tracing multi attachment

jiri olsa / isovalent @ cisco

tracing multi attachment

fast attachment to all traceable functions

trampoline runtime is faster

```
fentry          :   74.602 ± 0.474M/s
fexit           :   59.906 ± 0.300M/s
fmodret         :   62.760 ± 0.120M/s
kprobe          :   36.247 ± 0.057M/s
kprobe-multi    :   57.150 ± 0.076M/s
kretprobe       :   12.896 ± 0.046M/s
kretprobe-multi:   17.353 ± 0.061M/s
```

how did we get in here..

- based on old ftrace direct API
- static multi trampoline (Menglong Dong)
- based on new ftrace direct API

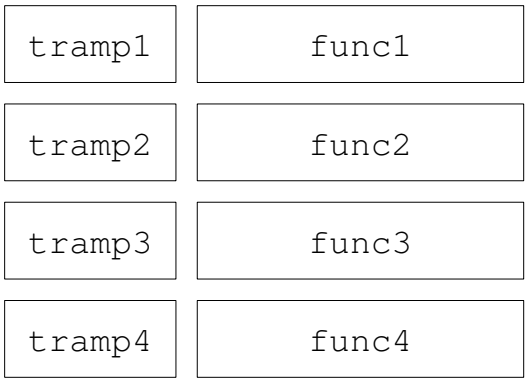
how did we get in here..

based on old ftrace direct API

static multi trampoline (Menglong Dong)

based on new ftrace direct API

```
register_ftrace_direct(tramp1.fops, tramp1.image)
register_ftrace_direct(tramp2.fops, tramp1.image)
register_ftrace_direct(tramp3.fops, tramp1.image)
register_ftrace_direct(tramp4.fops, tramp1.image)
```

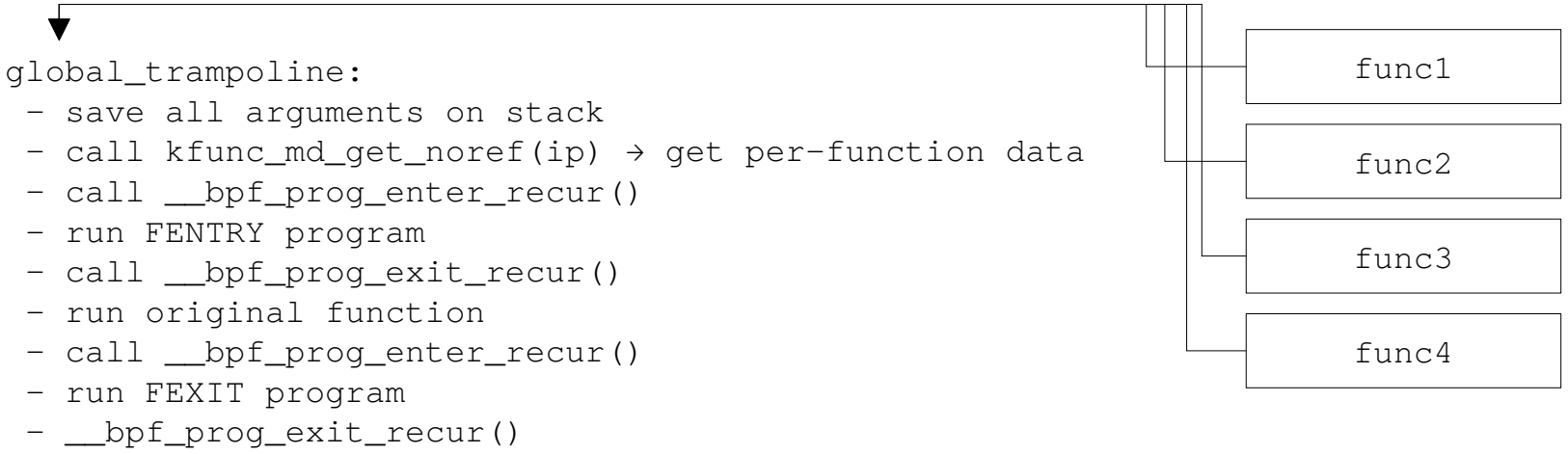


how did we get in here..

based on old ftrace direct API

static multi trampoline (Menglong Dong)

based on new ftrace direct API



how did we get in here..

