GREBE: Unveiling Exploitation Potential for Linux Kernel Bugs

Zhenpeng Lin, Yueqi Chen, Yuhang Wu, Dongliang Mu, Chensheng Yu, Xinyu Xing, Kang li

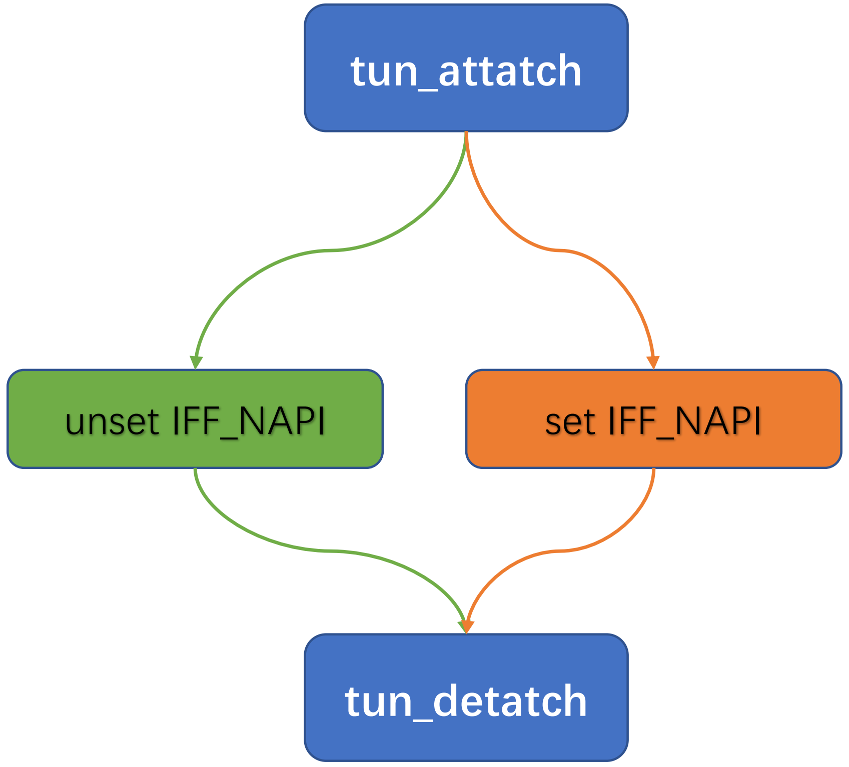
2022 IEEE Symposium on Security and Privacy (SP)

Background

```c
static void tun_attach(struct tun_struct *tun, ...)
{
    if (tun->flags & IFF_NAPI) {
        // initialize a timer
        hrtimer_init(&napi->timer, CLOCK_MONOTONIC,
            HRTIMER_MODE_REL_PINNED);
        // link current napi to the device's napi list
        list_add(&napi->dev_list, &dev->napi_list);
    }
}

static void tun_detach(struct tun_file *tfile, ...)
{
    struct tun_struct *tun = rtnl_dereference(tfile->tun);
    if (tun->flags & IFF_NAPI) {
        // GPF happens if timer is uninitialized
        hrtimer_cancel(&tfile->napi->timer);
        // remove the current napi from the list
        netif_napi_del(&tfile->napi);
    }
    destroy(tfile); // free napi
}

void free_netdev(struct net_device *dev) {
    list_for_each_entry_safe(p, n,
            &dev->napi_list, dev_list)
        netif_napi_del(p); // use-after-free
}
```

