

如何获取一个crash的复现程序

1. 将log程序转移至操作作用的虚拟机中

所有syzkaller相关虚拟机（VM109，
VM112~115）的密码皆为：qinjiaji

syzkaller: VM115

向日葵特征码：663511178

验证码：qaz123



2. 利用syz-repro从log中获取可复现程序

2.1 获取可复现的系统调用序列

```
1 ./bin/syz-repro -config=my613.cfg -output=repro.txt -crepro=repro.c -  
  title=bug_title.txt log0
```

该命令将执行以下操作：

1. 加载 `manager.cfg` 配置文件。
2. 读取 `crash.log` 崩溃日志文件。
3. 创建虚拟机池并开始重现过程。
4. 将重现的程序保存到 `repro.txt`。
5. 生成 C 语言的重现程序并保存到 `repro.c`。
6. 保存 bug 的标题到 `bug_title.txt`。

syz-repro会不断尝试最小化复现程序，过程如下：

但值得注意的是，最终的复现程序由于过于简短，反而不能稳定触发crash。此时可选取中间POC作为最终的复现程序。

```

2 perf_event_open(&(0x7f00000003c0)={0x2, 0x80, 0xe7, 0x0, 0x0, 0x0, 0x0, 0x1,  
    0x44c21d, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,  
    0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,  
    0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,  
    @perf_config_ext={0x202ece72, 0xfffffffffffffffe05}, 0x2802, 0x3, 0x91, 0x6,  
    0x8, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x14}, 0x0, 0xfffffffffffffffffff,  
    0xffffffffffffffffffff, 0x0) (async, rerun: 64)  
3 r0 = socket$pppoe(0x18, 0x1, 0x0) (rerun: 64)  
4 connect$pppoe(r0, &(0x7f0000000400)={0x18, 0x0, {0x2, @dev=  
    {'\xaa\xaa\xaa\xaa\xaa', 0xa}, '\lo\x00'}}, 0x1e) (async)  
5 r1 = socket$nL_rdma(0x10, 0x3, 0x14)  
6 openat$rDMA_cm(0xfffffffffffffff9c, &(0x7f0000000280), 0x2, 0x0) (async)  
7 sendmsg$RDMA_NLDEV_CMD_NEWLINK(r1, &(0x7f0000000440)={0x0, 0x0, &  
    (0x7f0000000400)={&(0x7f0000000280)=ANY=[@ANYBLOB="38000000141401"], 0x38},  
    0x1, 0x0, 0x0, 0x4000}, 0x10)  
  
8  
9 # V2  
10 perf_event_open(&(0x7f00000003c0)={0x2, 0x80, 0xe7, 0x0, 0x0, 0x0, 0x0, 0x1,  
    0x44c21d, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,  
    0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,  
    0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,  
    @perf_config_ext={0x202ece72, 0xfffffffffffffffe05}, 0x2802, 0x3, 0x91, 0x6,  
    0x8, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x14}, 0x0, 0xfffffffffffffffffff,  
    0xffffffffffffffffffff, 0x0) (async, rerun: 64)  
11 r0 = socket$pppoe(0x18, 0x1, 0x0) (rerun: 64)  
12 connect$pppoe(r0, &(0x7f0000000400)={0x18, 0x0, {0x2, @dev=  
    {'\xaa\xaa\xaa\xaa\xaa', 0xa}, '\lo\x00'}}, 0x1e) (async)  
13 socket$nL_rdma(0x10, 0x3, 0x14)  
14 openat$rDMA_cm(0xfffffffffffffff9c, &(0x7f0000000280), 0x2, 0x0) (async)  
15  
16 # V3  
17 perf_event_open(&(0x7f00000003c0)={0x2, 0x80, 0xe7, 0x0, 0x0, 0x0, 0x0, 0x1,  
    0x44c21d, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,  
    0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,  
    0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,  
    @perf_config_ext={0x202ece72, 0xfffffffffffffffe05}, 0x2802, 0x3, 0x91, 0x6,  
    0x8, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x14}, 0x0, 0xfffffffffffffffffff,  
    0xffffffffffffffffffff, 0x0) (async, rerun: 64)  
18 r0 = socket$pppoe(0x18, 0x1, 0x0) (rerun: 64)  
19 connect$pppoe(r0, &(0x7f0000000400)={0x18, 0x0, {0x2, @dev=  
    {'\xaa\xaa\xaa\xaa\xaa', 0xa}, '\lo\x00'}}, 0x1e) (async)  
20 socket$nL_rdma(0x10, 0x3, 0x14)  
21 # 红色部分是syz-repro提供的最终的POC，红色部分+绿色部分是在不断最小化的中间POC。经测试，  
    中间POC可以稳定触发该crash，而最终POC则不能。  
22 # 分析：按照sykaller文档所说，syz-repro并不完美，所以可能功能存在一定问题
```

```
</TASK>
extracting prog: 1m58.787654136s
minimizing prog: 21m57.530880135s
simplifying prog options: 7m25.791778345s
extracting C: 5m5.936792302s
simplifying C: 0s
opts: {Threaded:true Repeat:true RepeatTimes:0 Procs:8 Slowdown:1 Sandbox:none S
andboxArg:0 Leak:false NetInjection:true NetDevices:true NetReset:true Cgroups:t
rue BinfmtMisc:true CloseFDs:true KCSAN:false DevlinkPCI:true NicVF:true USB:tru
e VhciInjection:true Wifi:true IEEE802154:true Sysctl:true Swap:true UseTmpDir:t
rue HandleSegv:true Trace:false LegacyOptions:{Collide:false Fault:false FaultCa
ll:0 FaultNth:0}} crepro: false

perf_event_open(&(0x7f00000003c0)={0x2, 0x80, 0xe7, 0x0, 0x0, 0x0, 0x0, 0x1, 0x4
4c21d, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0
, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0
, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x203, 0x0, @perf_config_ext={0x202ece
72, 0xfffffffffffffe05}, 0x2802, 0x3, 0x91, 0x6, 0x8, 0x0, 0x0, 0x0, 0x0, 0x0, 0
x14}, 0x0, 0xffffffffffffffff, 0xffffffffffffffff, 0x0) (rerun: 64)

program saved to repro.txt
```

