AKIBA

Воркшоп по отладке ядра Linux в Qemu

25.01.2025

@g1inko
@Daniil159x

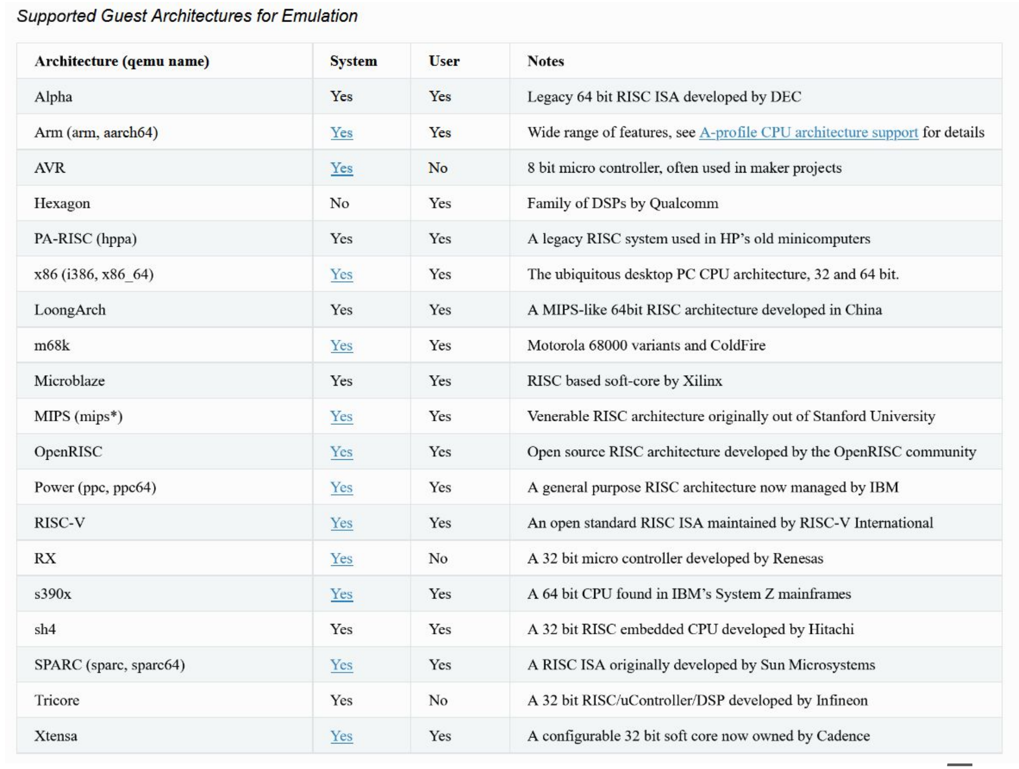
Что такое этот ваш QEMU ?

Эмулятор/виртуальная машина

- qemu-system
- qemu-user

Чтобы отлаживать ядро, нужны:

- vmlinux
- дебаг символы ядра, сорцы
- gdb хелперы



Supported Guest Architectures for Emulation

Architecture (qemu name)	System	User	Notes
Alpha	Yes	Yes	Legacy 64 bit RISC ISA developed by DEC
Arm (arm, aarch64)	Yes	Yes	Wide range of features, see A-profile CPU architecture support for details
AVR	Yes	No	8 bit micro controller, often used in maker projects
Hexagon	No	Yes	Family of DSPs by Qualcomm
PA-RISC (hppa)	Yes	Yes	A legacy RISC system used in HP's old minicomputers
x86 (i386, x86_64)	Yes	Yes	The ubiquitous desktop PC CPU architecture, 32 and 64 bit.
LoongArch	Yes	Yes	A MIPS-like 64bit RISC architecture developed in China
m68k	Yes	Yes	Motorola 68000 variants and ColdFire
Microblaze	Yes	Yes	RISC based soft-core by Xilinx
MIPS (mips*)	Yes	Yes	Venerable RISC architecture originally out of Stanford University
OpenRISC	Yes	Yes	Open source RISC architecture developed by the OpenRISC community
Power (ppc, ppc64)	Yes	Yes	A general purpose RISC architecture now managed by IBM
RISC-V	Yes	Yes	An open standard RISC ISA maintained by RISC-V International
RX	Yes	No	A 32 bit micro controller developed by Renesas
s390x	Yes	Yes	A 64 bit CPU found in IBM's System Z mainframes
sh4	Yes	Yes	A 32 bit RISC embedded CPU developed by Hitachi
SPARC (sparc, sparc64)	Yes	Yes	A RISC ISA originally developed by Sun Microsystems
Tricore	Yes	No	A 32 bit RISC/uController/DSP developed by Infineon
Xtensa	Yes	Yes	A configurable 32 bit soft core now owned by Cadence

