BPF Debugging

Yonghong Song
Use Case

Problem: some tunnel remote IPv6 address not set up at line 448!
Let us examine whether due to map lookup failures.
Break At A Particular Place

- Check whether we hit line 431 and 445 if dst->dstv6 == <ADDR>.
else {
    cval = bpf_map_lookup_elem(&ctl_array, &v4_intf_pos);

    if (!cval)
        return TC_ACT_SHOT;
    ifindex = cval->ifindex;
    tkey.remote_ipv4 = dst->dst;
}

vip_num = vip_info->vip_num;
data_stats = bpf_map_lookup_elem(&stats, &vip_num);
if (!data_stats)
    return TC_ACT_SHOT;

data_stats->pkts++;
data_stats->bytes += pkt_bytes;
bpf_skb_set_tunnel_key(skb, &tkey, sizeof(tkey), tun_flag);

- Now suppose Line 445 is the issue.
- Possible actions:
  - print vip_num
  - dump stats map
High-Level Action

- Break at `<file>`:431 if `dst->dstv6 == <ADDR>`
- Break at `<file>`:445 if `dst->dstv6 == <ADDR>`
- Break at `<file>`:445 if `dst->dstv6 == <ADDR>` print vip_num, stats
Proposed Spec

bpftool prog inspect <PROG> <INSPECT_SPEC> <INSPECT_SPEC> ...

INSPECT_SPEC := <LOC> <ACTION> if <CONDITION>

LOC := <func_name>:<offset>

ACTION := [skip <n>] print {EXPR | reg(<REG>) | mem(<ADDR>, <SIZE>)},

CONDITION := EXPR | watch <ADDR>, <SIZE>, <TYPE>

Begin with debugging jited assembly only!

Extend to source base later (BTF support to map variables to registers/memory locations)
Debugging Command

bpftool prog inspect id 14
  ‘process_packet:230
  print “hit 1”
  if $r1 == 0 and mem($r9, 0) == <ADDR1> and
    mem($r9, 8) == <ADDR2>’

bpftool prog inspect id 14
  ‘process_packet:289
  print mem($r10 - 116, 8), stats
  if $r1 == 0 and mem($9, 0) == <ADDR1> and
    mem($r9, 8) == <ADDR2>’

int process_packet(...):
; static int process_packet(...)
  25: (bf) r6 = r4
  26: (bf) r7 = r3
  27: (bf) r8 = r2
  ....
  242: (bf) r2 = r10
; cval = bpf_map_lookup_elem(&ctl_array, &v6_intf_pos);
  243: (07) r2 += -88
  244: (18) r1 = map[id:184]
  ...
; if (!cval)
  255: (15) if r1 == 0x0 goto pc+79
; ifindex = cval->ifindex;

Source annotated codes should help find register/stack
correlation to local variables.
Kernel Support

- Construct condition and action from each \texttt{INSPECT\_SPEC} to a bpf kprobe prog
- The bpf prog is attached to jited code at specified location.
- The bpf prog share maps/globals, in read-only mode, with to-be-debugged bpf programs.
- Kernel support:
  - Nested kprobe and all other bpf programs
  - kprobe infra change to permit bpf jit addresses and bpf\_prog\_<TAG> lookup and permit nested kprobes.
- Debugging at bpf assembly level requires easy mapping of insn and registers from bpf xlated codes to jited codes, which is the case for x64.
Single Step Support

- QEMU
- Sleepable BPF program