2.2 将系统调用序列转换为C程序

```
1 ./bin/syz-prog2c -prog=repro.syz -os=linux -arch=amd64 -build > repro
2
3 ./bin/syz-prog2c -prog=repro.syz > generated_program.c
4
5 ./generated_program
```

`-build`：如果设置为 `true`，工具将在生成 C 源代码后尝试编译它。

`-threaded`：生成多线程程序。

`-repeat`：指定程序重复执行的次数（`<=0` 表示无限重复）。

`-procs`：并行进程的数量。

`-slowdown`：模拟执行时的慢速（例如，由于仿真或插装引起的执行减慢）。

`-prog`：指定要转换的 Syzkaller 程序文件。

```
x68\x76\x86\x5d\x89\x5b\x25\x82\xef\x04\x9d\xf3\x9c\xb3\xb0\xe6\x66\x4a\x7b\xdf\x2b\x3f\x69\xc6\x04\xa9\x8b\xb9\xf7\x34\xde\x33\x47\x93\x67\x09\xa2\x7d\x8a\x34\xc5\x89\xbd\x72\x6e\xaf\xee\x7e\x13\x8f\x4a\x6c\xc6\x89\xf6\xff\x69\x6f\x79\xc0\xda\x26\x73\xf2\xa0\xb1\x31\x29\xea\xee\x1e\xb0\x8e\xae\xf1\x44\xcd\x15\xb6\x78\xa5\xde\xb5\x14\x1d\x23\x1d\xbb\x99\xfb\xbd\x35\x95\x8e\x5c\xf8\xb2", 290);
...
        syscall(__NR_write, /*fd=*/r[0], /*buffer=*/0x20000140ul, /*count=*/8ul)
;
        for (int i = 0; i < 64; i++) {
            syscall(__NR_write, /*fd=*/r[0], /*buffer=*/0x20000140ul, /*count=*/8ul)
;
        }
        return 0;
}
binary build OK
(base) qjj@syzkaller109:~/go1.22.1_projects/go_projects/syzkaller$
```

