执行bpf/test_flow_dissector.sh用例会导致物理机crash重启故障分析

一、环境信息

```
[root@localhost bpf]# cat /etc/FusionOS-latest
FusionOSversion=FusionOS-23_23.1.2.B002_x86_64
compiletime=2024-05-24-01-57-36
gccversion=10.3.1-28.u12.fos23
kernelversion=5.10.0-136.75.0.155.u112.fos23
openjdkversion=1.8.0.402.b06-0.u1.fos23
```

二、测试步骤

```
# 安装 linux source 编译依赖包
$ yum install -y jq kernel-devel binutils-devel readline-devel traceroute patch
rpm-build bc rsync llvm libcap-ng-devel fuse-devel popt-devel libcap-devel clang
net-tools numactl-devel elfutils-devel libhugetlbfs-devel python3-docutils
openssl-devel make python bpftool nc libmnl-devel libubsan-static libasan-static
socat iperf3 ipvsadm conntrack-tools bison flex
# 安装 kernel-source软件包并配置
$ dnf install -y kernel-source
$ setenforce 0
$ ulimit -l unlimited
$ systemctl stop firewalld
$ timedatectl set-ntp false
$ sysctl -w net.ipv4.conf.all.rp_filter=0
$ sysctl -w net.ipv4.conf.default.rp_filter=0
# 执行测试
$ cd /usr/src/linux-5.10.0-
136.75.0.155.u112.fos23.x86_64/tools/testing/selftests/bpf
$ make
$ sh ./test_flow_dissector.sh
```

三、执行状态

执行测试用例后系统卡死，之后便复位重启

四、分析定位

（1）定位测试用例执行失败的命令行

```
$ sh -x ./test_flow_dissector.sh
..............
+ tc filter del dev lo ingress pref 1337
Error: Cannot find specified filter chain.
We have an error talking to the kernel
+ echo 'Testing port range...'
Testing port range...
+ tc filter add dev lo parent ffff: protocol ip pref 1337 flower ip_proto udp
src_port 8-10 action drop
...系统开始复位...
```

分析测试脚本，提取失败命令

```
# 在本地回环接口（lo）上添加一个 ingress（入口）队列调度器
tc qdisc add dev lo ingress
# 在本地回环接口（lo）上添加一个过滤器规则
tc filter add dev lo parent ffff: protocol ip pref 1337 flower ip_proto udp
src_port 8-10 action drop
```

（2）分析 kernel 的 vmcore 文件

```
# 下载 kernel-debuginfo 与 crash
$ dnf install -y kernel-debuginfo
# 拷贝 vmlinux 到 vmcore 同级目录下
$ cp /usr/lib/debug/usr/lib/modules/"$(uname -r)"/vmlinux ./
```

（3）使用 crash 解析 vmcore，并执行分析

```
$ crash vmlinux vmcore
$ bt
PID: 2104     TASK: ff2812ae123a0000  CPU: 14   COMMAND: "tc"
 #0 [ff302b01808e7788] panic at ffffffffb1c62cce
 #1 [ff302b01808e7828] no_context at ffffffffb127d79c
 #2 [ff302b01808e7860] __bad_area_nosemaphore at ffffffffb127d8a2
 #3 [ff302b01808e78a8] exc_page_fault at ffffffffb1cad5bd
 #4 [ff302b01808e7900] asm_exc_page_fault at ffffffffb1e00afe
    [exception RIP: fl_init+142]
    RIP: ffffffffc09a574e  RSP: ff302b01808e79b0  RFLAGS: 00010286
    RAX: 0200000400000000  RBX: 00000000c09af0e0  RCX: 0000000000000000
    RDX: 0000000000000200  RSI: ffffffffc09ac140  RDI: ff2812ae10427200
    RBP: ff302b01808e7ae0   R8: 0000000000000200   R9: ff2812ae10427200
    R10: ff2812ae61080200  R11: 0000000000000000  R12: ff2812ae05c88400
    R13: 0000000005390000  R14: 0000000000000008  R15: ff2812ae05c88800
    ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
 #5 [ff302b01808e79b0] tcf_proto_create at ffffffffb1b15122
 #6 [ff302b01808e79f0] tc_new_tfilter at ffffffffb1b16771
 #7 [ff302b01808e7b28] rtnetlink_rcv_msg at ffffffffb1abd3c0
 #8 [ff302b01808e7ba8] netlink_rcv_skb at ffffffffb1b2335e
 #9 [ff302b01808e7c10] netlink_unicast at ffffffffb1b2292a
#10 [ff302b01808e7c50] netlink_sendmsg at ffffffffb1b22da6
#11 [ff302b01808e7cc8] sock_sendmsg at ffffffffb1a7f38f
#12 [ff302b01808e7ce0] ____sys_sendmsg at ffffffffb1a7f702
#13 [ff302b01808e7d58] ___sys_sendmsg at ffffffffb1a81d85
#14 [ff302b01808e7ea8] __sys_sendmsg at ffffffffb1a81e59
#15 [ff302b01808e7f38] do_syscall_64 at ffffffffb1ca9e5d
```

```
#16 [ff302b01808e7f50] entry_SYSCALL_64_after_hwframe at ffffffffb1e00099
    RIP: 00007f544a2c8e27  RSP: 00007ffec63ea438  RFLAGS: 00000246
    RAX: ffffffffffffffda  RBX: 0000000000000000  RCX: 00007f544a2c8e27
    RDX: 0000000000000000  RSI: 00007ffec63ea4b0  RDI: 0000000000000003
    RBP: 0000000000000001   R8: 0000000000000001   R9: 00007ffec63ea3ec
    R10: 000000000000000c  R11: 0000000000000246  R12: 00000000665ec612
    R13: 00007ffec63ea5a0  R14: 0000000000000008  R15: 000055728828df00
    ORIG_RAX: 000000000000002e  CS: 0033  SS: 002b
```

