEG         : 0x0000000000001666
Boot HART MEDELEG         : 0x0000000000f0b509
```

出现问题

opensbi启动成功，但是没有其他反应，同时观察到某一核的cpu占用很高，在运行五分钟后也依然没有其他输出

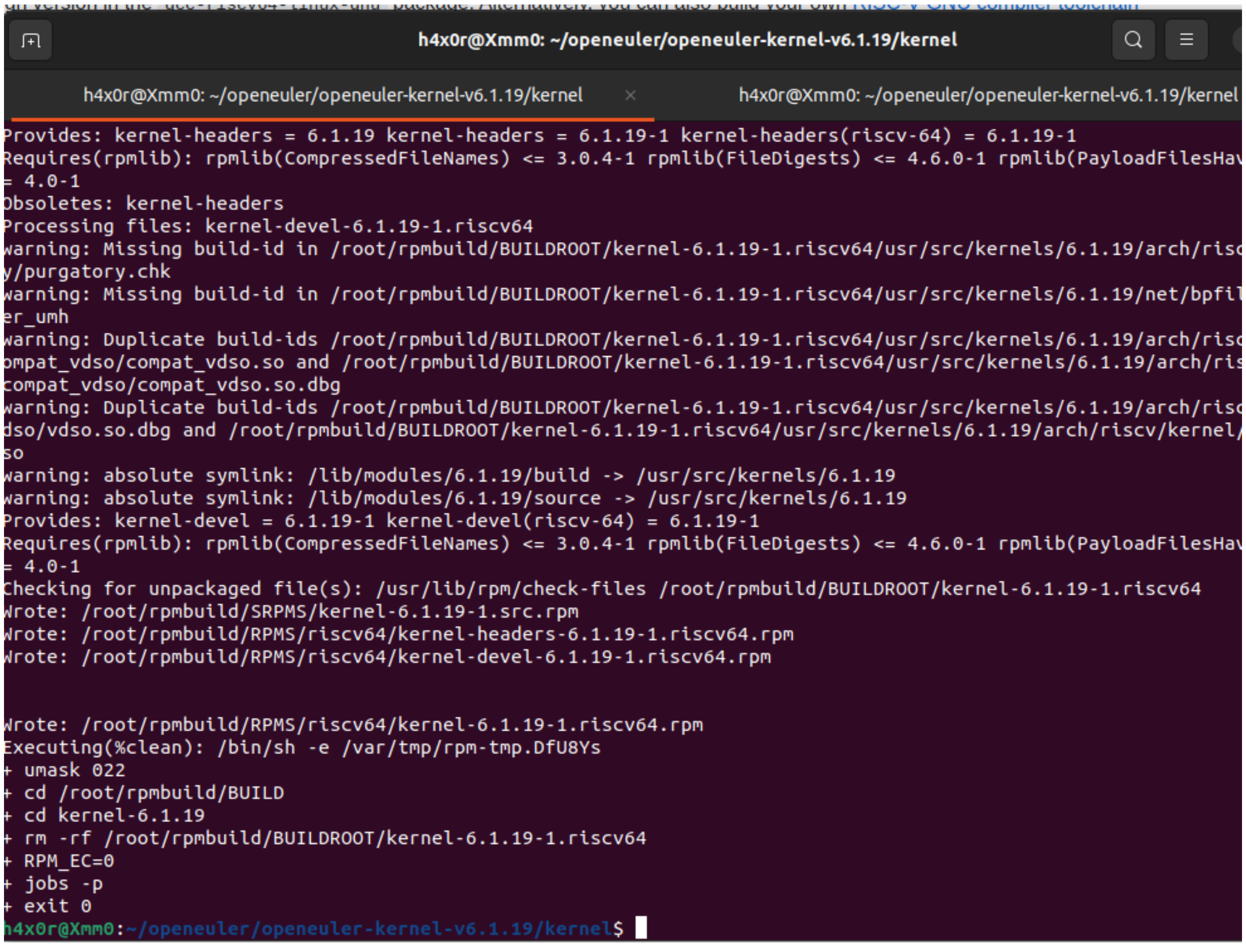

Temp [C]
93.8
Acpitz,0
Edge,0
Tctl,0
Tccd1,0
0.0
Frequency [MHz]
5491
3660
1830
0
Avg|Core|Core|Core|Core|Core|Core|Core|Core|Core|Core| Core|Core| Core|Core| Core|Core| Core|Core| Core|
Util [%]
100
0
Avg|Core|Core|Core|Core|Core|Core|Core|Core| Core|Core| Core|Core| Core|Core| Core|Core| Core|Core| Core|

```
  CPU Detected
 AMD Ryzen 9 7950X
 16-Core Processor

      Modes
(X) Monitor
( ) Stress

   Stress Timer
     00:00:00

  Control Options
 < Graphs          >
 < Summaries       >
 < Stress Options  >
 < Reset           >
 < Help            >
 < About           >
 < Save Settings   >
 < Quit            >