3. 利用syz-execprog检验可复现程序的有效性

syzkaller官方文档: [reproducing_crashes.md](#)

以下以1220_6.13rc_KASAN中的KASAN_slab-out-of-bounds Write in snd_seq_oss_synth_sysex为例进行说明:

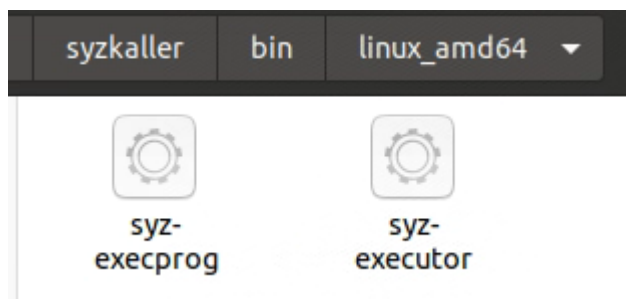
注意reproducing_crashes.md 中复现的是POC中列出的crash，以下均基于Syz Reproducer实现本地发现的crash。

3.1 编译syz-execprog, syz-executor, syz-exec2prog

在syzkaller路径下运行:

```
1 make execprog
2 make executor
3 make prog2c
```

编译结果存放在syzkaller/bin/linux_amd64中，注意该路径



3.2 确定内核版本并编译

- 1 本地一般已经具有编译后的内核，记录路径即可

~~Kernel Commit: 导致该crash的内核提交版本。~~

```
1 # 6.13rc3 2024-12-15
2 https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/log/?
   id=78d4f34e2115b517bcbfe7ec0d018bbbbc6f9b0b8
```

~~Kernel Config: 用于构建内核的配置文件。此例选择本地的6.13rc3所采用配置文件~~

3.3 启动虚拟机

- 1 需要一个可启动磁盘镜像。Syzbot目前使用基于Buildroot的小型镜像，可以在本地构建或下载.NET镜像。
- 2 本地的crash可以直接使用syzkaller的启动镜像。

Disk Image启动命令：

```
1 # 用debug指令获取镜像的启动命令
2 ./bin/syz-manager -config=my613.cfg -debug
```

```
1 qemu-system-x86_64 \
2   -m 2048 \
3   -smp 4 \
4   -chardev socket,id=SOCKSYZ,server=on,wait=off,host=localhost,port=58191 \
5   -mon chardev=SOCKSYZ,mode=control \
6   -display none \
7   -serial stdio \
8   -no-reboot \
9   -name VM-0 \
10  -device virtio-rng-pci \
11  -enable-kvm \
12  -cpu host,migratable=off \
```

```
13 -device e1000,netdev=net0 \  
14 -netdev user,id=net0,restrict=on,hostfwd=tcp:127.0.0.1:64317-:22 \  
15 -hda /home/qjj/go1.22.1_projects/go_projects/image/bullseye.img \  
16 -snapshot \  
17 -kernel  
    /home/qjj/go1.22.1_projects/go_projects/Newlinux/arch/x86/boot/bzImage \  
18 -append "root=/dev/sda console=ttyS0"
```

3.4 修改虚拟机配置

1. 新增密码

```
1 passwd
```

2. 允许外部访问

打开config文件：

```
1 nano /etc/ssh/sshd_config
```

在文件中新增以下字段：

```
1 PermitRootLogin yes  
2 PasswordAuthentication yes
```

重启服务

```
1 systemctl restart ssh  
2 systemctl status ssh
```

3.5 ~~准备Syz Reproducer：准备用于复现 crash 的程序。~~


```
1 # 复现syzbot报告的crash时用如下指令获取复现程序
2 wget -O 'repro.c' 'https://syzkaller.appspot.com/x/repro.c?x=177beac0580000'
3 # 复现本地发现的crash时则直接将对应复现程序转移至syzkaller目录中
```