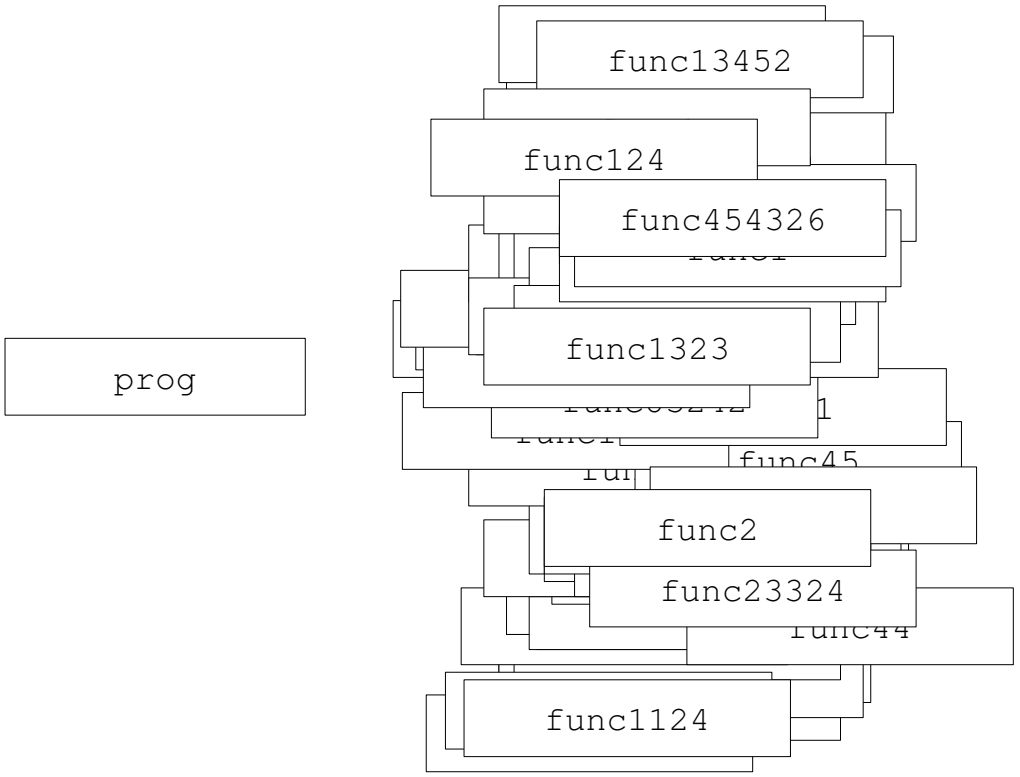
based on old ftrace direct API

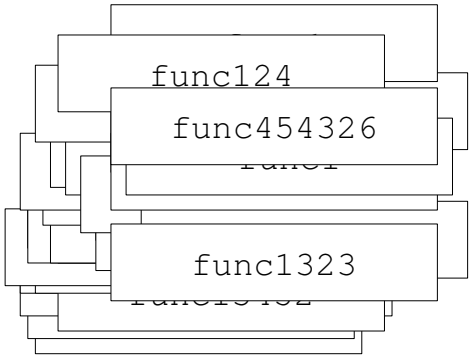
static multi trampoline (Menglong Dong)

based on new ftrace direct API

```
update_ftrace_direct_add(direct_ops, hash_reg)
update_ftrace_direct_mod(direct_ops, hash_mod)
update_ftrace_direct_del(direct_ops, hash_del)
```

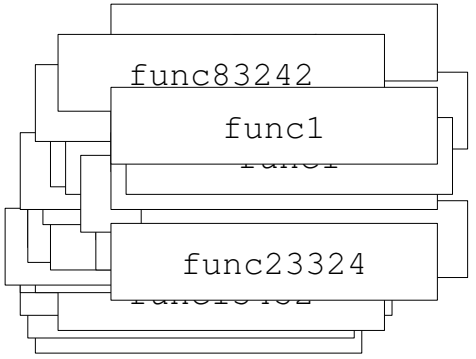
```
struct ftrace_func_entry {
        struct hlist_node hlist;
        unsigned long ip;
        unsigned long direct;
};
```


