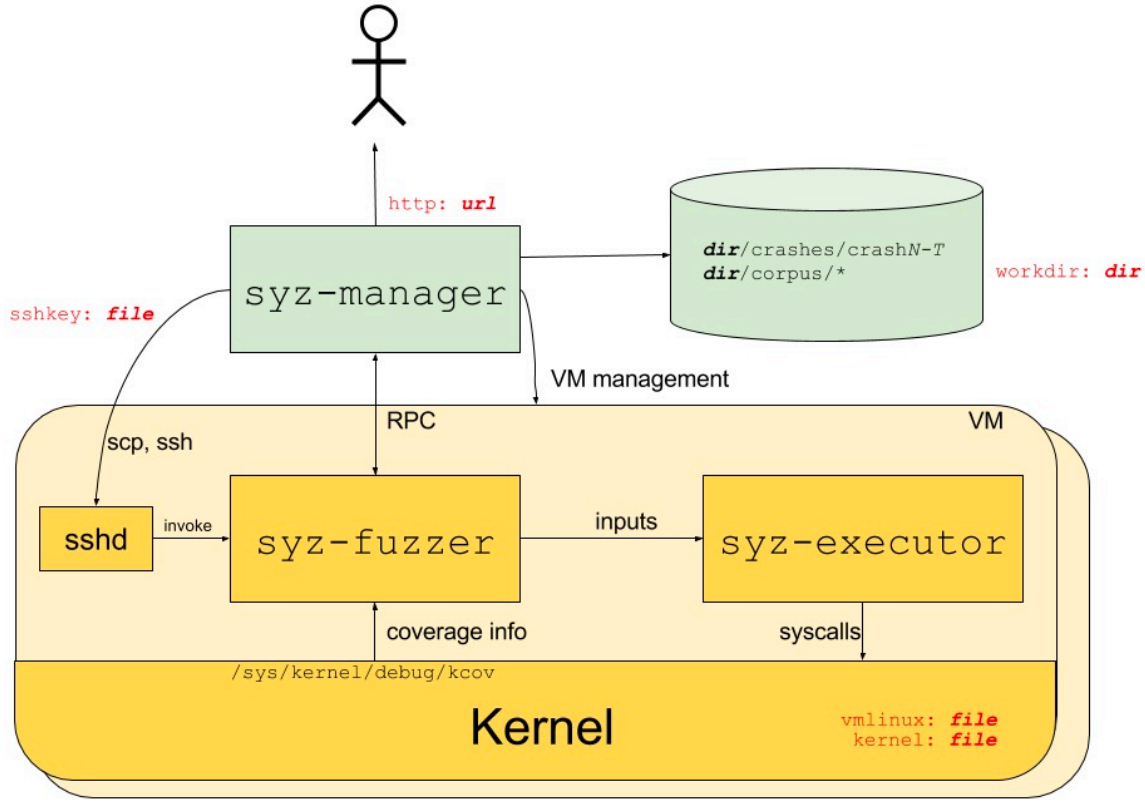
fuzz-使用Syzkaller测试Linux内核

Syzkaller由Google主导，由Go语言编写，根据其官网的介绍：

syzkaller 是一个非监督式、覆盖率指导的模糊测试器，支持： Akaros 、FreeBSD、Fuchsia、 gVisor、 Linux 、NetBSD、 OpenBSD、 Windows。

其整体框架如下：



图中黄色部分运行在QEMU虚拟机中，而绿色部分运行在宿主机中。
测试者从宿主机中通过syz-manager启动测试，它首先根据配置使用编译好的内核启动QEMU虚拟机，然后通过ssh远程连接发送syz-fuzzer可执行文件并到虚拟机中并启动之，再后两个syz-* 通过RPC进行通信，后者对内核进行fuzzing test并将相关结果发送回前者

1. 安装Syzkaller

需要先安装go语言环境，Ubuntu下直接sudo apt install golang万事大吉，然后：

```
mv go goroot
mkdir gopath
export GOPATH=`pwd`/gopath
export GOROOT=`pwd`/goroot
export PATH=$GOPATH/bin:$PATH
export PATH=$GOROOT/bin:$PATH
go get -u -v github.com/google/syzkaller/
make -C ~/go/src/github.com/google/syzkaller/ # ~/go/应替换成实际GOPATH，下同
ls ~/go/src/github.com/google/syzkaller/bin/
```

