```
#1  0x00007fffb1cfcdcb in ice_sq_send_cmd (hw=0x17ff6a300, cq=0x17ff6bdc0, desc=0x7fffac8be0f0, buf=0x17e61f140, buf_size=6,
cd=0x0)
    at ../src-dpdk/drivers/net/ice/base/ice_controlq.c:1126
#2  0x00007fffb1d2cc84 in ice_sq_send_cmd_retry (hw=0x17ff6a300, cq=0x17ff6bdc0, desc=0x7fffac8be0f0, buf=0x17e61f140, buf_size=6,
cd=0x0)
    at ../src-dpdk/drivers/net/ice/base/ice_common.c:1729
#3  0x00007fffb1d3ae4f in ice_aq_send_cmd (hw=0x17ff6a300, desc=0x7fffac8be0f0, buf=0x17e61f140, buf_size=6, cd=0x0)
    at ../src-dpdk/drivers/net/ice/base/ice_common.c:1788
#4  0x00007fffb1d3b8f8 in ice_aq_alloc_free_res (hw=0x17ff6a300, num_entries=1, buf=0x17e61f140, buf_size=6,
opc=ice_aqc_opc_alloc_res, cd=0x0)
    at ../src-dpdk/drivers/net/ice/base/ice_common.c:2124
#5  0x00007fffb1d3ba1d in ice_alloc_hw_res (hw=0x17ff6a300, type=96, num=1, btm=false, res=0x7fffac8c0602)
    at ../src-dpdk/drivers/net/ice/base/ice_common.c:2154
#6  0x00007fffb1ee34e7 in ice_alloc_prof_id (hw=0x17ff6a300, blk=ICE_BLK_RSS, prof_id=0x7fffac8c066a "\214\254")
    at ../src-dpdk/drivers/net/ice/base/ice_flex_pipe.c:3310
#7  0x00007fffb1f03f44 in ice_add_prof (hw=0x17ff6a300, blk=ICE_BLK_RSS, id=17179875328, ptypes=0x17e62003c "", attr=0x0,
attr_cnt=0, es=0x17e61ff10,
    masks=0x17e61ffda, fd_swap=true) at ../src-dpdk/drivers/net/ice/base/ice_flex_pipe.c:5029
#8  0x00007fffb1f20835 in ice_flow_add_prof_sync (hw=0x17ff6a300, blk=ICE_BLK_RSS, dir=ICE_FLOW_RX, prof_id=17179875328,
segs=0x17e620140, segs_cnt=1 '\001',
    acts=0x0, acts_cnt=0 '\000', prof=0x7fffac8c5178) at ../src-dpdk/drivers/net/ice/base/ice_flow.c:2245
#9  0x00007fffb1f216de in ice_flow_add_prof (hw=0x17ff6a300, blk=ICE_BLK_RSS, dir=ICE_FLOW_RX, prof_id=17179875328,
segs=0x17e620140, segs_cnt=1 '\001',
    acts=0x0, acts_cnt=0 '\000', prof=0x7fffac8c5178) at ../src-dpdk/drivers/net/ice/base/ice_flow.c:2646
#10 0x00007fffb1f4806e in ice_add_rss_cfg_sync (hw=0x17ff6a300, vsi_handle=0, cfg=0x7fffac8c51d0) at ../src-
dpdk/drivers/net/ice/base/ice_flow.c:4276
#11 0x00007fffb1f481e9 in ice_add_rss_cfg (hw=0x17ff6a300, vsi_handle=0, cfg=0x7fffac8c5270) at ../src-
dpdk/drivers/net/ice/base/ice_flow.c:4329
#12 0x00007fffb2112e17 in ice_add_rss_cfg_wrap (pf=0x17ff6d318, vsi_id=0, cfg=0x7fffac8c5270) at ../src-
dpdk/drivers/net/ice/ice_ethdev.c:2973
#13 0x00007fffb2112f6d in ice_rss_hash_set (pf=0x17ff6d318, rss_hf=12220) at ../src-dpdk/drivers/net/ice/ice_ethdev.c:3013
#14 0x00007fffb2128c75 in ice_init_rss (pf=0x17ff6d318) at ../src-dpdk/drivers/net/ice/ice_ethdev.c:3263
#15 0x00007fffb2128d8b in ice_dev_configure (dev=0x7fffb44e59c0 <rte_eth_devices>) at ../src-dpdk/drivers/net/ice/ice_ethdev.c:3292
#16 0x00007fffb061c888 in rte_eth_dev_configure (port_id=0, nb_rx_q=2, nb_tx_q=13, dev_conf=0x7fffac8caea8) at ../src-
dpdk/lib/ethdev/rte_ethdev.c:1622
#17 0x00007fffb3aa7a2d in dpdk_device_setup (xd=0x7fffbf5ee000) at /root/vcgnat/vpp_base/src/plugins/dpdk/device/common.c:191
#18 0x00007fffb3af2258 in dpdk_lib_init (dm=0x7fffb44d5438 <dpdk_main>) at /root/vcgnat/vpp_base/src/plugins/dpdk/device/init.c:482
#19 0x00007fffb3af0a04 in dpdk_process (vm=0x7fffb6880680, rt=0x7fffb9c6f6c0, f=0x0) at
/root/vcgnat/vpp_base/src/plugins/dpdk/device/init.c:1422
#20 0x00007ffff6e763ed in vlib_process_bootstrap (_a=140736125130936) at /root/vcgnat/vpp_base/src/vlib/main.c:1235
#21 0x00007ffff6d15c38 in clib_calljmp () at /root/vcgnat/vpp_base/src/vppinfra/longjmp.S:123
#22 0x00007fffcabad8b0 in ?? ()
```

```
#22 0x00007ffffaebea8b0 in ?? ()
#23 0x00007ffff6e75e0f in vlib_process_startup (vm=0x7fffb6880680, p=0x7fffb9c6f6c0, f=0x0) at
/root/vcgnat/vpp_base/src/vlib/main.c:1260
#24 0x00007ffff6e715aa in dispatch_process (vm=0x7fffb6880680, p=0x7fffb9c6f6c0, f=0x0, last_time_stamp=133363959581828)
    at /root/vcgnat/vpp_base/src/vlib/main.c:1316
#25 0x00007ffff6e71ea5 in vlib_main_or_worker_loop (vm=0x7fffb6880680, is_main=1) at /root/vcgnat/vpp_base/src/vlib/main.c:1515
#26 0x00007ffff6e7450a in vlib_main_loop (vm=0x7fffb6880680) at /root/vcgnat/vpp_base/src/vlib/main.c:1728
#27 0x00007ffff6e742f2 in vlib_main (vm=0x7fffb6880680, input=0x7fffaebedfa8) at /root/vcgnat/vpp_base/src/vlib/main.c:2017
#28 0x00007ffff6ed642e in thread0 (arg=140736255755904) at /root/vcgnat/vpp_base/src/vlib/unix/main.c:671
#29 0x00007ffff6d15c38 in clib_calljmp () at /root/vcgnat/vpp_base/src/vppinfra/longjmp.S:123
#30 0x00007fffffffcef0 in ?? ()
#31 0x00007ffff6ed5f5e in vlib_unix_main (argc=79, argv=0x4464f0) at /root/vcgnat/vpp_base/src/vlib/unix/main.c:751
#32 0x0000000000406b23 in main (argc=79, argv=0x4464f0) at /root/vcgnat/vpp_base/src/vpp/vnet/main.c:342
```