prog
func13452
func124
func454326
func1323
func45
func2
func23324
func1124

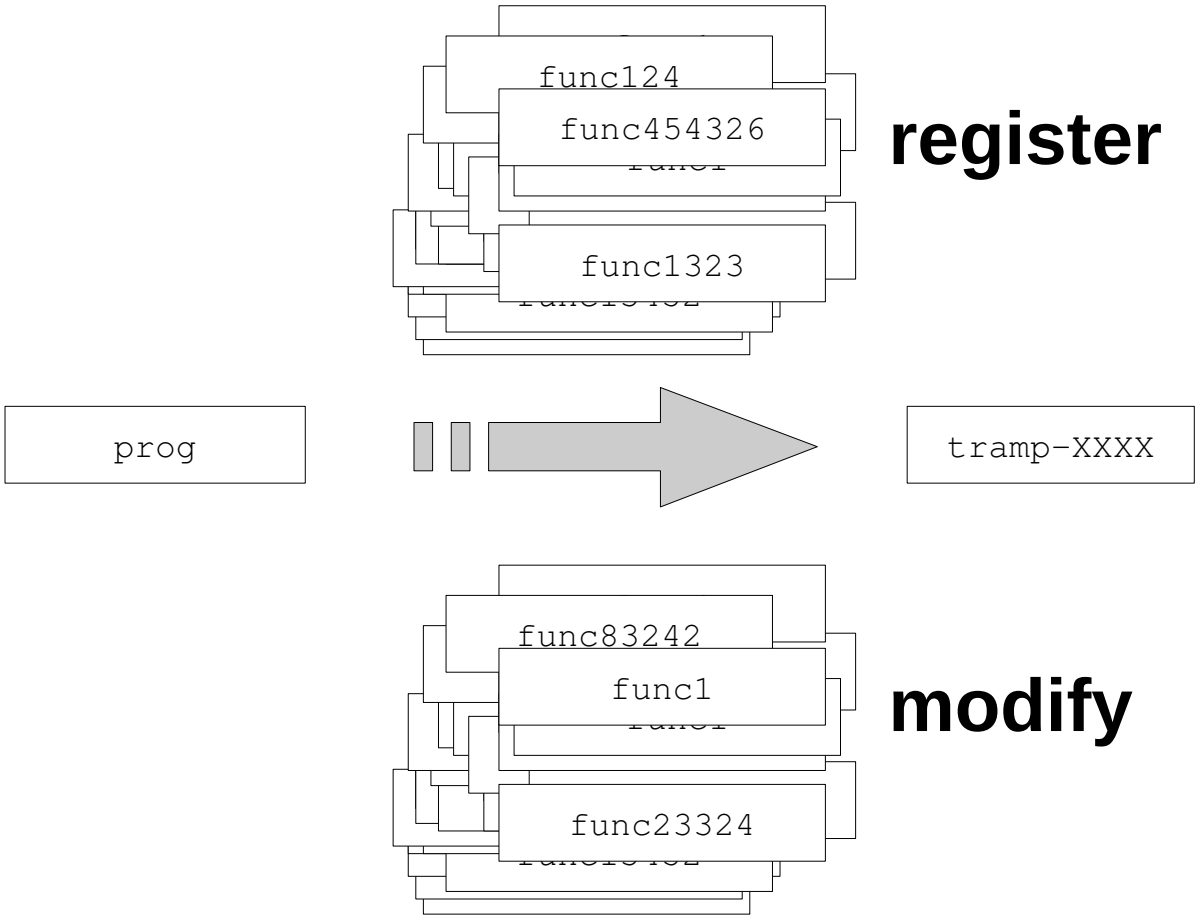


register

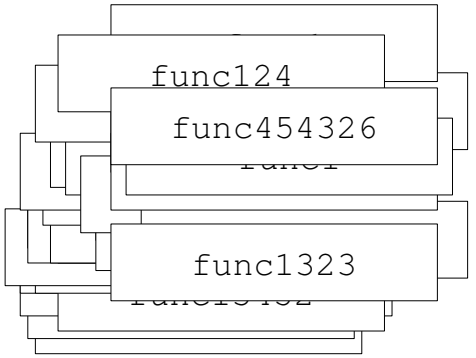
prog



modify



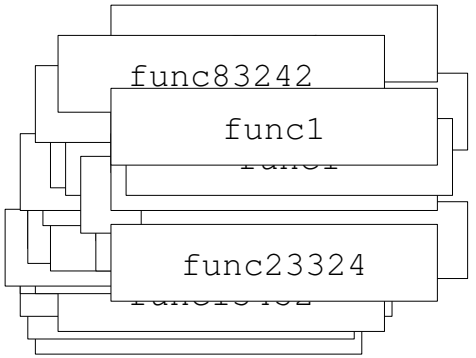

```
push   %rbp
mov    %rsp,%rbp
sub    $0x40,%rsp
mov    %rbx,-0x30(%rbp)
mov    $0x3,%eax
mov    %rax,-0x28(%rbp)
mov    %rdi,-0x20(%rbp)
mov    %rsi,-0x18(%rbp)
mov    %rdx,-0x10(%rbp)
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df00
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900001e0000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa001379a
lea    -0x20(%rbp),%rdi
call   0xffffffffa000f2d8
movabs $0xffffc900001e0000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
mov    -0x20(%rbp),%rdi
mov    -0x18(%rbp),%rsi
mov    -0x10(%rbp),%rdx
call   0xffffffff824e2b29
mov    %rax,-0x8(%rbp)
nopl   0x0(%rax,%rax,1)
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900002d5000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa00137f4
lea    -0x20(%rbp),%rdi
call   0xffffffffa0000bdc
movabs $0xffffc900002d5000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df40
mov    -0x8(%rbp),%rax
mov    -0x30(%rbp),%rbx
leave
ret
```