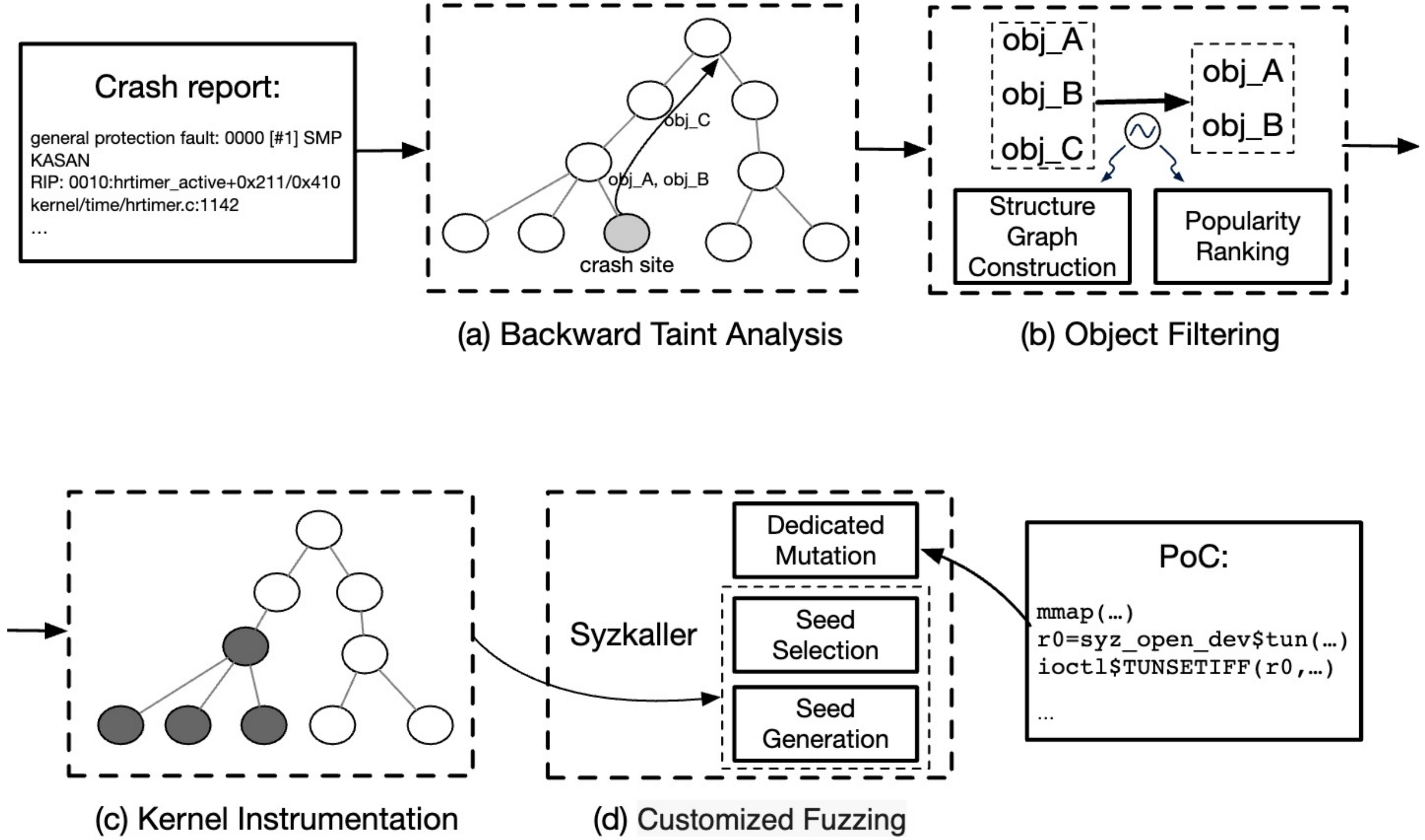


Why not directed fuzzing

- Identify the root cause of bug is challenge (hard to trigger)
- Only different path is not enough cannot (need vary the context)
- Syzkaller!
- No need to the root casuse
- Syscall sequense to vary the context

Kernel object fuzzing

- Inappropriate usage of objects
- Incorrect value involved in computation with a kernel object