`dis -l ffffffffb1b15122` 查看在 `tcf_proto_create` 函数执行错误点，进而确定失败函数

```
$ crash> dis -l ffffffffb1b15122
/usr/src/debug/kernel-5.10.0-136.75.0.155.u112.fos23.x86_64/linux-5.10.0-
136.75.0.155.u112.fos23.x86_64/net/sched/cls_api.c: 275
0xffffffffb1b15122 <tcf_proto_create+162>:      nopl   (%rax)
```

`dis -s tcf_proto_create` 查看函数内容信息，结合 kernel 内核源码，确定 RIP 指向的失败函数。

```
static int fl_init(struct tcf_proto *tp)
{
        struct cls_fl_head *head;

        head = kzalloc(sizeof(*head), GFP_KERNEL);
        if (!head)
                return -ENOBUFS;

        spin_lock_init(&head->masks_lock);
        INIT_LIST_HEAD_RCU(&head->masks);
        INIT_LIST_HEAD(&head->hw_filters);
        rcu_assign_pointer(tp->root, head);
        idr_init(&head->handle_idr);

        tmplt_reoffload = &fl_tmplt_reoffload;              <------ 问题代码行

        return rhashtable_init(&head->ht, &mask_ht_params);
}
```

使用 `dis -l fl_init+142` 确定失败的代码行信息。这段信息表示将 `0xffffffffc09a3f10` 加载到相对于RIP寄存器的偏移量为 `-0xe690cd9` 的内存位置中。最终保存在 `0xffffffffb2314a80` 的 `tmplt_reoffload` 变量中。

```
$ crash> dis -l fl_init+142
0xffffffffc09a574e <fl_init+142>:       movq
$0xffffffffc09a3f10,-0xe690cd9(%rip)        # 0xffffffffb2314a80
<tmplt_reoffload>
```

`tmplt_reoffload = &fl_tmplt_reoffload;` 代码功能是对 `tmplt_reoffload` 外部函数指针变量进行赋值操作。查看 `fl_tmplt_reoffload` 没有任何问题，查看导出的 `tmplt_reoffload` 函数指针，暂时也没有新的发现。于是通过 git 信息确定涉及此处修改的 MR。

https://gitee.com/openeuler/kernel/commit/fca6540756e27c7d862b47f0e41c83dc8c7e600d

通过分析该MR涉及代码发现，在 tmplt_reoffload 指针声明阶段存在问题。此处使用了 `const` 声明，导致此处指针变量为常数。导致 `tmplt_reoffload = &fl_tmplt_reoffload;` 执行失败。

```
void (* const tmplt_reoffload)(struct tcf_chain *chain, bool add,
                               flow_setup_cb_t *cb, void *cb_priv);
EXPORT_SYMBOL(tmplt_reoffload);
```

（4）确定问题 patch

定位到问题点，通过 git 回溯到涉及 MR 提交，追溯到相关提交代码。

https://gitee.com/openeuler/kernel/commit/2eaffd1683bc34849029fdecb0b6067bb11c18d2

https://gitee.com/openeuler/kernel/commit/fca6540756e27c7d862b47f0e41c83dc8c7e600d

通过代码回退，验证确实是由此引入的问题。

（5）此处检索上游社区，已经给出修改方案

https://gitee.com/openeuler/kernel/commit/876c381235efef6758766e2e2fb26d4fdf10b551