3.6 将 Reproducer 传输到虚拟机并运行：

使用 `scp` 将编译好的 `repro` 程序传输到虚拟机：使用 `scp` 命令将 syzkaller 的二进制文件（syzkaller 目录下 `bin/linux_amd64` 目录中的所有文件）和 Reproducer 程序（当前目录下的 `repro.syz` 文件）传输到远程虚拟机（通过 SSH 端口转发到本地的 127.0.0.1:64317），注意端口应与第三步的vm启动命令中内容相符。

```
1 #上传复现工具及复现程序到虚拟机中
2 scp -P 64317 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -o
  IdentitiesOnly=yes
  /home/qjj/go1.22.1_projects/go_projects/syzkaller/bin/linux_amd64/*
  ./repro.syz root@127.0.0.1:/root/
3
4 #执行复现程序
5 ssh -p 64317 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -o
  IdentitiesOnly=yes root@127.0.0.1 './syz-execprog -enable=all -repeat=0 -
  procs=8 -cover=0 repro.syz'
```

```
1 export SYZKALLER_PATH=~/.syzkaller"
2 scp -P 64317 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -o
  IdentitiesOnly=yes $SYZKALLER_PATH/bin/linux_amd64/* ./repro.syz
  root@127.0.0.1:/root/
3
4 # 以下是绝对路径
5 scp -P 64317 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -o
  IdentitiesOnly=yes
  /home/qjj/go1.22.1_projects/go_projects/syzkaller/bin/linux_amd64/*
  ./repro.syz root@127.0.0.1:/root/
```

```
qjj@syzkaller109: ~/go1.22.1_projects/go_projects/syzkaller
root@syzkaller:~# ls
-rwxr-xr-x 1 root root 10240 repro.syz syz-execprog syz-executor
root@syzkaller:~# cd repro.syz
-bash: cd: repro.syz: Not a directory
root@syzkaller:~# cat repro.syz
/*
 * {Threaded:true Repeat:true RepeatTimes:0 Procs:1 Slowdown:1 Sandbox: SandboxArgs:0 Leak:false NetInjection:false NetDevices:false NetReset:false Cgroups:false
 * BinfmtMisc:false CloseFDs:false KCSAN:false DevlinkPCI:false NicVF:false USB:false
 * VhciInjection:false Wifi:false IEEE802154:false Sysctl:false Swap:false UseTmpDir:false HandleSegv:false Trace:false LegacyOptions:{Collide:false Fault:false
 * FaultCall:0 FaultNth:0}}
 */
r0 = openat$sequencer2(0xffffffffffffffff9c, &(0x7f0000000500), 0x1, 0x0)
write$sequencer(r0, &(0x7f0000000540)=ANY=[@ANYBLOB="94043954"], 0x1d) (async, rerun: 64)
write$sequencer(r0, &(0x7f0000000140)=ANY=[@ANYBLOB="9404ab3f86f5dad65cf389f7d6f3d789ad312df7e83633f4a05f37936b28cfc9937e461ff1fce0267e6e92a7d903cc2a4bae114ce018707ca205006a1246faadebbe3b1a1839e7e3ee9169295f233aa2dbacf2b7bbde120acc5e471a91bf17004232a20641fa8f6984a746788c25e8a4e6ec2c2e75c5f0c02000000000000000da92eb074de0ce3750cb7c138df4703269cb42763969b01d9a196f449d34fddbc44ae4291e6ca0241360fca1579dce30b8972206cd30016baf961dd80595df83b96876865d895b2582ef049df39cb3b0e6664a7bdf2b3f69c604a98bb9f734de3347936709a27d8a34c589bd726eafee7e138f4a6cc689f6ff696f79c0da2673f2a0b13129eaaee1eb08eaef144cd15b678a5deb5141d231dbb99fbbd35958e5cf8b2"], 0x8) (rerun: 64)
root@syzkaller:~#
```

然后通过 `ssh` 或者虚拟机中手动执行程序：

```
1 ssh -p 10022 -o UserKnownHostsFile=/dev/null \      -o StrictHostKeyChecking=no
  \      -o IdentitiesOnly=yes root@127.0.0.1 'chmod +x ./repro && ./repro'
2
3 # 在虚拟机中给予执行权限
4 chmod +x /root/repro.syz
5
6 # 在另一个终端中运行syz-execprog，推荐采用此项目，否则不便观察当前的运行次数
7 ssh -p 64317 -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -o
  IdentitiesOnly=yes root@127.0.0.1 './syz-execprog -enable=all -repeat=0 -
  procs=6 -cover=0 repro.syz'
8
9 # 在虚拟机中手动运行yz-execprog
10 sudo ./syz-execprog -executor=./syz-executor -repeat=16 -procs=16 -cover=0
  repro.syz
```

