环境搭建

安装qemu

sudo apt-get install qemu qemu-system

内核安装与编译

https://mirrors.edge.kernel.org/pub/linux/kernel

到这个网站上找一个内核版本，wget下来

```
tar -zxvf 文件名.tar.gz
cd linux-5.4.7/
apt-get install libncurses5-dev build-essential kernel-package
make menuconfig
```

配置一些选项，主要就是：

```
KernelHacking –>
```

```
选中Compile the kernel with debug info
选中Compile the kernel with frame pointers
选中KGDB:kernel debugging with remote gdb，其下的全部都选中。
Processor type and features–>

去掉Paravirtualized guest support
KernelHacking–>

去掉Write protect kernel read-only data structures（否则不能用软件断点）
```

搞完之后

```
make
make all
make modules
```

之后还要安装busybox来启动内核

```
cd ..
wget https://busybox.net/downloads/busybox-1.29.3.tar.bz
tar -jxvf busybox-1.29.3.tar.bz2
cd busybox-1.29.3
make menuconfig
make install
```

一些相关的配置

make menuconfig 设置

Busybox Settings -> Build Options -> Build Busybox as a static binary 编译成 静态文件

关闭下面两个选项

Linux System Utilities -> [] Support mounting NFS file system 网络文件系统 Networking Utilities -> [] inetd (Internet超级服务器)

构建文件系统

```
cd _install
mkdir proc sys dev etc etc/init.d
vim etc/init.d/rcS  /
```

在rcs文件里写入一下内容：

```
#!/bin/sh
mount -t proc none /proc
mount -t sysfs none /sys
/sbin/mdev -s
```

而后

```
chmod +x etc/init.d/rCS //一定要有，否则会出现启动tty no such file错误。
find . | cpio -o --format=newc > ../rootfs.img
```

这样配置过程就完成了，qemu就可以启动内核了

```
root@s3cunDa:/# qemu-system-x86_64 -kernel /pwn/kernel_test/linux-4.4.11/arch/x86/boot/bzImage -initrd /pwn/kernel_test/busybox-1.29.3/rootfs.img -append "console=ttyS0 root=/dev/ram rdinit=/sbin/init" -cpu kvm64,+smep,+smap --nographic -gdb tcp::1111
```

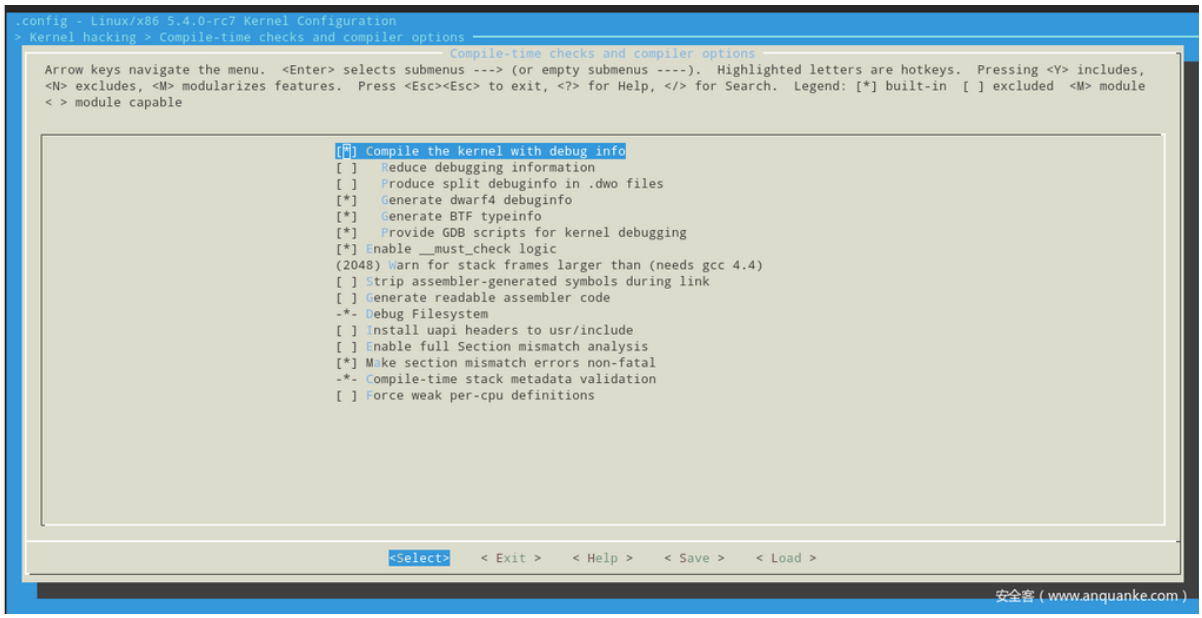
挂载了tcp端口，我们就可以利用gdb调试内核了，在调试之前输入gdb -q vmlinux的路径，这样调试的时候就会有相应的符号，进入gdb后输入remote target 端口号就可以调试了。