register

prog

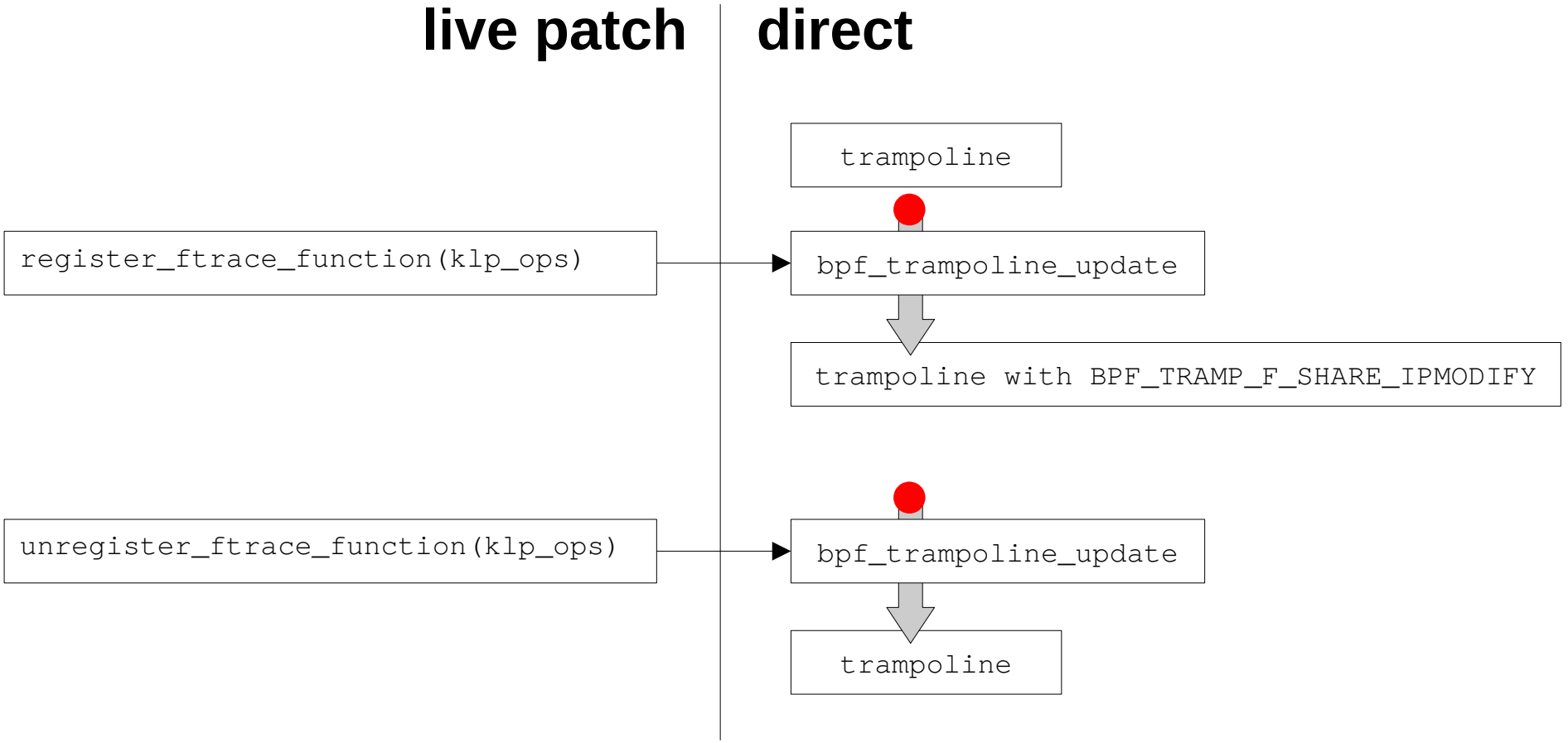
```
update_ftrace_direct_add(direct_ops, hash_reg)
update_ftrace_direct_mod(direct_ops, hash_mod)
```



modify

locking

- **each trampoline has lock**
- **what if I want to attach program 20k functions**
- **there’s 48 lockdep limit**
- **pool of 32 locks (Andrii)**
- **possible issue with live-patch**




```
if (!mutex_trylock(select_trampoline_lock(tr))) {
    /* sleep 1 ms to make sure whatever holding mutex makes
     * some progress.
     */
    msleep(1);
    return -EAGAIN;
}
```

```
struct { /* struct used by BPF_LINK_CREATE command */
  union {
    ...
    struct {
      __aligned_u64   ids;
      __aligned_u64   cookies;
      __u32           cnt;
    } tracing_multi;
  };
} link_create;
```

kernel

---

libbpf

```
struct bpf_link *
bpf_program__attach_tracing_multi(const struct bpf_program *prog,
                                  const char *pattern,
                                  const struct bpf_tracing_multi_opts *opts);

struct bpf_tracing_multi_opts {
  size_t sz;
  __u32 *ids;
  __u64 *cookies;
  size_t cnt;
  size_t :0;
};
```

module attachment

```
struct { /* anonymous struct used by BPF_PROG_LOAD command */
  union {
    ...
    /* or valid module BTF object fd or 0 to attach to vmlinux */
    __u32 attach_btf_obj_fd;
  };
};
```

kernel

---

libbpf

```
SEC("fentry.multi/bpf_testmod:bpf_testmod_fentry_test*")
int BPF_PROG(test_fentry)
{
  ...
```

```
link = bpf_program__attach_tracing_multi(skel->progs.test_fentry,
                  "bpf_testmod:bpf_testmod_fentry_test*", NULL);
```

supported program types

- **FENTRY / FEXIT / FSESSION**

```
BPF_TRACE_FENTRY_MULTI
BPF_TRACE_FEXIT_MULTI
BPF_TRACE_FSESSION_MULTI
```

not supported program types

- **FMOD_RET / LSM**

sleepable programs

attach cookies

arguments

single trampoline attach

- **program verified against specific function 1:1**

multi trampoline attach

- **program verified against stub function**
- **attached to multiple standard trampolines 1:N**

arguments

no direct access

```
bpf_get_func_arg()
bpf_get_func_ret()
bpf_get_func_arg_cnt()
```

can’t traverse arguments objects

but we can access arguments in fexit program

thanks, questions?