https://www.qemu.org/docs/master/about/emulation.html

Установка ubuntu server внутри qemu

1. qemu-img create -f qcow2 disk.qcow2 30G
2. qemu-system-x86_64 \
   ```
   -boot d \
   -cdrom ubuntu-24.04.1-live-server-amd64.iso \
   -m 4G \
   -cpu host \
   -enable-kvm -smp 4 \
   -hda ./disk.qcow2
   ```
3. далее далее далее
   a. на этапе разбиения диска - убрать LVM раздел
4. откиньтесь на спинку кресла

Достаем ядро (без загрузки системы)

```
1.   modprobe nbd max_part=8
2.   qemu-nbd --connect=/dev/nbd0 ./disk.qcow2
3.   fdisk /dev/nbd0 -l
4.   mount /dev/nbd0p2 /mnt/qemu/

5.   cp /mnt/qemu/boot/vmlinuz-6.8.0-49-generic ./
6.   cp /mnt/qemu/boot/initrd.img-6.8.0-49-generic ./

7.   umount /mnt/qemu/
8.   qemu-nbd --disconnect /dev/nbd0
9.   rmmod nbd
```

https://gist.github.com/shamil/62935d9b456a6f9877b5

Загрузка ubuntu server внутри qemu

```
#!/bin/bash

BOOT_DIR="./boot"
VERSION="6.8.0-49-generic"

exec qemu-system-x86_64 \
     -kernel "${BOOT_DIR}/vmlinuz-${VERSION}" \
     -initrd "${BOOT_DIR}/initrd.img-${VERSION}" \
     -nographic \
     -cpu host -enable-kvm \
     -append "console=ttyS0 root=/dev/sda2 nokaslr" \
     -no-reboot \
     -s \
     -m 4G \
     -device e1000,netdev=net0 \
     -netdev user,id=net0,hostfwd=tcp::5555-:22 \
     -hda ./disk.qcow2
```

Serial - это не стабильно

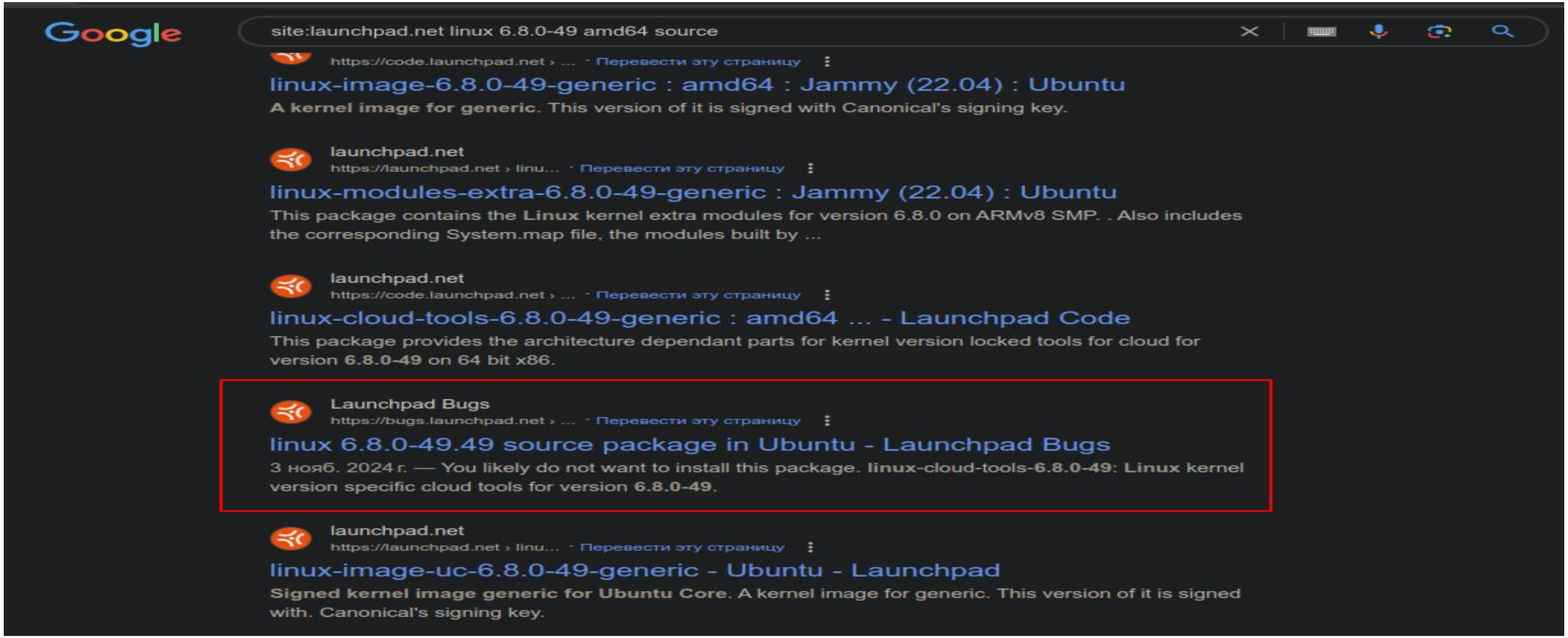
1.
```
echo 0 | sudo tee \
/proc/sys/kernel/yama/ptrace_scope
# чтобы аттачиться и парсить память
```