编译后生成syz-fuzzer和syz-manager等命令，将其目录添加到系统PATH环境变量中完成Sykaller的安装

2. 安装qemu-kvm

```
sudo apt install qemu-kvm
sudo usermod -aG kvm $USER
```

kvm功能可以让QEMU执行得更快，同时为了让启用kvm支持不需要root权限，需要将当前用户添加到kvm组中去，完成后注销登陆即可。

3. 制作系统镜像

Syzkaller虽然是对内核进行测试，但是内核一个光杆司令是没办法用的，所以需要制作一个系统镜像，其中需要提供ssh、gcc等工作环境。

偷懒的方法是使用Syzkaller官方提供的脚本：

```
# 安装debootstrap
sudo apt install debootstrap
```

```
# 下载脚本
wget https://raw.githubusercontent.com/google/syzkaller/master/tools/create-image.sh -O create-image.sh
# 添加可执行权限
chmod +x create-image.sh
# 使用清华源
sed -i -e 's~sudo debootstrap .*~\0 https://mirrors.tuna.tsinghua.edu.cn/debian/~' create-image.sh
# 制作镜像，1024MB
./create-image.sh -s 1024
```

其脚本大致的操作为：

- 基于debian stretch并添加openssh-server、curl、tar等软件包，在当前目录下创建了一个chroot系统目录
- 将镜像系统的root用户密码设置为空、生成ssh密钥对并添加之，更改了一些必要的系统选项
- 将该目录写入到一个ext4格式的镜像文件中去，即./stretch.img

4. 编译待测内核

```
# 先克隆一个linux 5.0版的代码仓库并进入目录

# 先采用默认配置
make defconfig
# 启用kvm
make kvmconfig
# Syzkaller需要启用一些调试功能
echo '
CONFIG_KCOV=y
CONFIG_DEBUG_INFO=y
CONFIG_KASAN=y
CONFIG_KASAN_INLINE=y
CONFIG_CONFIGFS_FS=y
```

```
CONFIG_SECURITYFS=y' >> .config
# 再次对新引入的配置采用默认值
make olddefconfig
```

4.2. 注入内核bug

```
diff --git a/fs/open.c b/fs/open.c
index 0285ce7db..3ab215a93 100644
--- a/fs/open.c
+++ b/fs/open.c
@@ -523,6 +523,12 @@ SYSCALL_DEFINE1(chroot, const char __user *, filename)

 static int chmod_common(const struct path *path, umode_t mode)
 {
+    static umode_t old_mode = 0xffff;
+    if (old_mode == 0 && mode == 0) {
+        path = NULL;
+    }
+    old_mode = mode;
+
         struct inode *inode = path->dentry->d_inode;
         struct inode *delegated_inode = NULL;
         struct iattr newattrs;
```

往fs/open.c中添加一段代码，使得当连续两次chmod调用的mode参数值为0时会产生空指针解引用异常。

4.3. 编译内核二进制文件

直接make -j 16搞定。

5. 尝试虚拟机启动

```
qemu-system-x86_64 -m 1G -enable-kvm -drive file=<stretch.img路径>,format=raw -kernel <linux源码路径>/arch/x86/boot/bzImage -append root=/dev/sda
```

正常的话，会进入TTY（，显示界面可能会有旧输出的残留，仔细看），用户root，密码为空，就可以使用这个系统了。

执行uname -a，会输出内核信息，就是前面编译的版本。

6. 配置Syzkaller

Syzkaller通过json文件进行配置

```
{
    "target": "linux/amd64",
    "http": "127.0.0.1:8080",
    "workdir": "${WORKDIR}",
    "kernel_obj": "${LINUX}/vmlinux",
    "image": "${IMAGE}/stretch.img",
    "sshkey": "${IMAGE}/stretch.id_rsa",
    "syzkaller": "${GOPATH}/src/github.com/google/syzkaller",
    "enable_syscalls": ["chmod"],
    "procs": 1,
    "type": "qemu",
    "vm": {
        "count": 1,
        "kernel": "${LINUX}/arch/x86/boot/bzImage",
        "cpu": 1,
        "mem": 1024
```

```
    }
}
```

其中：

{WORKDIR}是需要替换为所需的工作目录，之后生成的crash文件将会位于其中，{LINUX}为Linux源码目录，{IMAGE}为方才制作的系统镜像与密钥文件目录，{GOPATH}替换为安装Syzkaller所使用的GOPATH。
enable_syscalls设置为["chmod"]，表示只对chmod调用进行测试。