in frame #0, all parameters are zero.

by examining the source code, we found calling into frame #0 is just a wrapper.

```
enum ice_status
ice_sq_send_cmd(struct ice_hw *hw, struct ice_ctl_q_info *cq,
                struct ice_aq_desc *desc, void *buf, u16 buf_size,
                struct ice_sq_cd *cd)
{
        enum ice_status status = ICE_SUCCESS;

        /* if reset is in progress return a soft error */
        if (hw->reset_ongoing)
                return ICE_ERR_RESET_ONGOING;

        ice_acquire_lock(&cq->sq_lock);
        status = ice_sq_send_cmd_nolock(hw, cq, desc, buf, buf_size, cd);
        ice_release_lock(&cq->sq_lock);

        return status;
}
```

no explicit memory violation from source code can be observed.

examine the current instruction by inspecting the **RIP** register:

```
rax            0x17ff6a300         6441837312
rbx            0x17ffb2540         6442132800
rcx            0x17e61f140         6415315264
rdx            0x7fffac8be0f0      140736088236272
rsi            0x17ff6bdc0         6441844160
rdi            0x17ff6a300         6441837312
rbp            0x7fffac8b7230      0x7fffac8b7230
rsp            0x7fffac8abb20      0x7fffac8abb20
r8             0x6                 6
r9             0x0                 0
r10            0x7fffafb72e19      140736141405721
r11            0x7fffac8ac230      140736088162864
r12            0x405aa0            4217504
r13            0x7fffffffe400      140737488348160
r14            0x0                 0
r15            0x0                 0
rip            0x7fffb1cd922a      0x7fffb1cd922a <ice_sq_send_cmd_nolock+40>
eflags         0x13202             [ IF RF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
k0             0x0                 0
k1             0xffff              65535
k2             0x0                 0
k3             0x0                 0
k4             0x0                 0
k5             0x0                 0
k6             0x0                 0
k7             0x0                 0
```