  Visual Options
[ ] UTF-8
Refresh[s]:2.0

     Summaries
Temp            [C]
Acpitz,0       16.8
Edge,0         41.0
Tctl,0         75.9
Tccd1,0        76.0
Tccd2,0        33.5
Composite,0    42.9
Composite,1    45.9

Frequency     [MHz]
Avg          3072.9
Core 0       3000.0
Core 1       3000.0
Core 2       3000.0
Core 3       5490.4
Core 4       3000.0
Core 5       3000.0
Core 6       3000.0
Core 7       3000.0
Core 8       2991.2
Core 9       3000.0
Core 10      3000.0
Core 11      3000.0
Core 12      2990.6
Core 13      2992.7
Core 14      3000.0
Core 15      3000.0
```

尝试解决

向群友提问后得到一种解决方案，可以先将内核编译为rpm包安装，提取出boot下的vmlinuz用于qemu启动

将内核编译为rpm包


h4x0r@Xmm0: ~/openeuler/openeuler-kernel-v6.1.19/kernel
h4x0r@Xmm0: ~/openeuler/openeuler-kernel-v6.1.19/kernel
h4x0r@Xmm0: ~/openeuler/openeuler-kernel-v6.1.19/kernel

```
Provides: kernel-headers = 6.1.19 kernel-headers = 6.1.19-1 kernel-headers(riscv-64) = 6.1.19-1
Requires(rpmlib): rpmlib(CompressedFileNames) <= 3.0.4-1 rpmlib(FileDigests) <= 4.6.0-1 rpmlib(PayloadFilesHav
= 4.0-1
Obsoletes: kernel-headers
Processing files: kernel-devel-6.1.19-1.riscv64
warning: Missing build-id in /root/rpmbuild/BUILDROOT/kernel-6.1.19-1.riscv64/usr/src/kernels/6.1.19/arch/risc
y/purgatory.chk
warning: Missing build-id in /root/rpmbuild/BUILDROOT/kernel-6.1.19-1.riscv64/usr/src/kernels/6.1.19/net/bpfil
er_umh
warning: Duplicate build-ids /root/rpmbuild/BUILDROOT/kernel-6.1.19-1.riscv64/usr/src/kernels/6.1.19/arch/risc
ompat_vdso/compat_vdso.so and /root/rpmbuild/BUILDROOT/kernel-6.1.19-1.riscv64/usr/src/kernels/6.1.19/arch/ris
compat_vdso/compat_vdso.so.dbg
warning: Duplicate build-ids /root/rpmbuild/BUILDROOT/kernel-6.1.19-1.riscv64/usr/src/kernels/6.1.19/arch/risc
dso/vdso.so.dbg and /root/rpmbuild/BUILDROOT/kernel-6.1.19-1.riscv64/usr/src/kernels/6.1.19/arch/riscv/kernel/
so
warning: absolute symlink: /lib/modules/6.1.19/build -> /usr/src/kernels/6.1.19
warning: absolute symlink: /lib/modules/6.1.19/source -> /usr/src/kernels/6.1.19
Provides: kernel-devel = 6.1.19-1 kernel-devel(riscv-64) = 6.1.19-1
Requires(rpmlib): rpmlib(CompressedFileNames) <= 3.0.4-1 rpmlib(FileDigests) <= 4.6.0-1 rpmlib(PayloadFilesHav
= 4.0-1
Checking for unpackaged file(s): /usr/lib/rpm/check-files /root/rpmbuild/BUILDROOT/kernel-6.1.19-1.riscv64
Wrote: /root/rpmbuild/SRPMS/kernel-6.1.19-1.src.rpm
Wrote: /root/rpmbuild/RPMS/riscv64/kernel-headers-6.1.19-1.riscv64.rpm
Wrote: /root/rpmbuild/RPMS/riscv64/kernel-devel-6.1.19-1.riscv64.rpm


Wrote: /root/rpmbuild/RPMS/riscv64/kernel-6.1.19-1.riscv64.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.DfU8Ys
+ umask 022
+ cd /root/rpmbuild/BUILD
+ cd kernel-6.1.19
+ rm -rf /root/rpmbuild/BUILDROOT/kernel-6.1.19-1.riscv64
+ RPM_EC=0
+ jobs -p
+ exit 0
h4x0r@Xmm0:~/openeuler/openeuler-kernel-v6.1.19/kernel$
```

此处遇到一个问题，oerv 23.03的内核版本为6.1.19-2，恰好高于编译出的版本一些，导致rpm无法升级

于是将oerv23.03更换为oerv 22.03来解决

但是在提取出vmlinuz后opensbi依然在启动过程中挂起