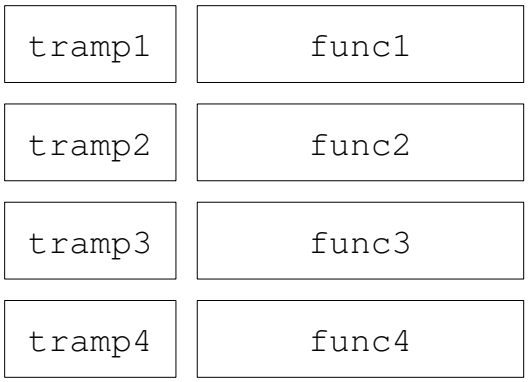
how did we get in here..

based on old ftrace direct API

static multi trampoline (Menglong Dong)

based on new ftrace direct API

```
register_ftrace_direct(tramp1.fops, tramp1.image)
register_ftrace_direct(tramp2.fops, tramp1.image)
register_ftrace_direct(tramp3.fops, tramp1.image)
register_ftrace_direct(tramp4.fops, tramp1.image)
```

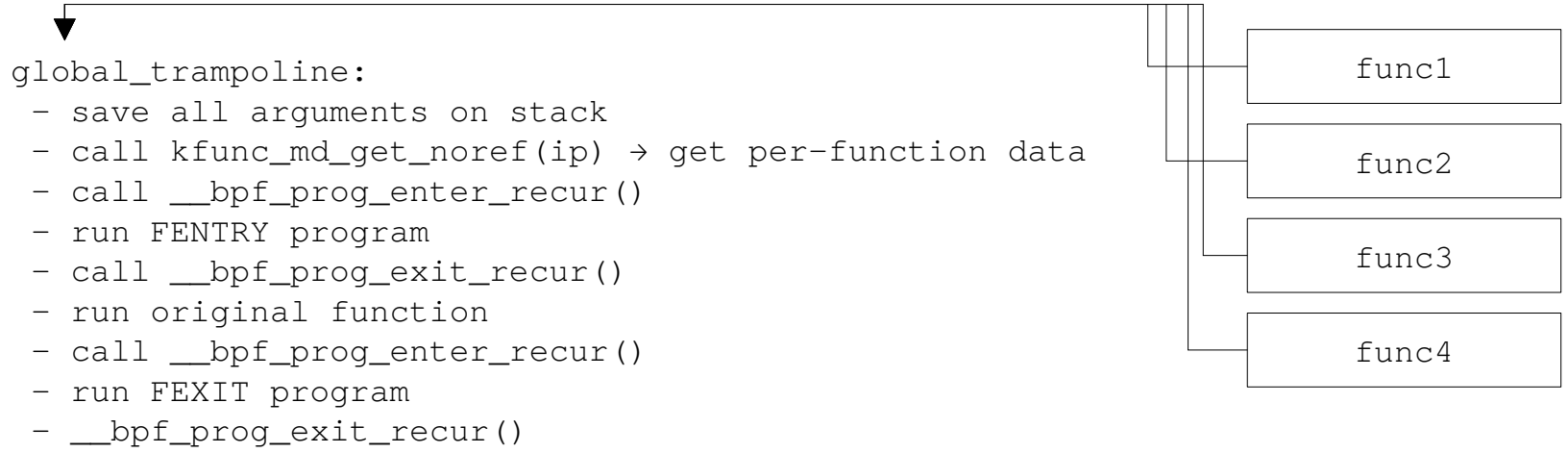


how did we get in here..

based on old ftrace direct API

static multi trampoline (Menglong Dong)

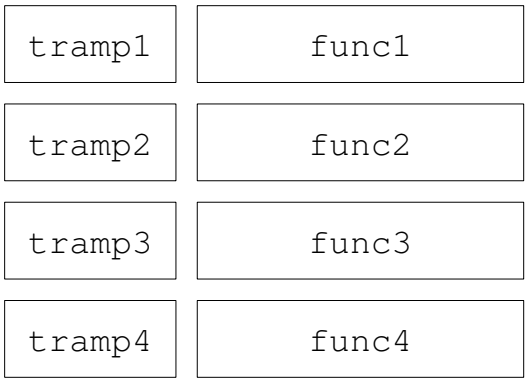
based on new ftrace direct API



how did we get in here..

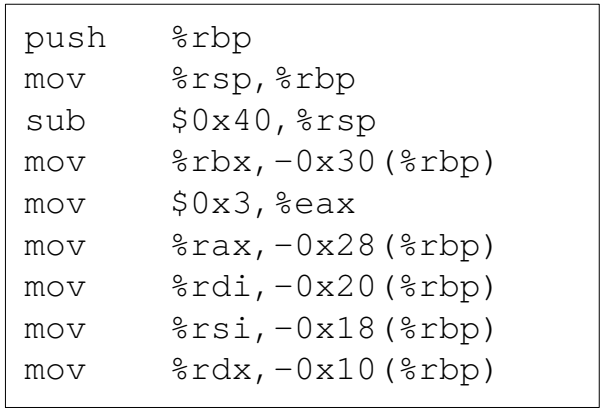
- based on old ftrace direct API
- static multi trampoline (Menglong Dong)
- based on new ftrace direct API

```
update_ftrace_direct_add(direct_ops, hash)
```