2.
```
./run.sh # запускаем ВМ
```

3.
```
ssh -p 5555 vm@127.0.0.1 \
    -o IdentitiesOnly=yes \
    -o ServerAliveCountMax=999999
```

4. теперь можно делать `ssh` и `scp`

Вытаскиваем символы и исходники .1



site:launchpad.net linux 6.8.0-49 amd64 source

https://code.launchpad.net › ... · Перевести эту страницу

linux-image-6.8.0-49-generic : amd64 : Jammy (22.04) : Ubuntu

A kernel image for generic. This version of it is signed with Canonical's signing key.

launchpad.net

https://launchpad.net › linu... · Перевести эту страницу

linux-modules-extra-6.8.0-49-generic : Jammy (22.04) : Ubuntu

This package contains the Linux kernel extra modules for version 6.8.0 on ARMv8 SMP. . Also includes the corresponding System.map file, the modules built by ...

launchpad.net

https://code.launchpad.net › ... · Перевести эту страницу

linux-cloud-tools-6.8.0-49-generic : amd64 ... - Launchpad Code

This package provides the architecture dependant parts for kernel version locked tools for cloud for version 6.8.0-49 on 64 bit x86.

Launchpad Bugs

https://bugs.launchpad.net › ... · Перевести эту страницу

linux 6.8.0-49.49 source package in Ubuntu - Launchpad Bugs

3 нояб. 2024 г. — You likely do not want to install this package. linux-cloud-tools-6.8.0-49: Linux kernel version specific cloud tools for version 6.8.0-49.