disassemble the code around 0x7fffb1cd922a.

```
   0x00007fffb1cd9202 <+0>:     endbr64
```

```
   0x00007fffb1cd9206 <+4>:     push   %rbp
   0x00007fffb1cd9207 <+5>:     mov    %rsp,%rbp
   0x00007fffb1cd920a <+8>:     lea    -0xb000(%rsp),%r11
   0x00007fffb1cd9212 <+16>:    sub    $0x1000,%rsp
   0x00007fffb1cd9219 <+23>:    orq    $0x0,(%rsp)
   0x00007fffb1cd921e <+28>:    cmp    %r11,%rsp
   0x00007fffb1cd9221 <+31>:    jne    0x7fffb1cd9212 <ice_sq_send_cmd_nolock+16>
   0x00007fffb1cd9223 <+33>:    sub    $0x710,%rsp
=> 0x00007fffb1cd922a <+40>:    mov    %rdi,-0xb6e8(%rbp)
   0x00007fffb1cd9231 <+47>:    mov    %rsi,-0xb6f0(%rbp)
   0x00007fffb1cd9238 <+54>:    mov    %rdx,-0xb6f8(%rbp)
   0x00007fffb1cd923f <+61>:    mov    %rcx,-0xb700(%rbp)
   0x00007fffb1cd9246 <+68>:    mov    %r8d,%eax
   0x00007fffb1cd9249 <+71>:    mov    %r9,-0xb710(%rbp)
   0x00007fffb1cd9250 <+78>:    mov    %ax,-0xb704(%rbp)
   0x00007fffb1cd9257 <+85>:    movq   $0x0,-0xb6a8(%rbp)
   0x00007fffb1cd9262 <+96>:    movb   $0x0,-0xb6d1(%rbp)
   0x00007fffb1cd9269 <+103>:   movl   $0x0,-0xb6cc(%rbp)
   0x00007fffb1cd9273 <+113>:   movl   $0x0,-0xb6c8(%rbp)
   0x00007fffb1cd927d <+123>:   movw   $0x0,-0xb6d0(%rbp)
   0x00007fffb1cd9286 <+132>:   movl   $0x0,-0xb6c4(%rbp)
   0x00007fffb1cd9290 <+142>:   mov    -0xb6e8(%rbp),%rax
   0x00007fffb1cd9297 <+149>:   movzbl 0x18c9(%rax),%eax
```

answer is clear, indirect addressing memory **-0xb6e8(%rbp)** is the problem,
0x7fffac8b7230-0xb6e8=0x7FFFAC8ABB48
access to 0x7FFFAC8ABB48 raises page fault. however, the memory location is on stack, which further means **stack overflow** occurs in our case.

the potential solution could be increasing the stack size of calling stack, but how we do it?
from **clib_calljmp()** in **longjmp.S** we know the stack is initially set .

```
cdecl(clib_calljmp):
        /* Make sure stack is 16-byte aligned. */
        movq %rdx, %rax
        andq $0xf, %rax
        subq %rax, %rdx

        /* Get return address. */
        pop %rax
```

```
        pop %rax

        /* Switch to new stack. */
        xchgq %rsp, %rdx

        /* Save return address on new stack. */
        push %rax

        /* Save old stack pointer on new stack. */
        push %rdx

        /* Get function. */
        movq %rdi, %rdx

        /* Move argument into place. */
        movq %rsi, %rdi

        /* Away we go. */
        callq *%rdx
```

according to x86_64 system V calling convention (https://en.wikipedia.org/wiki/X86_calling_conventions#System_V_AMD64_ABI), %rdx is the 3rd parameter which passes the new stack address.

by examining the source code of vlib_process_startup, we can confirm the parameter of stack address.

```
/* Called in main stack. */
static_always_inline uword
vlib_process_startup (vlib_main_t * vm, vlib_process_t * p, vlib_frame_t * f)
{
  vlib_process_bootstrap_args_t a;
  uword r;

  a.vm = vm;
  a.process = p;
  a.frame = f;

  r = clib_setjmp (&p->return_longjmp, VLIB_PROCESS_RETURN_LONGJMP_RETURN);
  if (r == VLIB_PROCESS_RETURN_LONGJMP_RETURN)
    {
```

```
    {
      vlib_process_start_switch_stack (vm, p);
      r = clib_calljmp (vlib_process_bootstrap, pointer_to_uword (&a),
                        (void *) p->stack + (1 << p->log2_n_stack_bytes));
    }
  else
    vlib_process_finish_switch_stack (vm);

  return r;
}
```

by switching to frame #23, we know the **vlib_process_t * p** is the **dpdk_process**.

```
VLIB_REGISTER_NODE (dpdk_process_node,static) = {
    .function = dpdk_process,
    .type = VLIB_NODE_TYPE_PROCESS,
    .name = "dpdk-process",
    .process_log2_n_stack_bytes = 17,
};
```