trampolines

- **generated for each function 1:1**
- **based on BTF**
- **execute fentry/fexit/fmod_ret/fsession**




```
push   %rbp
mov    %rsp,%rbp
sub    $0x40,%rsp
mov    %rbx,-0x30(%rbp)
mov    $0x3,%eax
mov    %rax,-0x28(%rbp)
mov    %rdi,-0x20(%rbp)
mov    %rsi,-0x18(%rbp)
mov    %rdx,-0x10(%rbp)
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df00
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900001e0000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa001379a
lea    -0x20(%rbp),%rdi
call   0xffffffffa000f2d8
movabs $0xffffc900001e0000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
mov    -0x20(%rbp),%rdi
mov    -0x18(%rbp),%rsi
mov    -0x10(%rbp),%rdx
call   0xffffffff824e2b29
mov    %rax,-0x8(%rbp)
nopl   0x0(%rax,%rax,1)
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900002d5000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa00137f4
lea    -0x20(%rbp),%rdi
call   0xffffffffa0000bdc
movabs $0xffffc900002d5000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df40
mov    -0x8(%rbp),%rax
mov    -0x30(%rbp),%rbx
leave
ret
```

```
movabs bpf_tramp_image,%rdi
call   __bpf_tramp_enter
```

```
push   %rbp
mov    %rsp,%rbp
sub    $0x40,%rsp
mov    %rbx,-0x30(%rbp)
mov    $0x3,%eax
mov    %rax,-0x28(%rbp)
mov    %rdi,-0x20(%rbp)
mov    %rsi,-0x18(%rbp)
mov    %rdx,-0x10(%rbp)
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df00
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900001e0000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa001379a
lea    -0x20(%rbp),%rdi
call   0xffffffffa000f2d8
movabs $0xffffc900001e0000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
mov    -0x20(%rbp),%rdi
mov    -0x18(%rbp),%rsi
mov    -0x10(%rbp),%rdx
call   0xffffffff824e2b29
mov    %rax,-0x8(%rbp)
nopl   0x0(%rax,%rax,1)
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900002d5000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa00137f4
lea    -0x20(%rbp),%rdi
call   0xffffffffa0000bdc
movabs $0xffffc900002d5000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df40
mov    -0x8(%rbp),%rax
mov    -0x30(%rbp),%rbx
leave
ret
```

```
xor     %edi,%edi
mov     %rdi,-0x40(%rbp)
movabs  $0xffffc900001e0000,%rdi
lea     -0x40(%rbp),%rsi
call    __bpf_prog_enter_recur
mov     %rax,%rbx
test    %rax,%rax
je      0xffffffffa001379a
```

```
push   %rbp
mov    %rsp,%rbp
sub    $0x40,%rsp
mov    %rbx,-0x30(%rbp)
mov    $0x3,%eax
mov    %rax,-0x28(%rbp)
mov    %rdi,-0x20(%rbp)
mov    %rsi,-0x18(%rbp)
mov    %rdx,-0x10(%rbp)
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df00
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900001e0000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa001379a
lea    -0x20(%rbp),%rdi
call   0xffffffffa000f2d8
movabs $0xffffc900001e0000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
mov    -0x20(%rbp),%rdi
mov    -0x18(%rbp),%rsi
mov    -0x10(%rbp),%rdx
call   0xffffffff824e2b29
mov    %rax,-0x8(%rbp)
nopl   0x0(%rax,%rax,1)
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900002d5000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa00137f4
lea    -0x20(%rbp),%rdi
call   0xffffffffa0000bdc
movabs $0xffffc900002d5000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df40
mov    -0x8(%rbp),%rax
mov    -0x30(%rbp),%rbx
leave
ret
```

```
lea     -0x20(%rbp),%rdi
call    fentry
```