launchpad.net

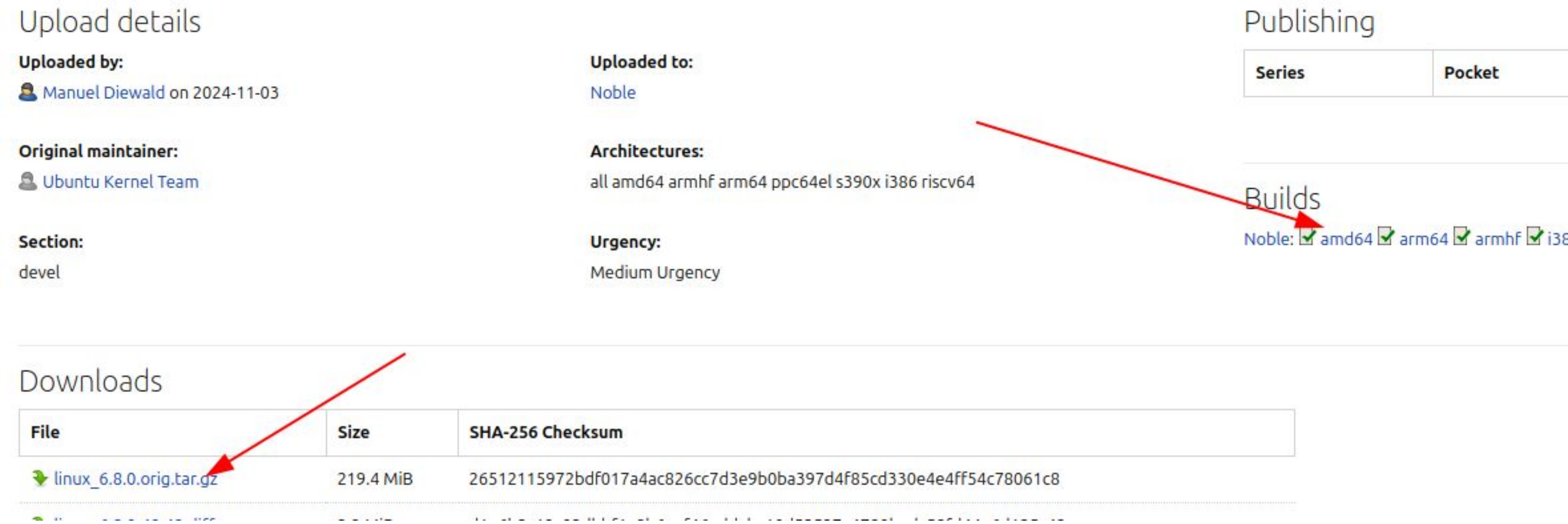
https://launchpad.net › linu... · Перевести эту страницу

linux-image-uc-6.8.0-49-generic - Ubuntu - Launchpad

Signed kernel image generic for Ubuntu Core. A kernel image for generic. This version of it is signed with. Canonical's signing key.

https://launchpad.net/ubuntu/+source/linux/6.8.0-49.49

Вытаскиваем символы и исходники .2



Upload details

Uploaded by:
Manuel Diewald on 2024-11-03

Uploaded to:
Noble

Original maintainer:
Ubuntu Kernel Team

Architectures:
all amd64 armhf arm64 ppc64el s390x i386 riscv64

Section:
devel

Urgency:
Medium Urgency

Publishing

Series	Pocket

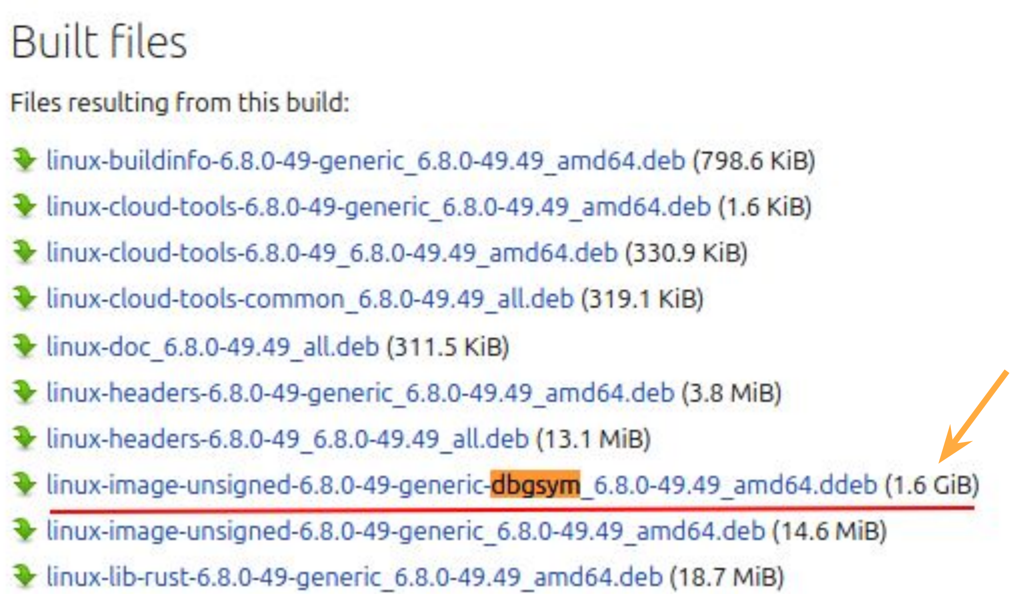
Builds

Noble: ✓ amd64 ✓ arm64 ✓ armhf ✓ i38

Downloads

File	Size	SHA-256 Checksum
linux_6.8.0.orig.tar.gz	219.4 MiB	26512115972bdf017a4ac826cc7d3e9b0ba397d4f85cd330e4e4ff54c78061c8