7. 执行测试

syz-manager --config config.json

输出如下

```
2019/12/16 20:46:31 loading corpus...
2019/12/16 20:46:31 serving http on http://127.0.0.1:8080
2019/12/16 20:46:31 serving rpc on tcp://[::]:45379
2019/12/16 20:46:31 booting test machines...
2019/12/16 20:46:31 wait for the connection from test machine...
2019/12/16 20:46:43 machine check:
2019/12/16 20:46:43 syscalls              : 1/3039
2019/12/16 20:46:43 code coverage         : enabled
2019/12/16 20:46:43 comparison tracing    :
CONFIG_KCOV_ENABLE_COMPARISONS is not enabled
2019/12/16 20:46:43 extra coverage        : extra coverage is not supported by the
kernel
2019/12/16 20:46:43 setuid sandbox        : enabled
2019/12/16 20:46:43 namespace sandbox     : /proc/self/ns/user does not exist
2019/12/16 20:46:43 Android sandbox       : enabled
2019/12/16 20:46:43 fault injection       : CONFIG_FAULT_INJECTION is not enabled
2019/12/16 20:46:43 leak checking         : CONFIG_DEBUG_KMEMLEAK is not
enabled
2019/12/16 20:46:43 net packet injection  : /dev/net/tun does not exist
```

```
2019/12/16 20:46:43 net device setup       : enabled
2019/12/16 20:46:43 concurrency sanitizer  : /sys/kernel/debug/kcsan does not exist
2019/12/16 20:46:43 devlink PCI setup      : PCI device 0000:00:10.0 is not available
2019/12/16 20:46:43 corpus                 : 0 (0 deleted)
2019/12/16 20:46:51 VMs 1, executed 0, cover 525, crashes 0, repro 0
2019/12/16 20:46:55 vm-0: crash: general protection fault in chmod_common
2019/12/16 20:46:55 reproducing crash 'general protection fault in chmod_common':
182 programs, 1 VMs, timeouts [15s 1m0s 6m0s]
2019/12/16 20:47:01 VMs 0, executed 0, cover 525, crashes 1, repro 1
2019/12/16 20:47:11 VMs 0, executed 0, cover 525, crashes 1, repro 1
2019/12/16 20:47:21 VMs 0, executed 0, cover 525, crashes 1, repro 1
```

由于注入的bug十分简单，运行后不久就触发了crash，并且该crash成功复现。
打开${WORKDIR}/crash/<some-sha-value>/log0文件，有该crash的log信息，定位到尾部：

```
12:46:45 executing program 0:
chmod(&(0x7f0000000000)='./file0\x00', 0x0)
chmod(&(0x7f0000000040)='./file0\x00', 0x51ed4676016e28cd)

12:46:45 executing program 0:
chmod(&(0x7f0000000080)='.\x00', 0x0)
chmod(&(0x7f0000000040)='./file0\x00', 0x0)
chmod(&(0x7f0000000000)='./file0\x00', 0xc4)

[   13.843869] kasan: CONFIG_KASAN_INLINE enabled
[   13.844352] kasan: GPF could be caused by NULL-ptr deref or user memory access
[   13.845029] general protection fault: 0000 [#1] SMP KASAN PTI
[   13.845593] CPU: 0 PID: 3036 Comm: syz-executor.0 Not tainted 5.0.0+ #1
[   13.846330] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS 1.10.2-
1ubuntu1 04/01/2014
[   13.847256] RIP: 0010:chmod_common+0xab/0x3d0
[   13.847725] Code: 66 0b 05 a8 60 9e 02 b8 00 00 00 00 4c 0f 44 e0 e8 2a 7c de ff 66
89 1d 93 60 9e 02 49 8d 44 24 08 48 89 44 24 38 48 c1 e8 03 <80> 3c 28 00 0f 85 e4 02
00 00 48 b8 00 00 00 00 00 fc ff df 49 8b
[   13.849561] RSP: 0018:ffff888033f4fd20 EFLAGS: 00010202
[   13.850052] RAX: 0000000000000001 RBX: 0000000000000000 RCX: ffffffff815546e6
```

```
[   13.850718] RDX: 00000000000000e0 RSI: ffffc9000045f000 RDI: ffff888033f4fea0
[   13.851389] RBP: dfffffc0000000000 R08: 0000000000000009 R09:
00000000b789e4eb
```

确实连续两次chmod调用的mode参数为0，导致了NULL-ptr deref or user memory access错误。

这个log文件可以进一步处理，譬如定位bug等.

> https://i-m.dev/posts/20200313-143737.html

https://xz.aliyun.com/t/5079

https://www.jianshu.com/p/790b733f80a2