3.7 `syz-execprog` 的参数说明：

- 1 `-executor string`: 指定执行程序的路径（你已经使用了 `-executor=./syz-executor`，这应该是正确的）。

- 2 -repeat int: 指定重复执行次数（你传递了 -repeat=0，表示无限次执行）。
- 3 -procs int: 指定并行进程数（你传递了 -procs=16）。
- 4 -cover int: 是否收集覆盖率信息（你传递了 -cover=0，表示不收集）。
- 5 -sandbox string: 指定沙箱模式（默认为 none，可以选择 setuid、namespace 或 android 等）。

成功开始执行后如图所示，左侧终端为虚拟机的串行启动框，右侧为另一个终端框，通过ssh调用虚拟机中的syz-executor进行复现。

```
[ 0] type 2 family 0 port 6081 - 0
[ 810.895797][ T13] netdevsim netdevsim3 netdevsim0 (unregistering): unset [1 s.
[ 0] type 2 family 0 port 6081 - 0
[ 811.132277][ T13] bond0 (unregistering): (slave bond_slave_0): Releasing ba
ckup interface
[ 811.143376][ T13] bond0 (unregistering): (slave bond_slave_1): Releasing ba
ckup interface
[ 811.161510][ T13] bond0 (unregistering): Released all slaves
[ 811.332397][ T13] veth1_macvtap: left promiscuous mode
[ 811.334656][ T13] veth0_macvtap: left promiscuous mode
[ 811.336670][ T13] veth1_vlan: left promiscuous mode
[ 811.965034][ T13] team0 (unregistering): Port device team_slave_1 removed
[ 812.018039][ T13] team0 (unregistering): Port device team_slave_0 removed
[ 813.388786][T47836] bond0: (slave bond_slave_0): Enslaving as an active inter
face with an up link
[ 813.400051][T47836] bond0: (slave bond_slave_1): Enslaving as an active inter
face with an up link
[ 813.488056][T47836] team0: Port device team_slave_0 added
[ 813.495253][T47836] team0: Port device team_slave_1 added
[ 814.360118][T47836] netdevsim netdevsim3 netdevsim0: renamed from eth0
[ 814.371313][T47836] netdevsim netdevsim3 netdevsim1: renamed from eth1
[ 814.386434][T47836] netdevsim netdevsim3 netdevsim2: renamed from eth2
[ 814.395861][T47836] netdevsim netdevsim3 netdevsim3: renamed from eth3

Warning: Permanently added '[127.0.0.1]:64317' s.
root@127.0.0.1's password:
2024/12/30 09:39:56 parsed 1 programs
2024/12/30 09:40:12 executed programs: 0
2024/12/30 09:40:23 executed programs: 12
2024/12/30 09:40:28 executed programs: 619
2024/12/30 09:40:33 executed programs: 1398
2024/12/30 09:40:38 executed programs: 2189
2024/12/30 09:40:43 executed programs: 2985
2024/12/30 09:40:48 executed programs: 3749
2024/12/30 09:40:53 executed programs: 4398
2024/12/30 09:40:59 executed programs: 5062
2024/12/30 09:41:04 executed programs: 5703
2024/12/30 09:41:09 executed programs: 6376
2024/12/30 09:41:14 executed programs: 7026
2024/12/30 09:41:19 executed programs: 7693
2024/12/30 09:41:24 executed programs: 8333
2024/12/30 09:41:29 executed programs: 8985
2024/12/30 09:41:34 executed programs: 9622
2024/12/30 09:41:39 executed programs: 10297
2024/12/30 09:41:44 executed programs: 10951
2024/12/30 09:41:49 executed programs: 11599
2024/12/30 09:41:54 executed programs: 12250
```

末端跳转