Вытаскиваем символы и исходники .2



Built files

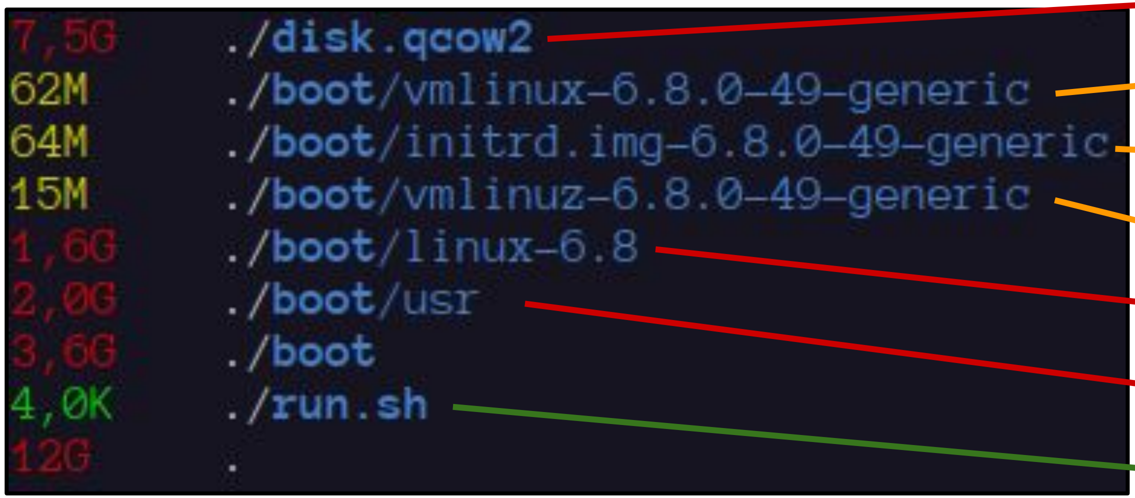
Files resulting from this build:

- linux-buildinfo-6.8.0-49-generic_6.8.0-49.49_amd64.deb (798.6 KiB)
- linux-cloud-tools-6.8.0-49-generic_6.8.0-49.49_amd64.deb (1.6 KiB)
- linux-cloud-tools-6.8.0-49_6.8.0-49.49_amd64.deb (330.9 KiB)
- linux-cloud-tools-common_6.8.0-49.49_all.deb (319.1 KiB)
- linux-doc_6.8.0-49.49_all.deb (311.5 KiB)
- linux-headers-6.8.0-49-generic_6.8.0-49.49_amd64.deb (3.8 MiB)
- linux-headers-6.8.0-49_6.8.0-49.49_all.deb (13.1 MiB)
- linux-image-unsigned-6.8.0-49-generic-dbgsym_6.8.0-49.49_amd64.ddeb (1.6 GiB)
- linux-image-unsigned-6.8.0-49-generic_6.8.0-49.49_amd64.deb (14.6 MiB)
- linux-lib-rust-6.8.0-49-generic_6.8.0-49.49_amd64.deb (18.7 MiB)

Готовим окружение

1. `mkdir boot`
2. `cp initrd.img-6.8.0-49-generic ./boot`
3. `cp vmlinuz-6.8.0-49-generic ./boot`
4. `dpkg-deb -x linux-image-unsigned-6.8.0-49-generic-dbgsym_6.8.0-49.49_amd64.ddeb ./boot`
5. `tar -xvf ./linux_6.8.0.orig.tar.gz -C ./boot/`
6. `cd boot;`
7. ~~bash ./linux-6.8/scripts/extract-vmlinux ./vmlinuz-6.8.0-49-generic > ./vmlinux-6.8.0-49-generic~~