环境搭建方法2

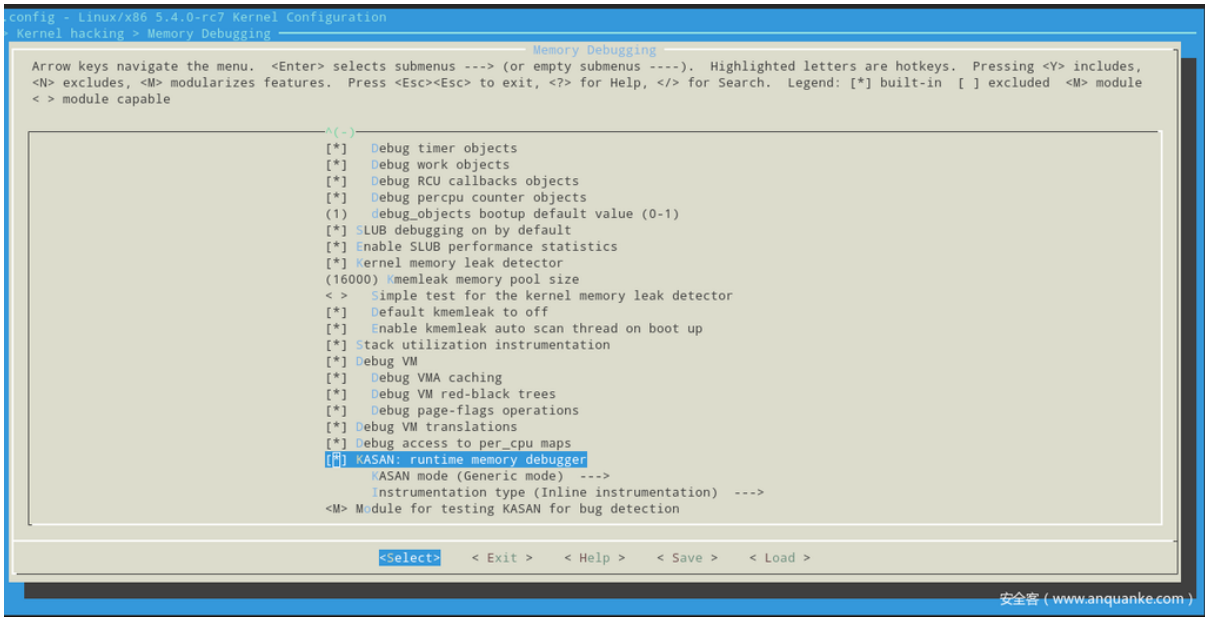
选择是5.4.7内核，先下载源码，然后

make menuconfig

在kernel hacking下的compile-time checks and compiler options选中添加符号表，这样的话，调试的时候会舒服很多



添加一些内存检测的机制kasan,还是在kernel hacking里面的memory debugging，能选的全选上



1.2 选择镜像

镜像的话，我使用的https://github.com/google/syzkaller/blob/master/docs/linux/setup_ubuntu-host_qemu-vm_x86-64-kernel.md，也可以在我的目录下找到。

主要这个可以用ssh，就不能每写一次poc就得打包一次，还可以在镜像里面直接编译，比较方便，但是会一些问题。

ssh连接不上qemu，如何解决

```
vi /etc/network/interfaces
```

```
auto lo
iface lo inet loopback
#auto eth0
#iface eth0 inet dhcp
auto enp0s3
iface enp0s3 inet dhcp
```

安装checksec

pip instasll pwntools

安装ropper

```
pip install capstone
python -m pip install --upgrade pip
pip install filebytes
```

```
pip install setuptools --upgrade
pip install ropper
```