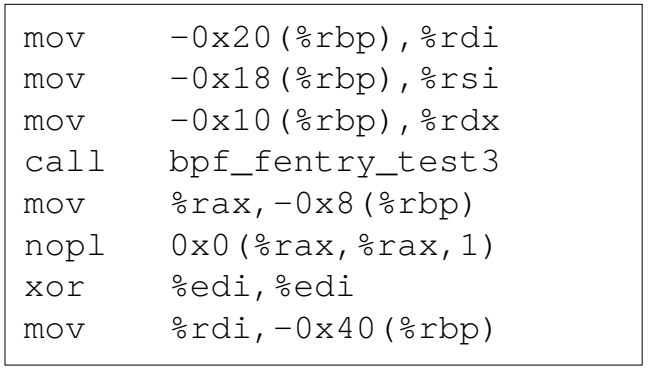
```
push   %rbp
mov    %rsp,%rbp
sub    $0x40,%rsp
mov    %rbx,-0x30(%rbp)
mov    $0x3,%eax
mov    %rax,-0x28(%rbp)
mov    %rdi,-0x20(%rbp)
mov    %rsi,-0x18(%rbp)
mov    %rdx,-0x10(%rbp)
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df00
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900001e0000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa001379a
lea    -0x20(%rbp),%rdi
call   0xffffffffa000f2d8
movabs $0xffffc900001e0000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
mov    -0x20(%rbp),%rdi
mov    -0x18(%rbp),%rsi
mov    -0x10(%rbp),%rdx
call   0xffffffff824e2b29
mov    %rax,-0x8(%rbp)
nopl   0x0(%rax,%rax,1)
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900002d5000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa00137f4
lea    -0x20(%rbp),%rdi
call   0xffffffffa0000bdc
movabs $0xffffc900002d5000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df40
mov    -0x8(%rbp),%rax
mov    -0x30(%rbp),%rbx
leave
ret
```

```
movabs $0xffffc900001e0000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   __bpf_prog_exit_recur
```

```
push   %rbp
mov    %rsp,%rbp
sub    $0x40,%rsp
mov    %rbx,-0x30(%rbp)
mov    $0x3,%eax
mov    %rax,-0x28(%rbp)
mov    %rdi,-0x20(%rbp)
mov    %rsi,-0x18(%rbp)
mov    %rdx,-0x10(%rbp)
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df00
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900001e0000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa001379a
lea    -0x20(%rbp),%rdi
call   0xffffffffa000f2d8
movabs $0xffffc900001e0000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
mov    -0x20(%rbp),%rdi
mov    -0x18(%rbp),%rsi
mov    -0x10(%rbp),%rdx
call   0xffffffff824e2b29
mov    %rax,-0x8(%rbp)
nopl   0x0(%rax,%rax,1)
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900002d5000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa00137f4
lea    -0x20(%rbp),%rdi
call   0xffffffffa0000bdc
movabs $0xffffc900002d5000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df40
mov    -0x8(%rbp),%rax
mov    -0x30(%rbp),%rbx
leave
ret
```

```
movabs $0xffffc900001e0000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   __bpf_prog_exit_recur
```

```
push   %rbp
mov    %rsp,%rbp
sub    $0x40,%rsp
mov    %rbx,-0x30(%rbp)
mov    $0x3,%eax
mov    %rax,-0x28(%rbp)
mov    %rdi,-0x20(%rbp)
mov    %rsi,-0x18(%rbp)
mov    %rdx,-0x10(%rbp)
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df00
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900001e0000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa001379a
lea    -0x20(%rbp),%rdi
call   0xffffffffa000f2d8
movabs $0xffffc900001e0000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
mov    -0x20(%rbp),%rdi
mov    -0x18(%rbp),%rsi
mov    -0x10(%rbp),%rdx
call   0xffffffff824e2b29
mov    %rax,-0x8(%rbp)
nopl   0x0(%rax,%rax,1)
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900002d5000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa00137f4
lea    -0x20(%rbp),%rdi
call   0xffffffffa0000bdc
movabs $0xffffc900002d5000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df40
mov    -0x8(%rbp),%rax
mov    -0x30(%rbp),%rbx
leave
ret
```




```
push   %rbp
mov    %rsp,%rbp
sub    $0x40,%rsp
mov    %rbx,-0x30(%rbp)
mov    $0x3,%eax
mov    %rax,-0x28(%rbp)
mov    %rdi,-0x20(%rbp)
mov    %rsi,-0x18(%rbp)
mov    %rdx,-0x10(%rbp)
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df00
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900001e0000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa001379a
lea    -0x20(%rbp),%rdi
call   0xffffffffa000f2d8
movabs $0xffffc900001e0000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
mov    -0x20(%rbp),%rdi
mov    -0x18(%rbp),%rsi
mov    -0x10(%rbp),%rdx
call   0xffffffff824e2b29
mov    %rax,-0x8(%rbp)
nopl   0x0(%rax,%rax,1)
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900002d5000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa00137f4
lea    -0x20(%rbp),%rdi
call   0xffffffffa0000bdc
movabs $0xffffc900002d5000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df40
mov    -0x8(%rbp),%rax
mov    -0x30(%rbp),%rbx
leave
ret
```

```
movabs $0xffffc900002d5000,%rdi
lea    -0x40(%rbp),%rsi
call   __bpf_prog_enter_recur
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa00137f4
```

```
push   %rbp
mov    %rsp,%rbp
sub    $0x40,%rsp
mov    %rbx,-0x30(%rbp)
mov    $0x3,%eax
mov    %rax,-0x28(%rbp)
mov    %rdi,-0x20(%rbp)
mov    %rsi,-0x18(%rbp)
mov    %rdx,-0x10(%rbp)
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df00
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900001e0000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa001379a
lea    -0x20(%rbp),%rdi
call   0xffffffffa000f2d8
movabs $0xffffc900001e0000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