Готовим окружение



```
7,5G        ./disk.qcow2
62M         ./boot/vmlinux-6.8.0-49-generic
64M         ./boot/initrd.img-6.8.0-49-generic
15M         ./boot/vmlinuz-6.8.0-49-generic
1,6G        ./boot/linux-6.8
2,0G        ./boot/usr
3,6G        ./boot
4,0K        ./run.sh
12G         .
```

образ диска

символы ядра

initrd (ранее initramfs)

запакованное ядро без символов

сорцы ядра

там лежат дебаг скрипты

запуск кему

gdb и pwndbg

git clone https://github.com/pwndbg/pwndbg + setup.sh

либо

~~RTFM~~

либо portable версия

https://github.com/pwndbg/pwndbg/releases/download/2025.01.20/pwndbg_2025.01.20_amd64.tar.xz

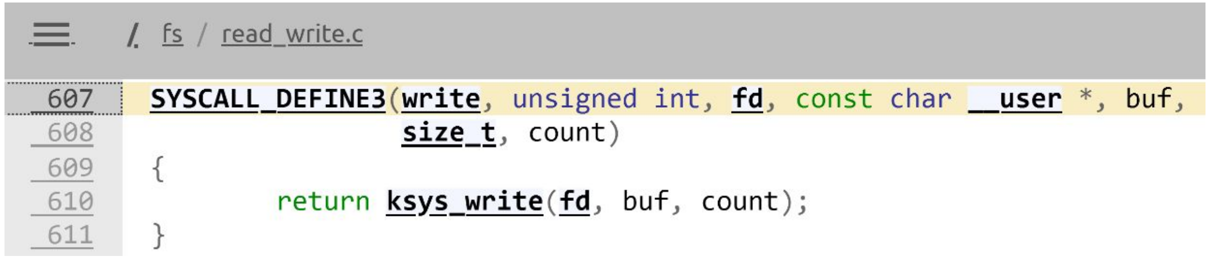
hello debugger

1. запускаем qemu через `./run.sh`
2. `cd boot`
3. `gdb ./usr/lib/debug/boot/vmlinux-6.8.0-49-generic` или `pwndbg/bin/pwndbg ...`
4. `source usr/share/gdb/auto-load/boot/vmlinux-6.8.0-49-generic/vmlinuz-6.8.0-49-generic-gdb.py`

hello debugger

1. (gdb) `target remote :1234`

2.
   ```
   set substitute-path /build/linux-uoESLx/linux-6.8.0 \
   <your>/boot/linux-6.8
   ```
3. break ksys_write
4. c
5. *pls wait…*


/ fs / read_write.c

```
SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,
                size_t, count)
{
        return ksys_write(fd, buf, count);
}
```

hello debugger (pt. 2)

1. **bt**
2. **info arg**
3. **info locals**
4. **hexdump <адрес или символ> [длина]**
5. **p <символ>** # покажет тип, если он связан с символом
6. **ptype <символ/тип>**
7. **vmmap**

8. **ctx**
9. **si [N]**
10. **nextcall**
11. **fin** # до выхода из функции

Ядерные хелперы для отладки

- `lx-ps`
- `ls-dmesg`
- …

- `apropos lx`

- `ptype struct task_struct`
- `p $lx_current()`
- `p $lx_current().pid`

Условные бряки

- `info break`
- `del 1 # или dis 1`
- `lx-ps`
- `b ksys_write if $lx_current().pid == 112`
- `c`

- `(gdb) b do_sys_openat2 if $lx_current().pid == 1015`
- `(vm) echo < /proc/self/comm`

Таск тайм

(vm) wget http://192.168.8.58:8000/guess_game_51.ko

sudo insmod guess_game_51.ko

cat /proc/guess

echo 123 > /proc/guess

(gdb) b proc_reg_write if `$lx_current().pid == <your bash>`

Ссылки

- https://docs.kernel.org/dev-tools/gdb-kernel-debugging.html
- https://wiki.qemu.org/Documentation
- https://wiki.qemu.org/Documentation/Networking
- https://pwndbg.re/CheatSheet.pdf
- https://github.com/pwndbg/pwndbg/blob/dev/FEATURES.md
- …