ropper -f vmlinux --nocolor > goldfish_ropper_gadgets

常见问题

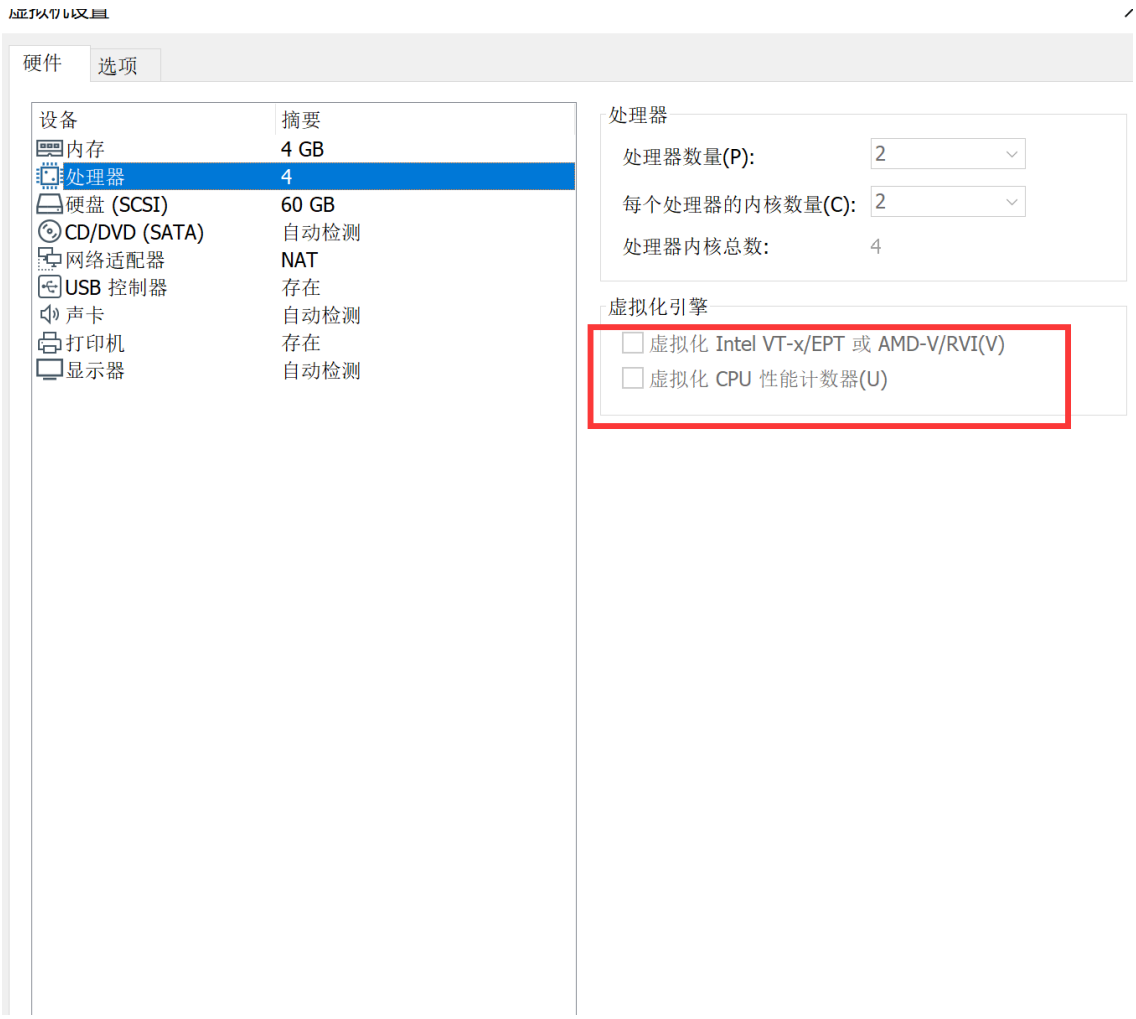
1.

```
sudo qemu-system-x86_64 -m 2G -smp 2 -kernel /boot/bzImage -append
"console=ttyS0 root=/dev/sda earlyprintk=serial"-drive file=/stretch.img,format=raw -net
user,host=10.0.2.10,hostfwd=tcp:127.0.0.1:10021-:22 -net nic,model=e1000 -enable-
kvm -nographic -pidfile vm.pid 2>&1 | tee vm.log
qemu-system-x86_64: -append console=ttyS0 root=/dev/sda earlyprintk=serial-drive:
Could not open 'file=/stretch.img,format=raw': No such file or directory
```

是因为缺少空格，检查语法

2. 当create_image卡住，可以考虑换国内的镜像。
   debootstrap 那一行末尾加上http://mirrors.163.com/debian/

3. cannot access kvm module



1.https://www.kernel.org/

1. E: Could not get lock /var/lib/dpkg/lock-frontend - open (11: Resource temporarily unavailable)

sudo rm /var/lib/dpkg/lock-frontend
sudo rm /var/lib/dpkg/lock
sudo rm /var/cache/apt/archives/lock

wget https://mirrors.edge.kernel.org/pub/linux/kernel/v5.x/linux-5.0.tar.gz
tar -xvf linux-5.0.tar.gz

https://cc-sir.github.io/2019/07/24/Linux-kernel-1/

https://bbs.pediy.com/thread-252826.htm

https://cc-sir.github.io/2019/07/24/Linux-kernel-0/

https://xz.aliyun.com/u/20469