mov    -0x20(%rbp),%rdi
mov    -0x18(%rbp),%rsi
mov    -0x10(%rbp),%rdx
call   0xffffffff824e2b29
mov    %rax,-0x8(%rbp)
nopl   0x0(%rax,%rax,1)
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900002d5000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa00137f4
lea    -0x20(%rbp),%rdi
call   0xffffffffa0000bdc
movabs $0xffffc900002d5000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df40
mov    -0x8(%rbp),%rax
mov    -0x30(%rbp),%rbx
leave
ret
```

```
lea    -0x20(%rbp),%rdi
call   fexit
```

```
push   %rbp
mov    %rsp,%rbp
sub    $0x40,%rsp
mov    %rbx,-0x30(%rbp)
mov    $0x3,%eax
mov    %rax,-0x28(%rbp)
mov    %rdi,-0x20(%rbp)
mov    %rsi,-0x18(%rbp)
mov    %rdx,-0x10(%rbp)
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df00
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900001e0000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa001379a
lea    -0x20(%rbp),%rdi
call   0xffffffffa000f2d8
movabs $0xffffc900001e0000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
mov    -0x20(%rbp),%rdi
mov    -0x18(%rbp),%rsi
mov    -0x10(%rbp),%rdx
call   0xffffffff824e2b29
mov    %rax,-0x8(%rbp)
nopl   0x0(%rax,%rax,1)
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900002d5000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa00137f4
lea    -0x20(%rbp),%rdi
call   0xffffffffa0000bdc
movabs $0xffffc900002d5000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df40
mov    -0x8(%rbp),%rax
mov    -0x30(%rbp),%rbx
leave
ret
```

```
movabs $0xffffc900002d5000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   __bpf_prog_exit_recur
```

```
push   %rbp
mov    %rsp,%rbp
sub    $0x40,%rsp
mov    %rbx,-0x30(%rbp)
mov    $0x3,%eax
mov    %rax,-0x28(%rbp)
mov    %rdi,-0x20(%rbp)
mov    %rsi,-0x18(%rbp)
mov    %rdx,-0x10(%rbp)
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df00
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900001e0000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa001379a
lea    -0x20(%rbp),%rdi
call   0xffffffffa000f2d8
movabs $0xffffc900001e0000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
mov    -0x20(%rbp),%rdi
mov    -0x18(%rbp),%rsi
mov    -0x10(%rbp),%rdx
call   0xffffffff824e2b29
mov    %rax,-0x8(%rbp)
nopl   0x0(%rax,%rax,1)
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900002d5000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa00137f4
lea    -0x20(%rbp),%rdi
call   0xffffffffa0000bdc
movabs $0xffffc900002d5000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df40
mov    -0x8(%rbp),%rax
mov    -0x30(%rbp),%rbx
leave
ret
```

```
movabs $0xffff88810b117000,%rdi
call   __bpf_tramp_exit
```

```
push   %rbp
mov    %rsp,%rbp
sub    $0x40,%rsp
mov    %rbx,-0x30(%rbp)
mov    $0x3,%eax
mov    %rax,-0x28(%rbp)
mov    %rdi,-0x20(%rbp)
mov    %rsi,-0x18(%rbp)
mov    %rdx,-0x10(%rbp)
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df00
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900001e0000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa001379a
lea    -0x20(%rbp),%rdi
call   0xffffffffa000f2d8
movabs $0xffffc900001e0000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
mov    -0x20(%rbp),%rdi
mov    -0x18(%rbp),%rsi
mov    -0x10(%rbp),%rdx
call   0xffffffff824e2b29
mov    %rax,-0x8(%rbp)
nopl   0x0(%rax,%rax,1)
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900002d5000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa00137f4
lea    -0x20(%rbp),%rdi
call   0xffffffffa0000bdc
movabs $0xffffc900002d5000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df40
mov    -0x8(%rbp),%rax
mov    -0x30(%rbp),%rbx
leave
ret
```

```
mov     -0x8(%rbp),%rax
mov     -0x30(%rbp),%rbx
leave
ret
```

```
push   %rbp
mov    %rsp,%rbp
sub    $0x40,%rsp
mov    %rbx,-0x30(%rbp)
mov    $0x3,%eax
mov    %rax,-0x28(%rbp)
mov    %rdi,-0x20(%rbp)
mov    %rsi,-0x18(%rbp)
mov    %rdx,-0x10(%rbp)
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df00
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900001e0000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa001379a
lea    -0x20(%rbp),%rdi
call   0xffffffffa000f2d8
movabs $0xffffc900001e0000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
mov    -0x20(%rbp),%rdi
mov    -0x18(%rbp),%rsi
mov    -0x10(%rbp),%rdx
call   0xffffffff824e2b29
mov    %rax,-0x8(%rbp)
nopl   0x0(%rax,%rax,1)
xor    %edi,%edi
mov    %rdi,-0x40(%rbp)
movabs $0xffffc900002d5000,%rdi
lea    -0x40(%rbp),%rsi
call   0xffffffff8166d5e0
mov    %rax,%rbx
test   %rax,%rax
je     0xffffffffa00137f4
lea    -0x20(%rbp),%rdi
call   0xffffffffa0000bdc
movabs $0xffffc900002d5000,%rdi
mov    %rbx,%rsi
lea    -0x40(%rbp),%rdx
call   0xffffffff8166d8a0
movabs $0xffff88810b117000,%rdi
call   0xffffffff8166df40
mov    -0x8(%rbp),%rax
mov    -0x30(%rbp),%rbx
leave
ret
```