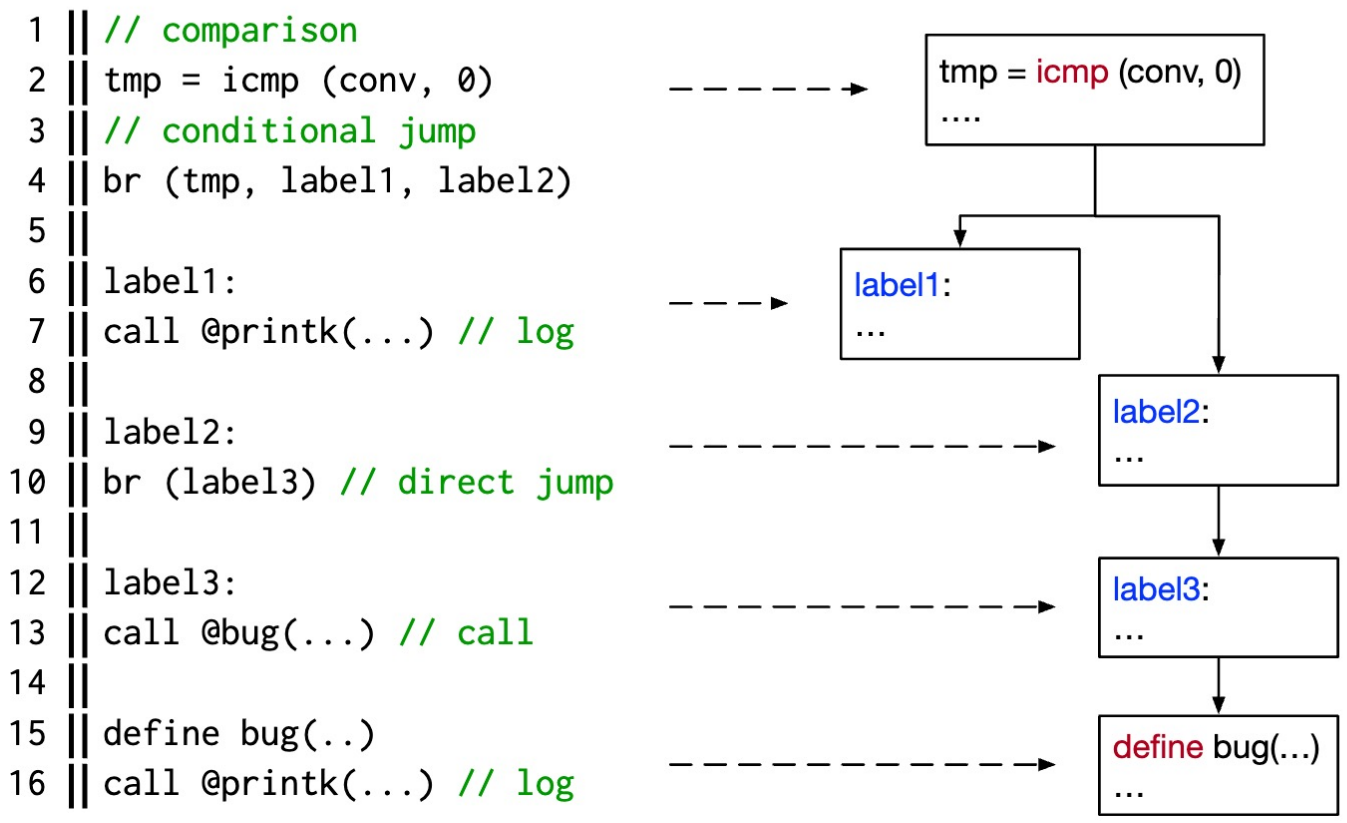
```
general protection fault: 0000 [#1] SMP
KASAN
RIP: 0010:hrtimer_active+0x211/0x410
kernel/time/hrtimer.c:1142
...
```

```
mmap(…)
r0=syz_open_dev$tun(…)
ioctl$TUNSETIFF(r0,…)
...
```

Critical struct identification

```
// in drivers/vhost/vhost.c
void vhost_dev_cleanup(struct vhost_dev *dev)
{
   WARN_ON(!list_empty(&dev->work_list));
   if (dev->worker) {
      kthread_stop(dev->worker);
      dev->worker = NULL;
      dev->kcov_handle = 0;
   }
}
// in include/asm-genric/bug.h
#define WARN_ON(condition) ({       \
  int __ret_warn_on = !!(condition);     \
  if (unlikely(__ret_warn_on))      \
    __WARN();       \
  unlikely(__ret_warn_on);      \
})
```

Critical struct identification



```
// comparison
tmp = icmp (conv, 0)
// conditional jump
br (tmp, label1, label2)

label1:
call @printk(...) // log

label2:
br (label3) // direct jump

label3:
call @bug(...) // call

define bug(..)
call @printk(...) // log
```

Critical struct identification

```
// source code
walk->offset = sg->offset;

// pseudo binary code after instrumentation
kasan_check_read(&sg->offset, sizeof(var));
tmp = LOAD(&sg->offset, sizeof(var)); // first access
kasan_check_write(&walk->offset, sizeof(var));
STORE(tmp, &walk->offset); // second access
```

Critical struct identification

- Backward taint
- Nested struct and union
- Loop counter

Critical struct identification

- Tainted value definition
- Syscall entry, interrupt handler
- Entry of function

Kernel Structure Ranking

- Popular struct such as struct list_head, struct socket

```c
// definition of struct sk_buff
struct sk_buff {
   union {
      struct rb_node rbnode;
   };
   ...
   struct skb_ext *extensions;
};
```

Kernel Structure Ranking

```c
static inline void *__skb_push(struct sk_buff *skb, ...)
{
    return skb->data;
}

int ip6_fraglist_init(...)
{
    struct frag_hdr *fh;
    // type casting from void* to struct frag_hdr*
    fh = __skb_push(skb, sizeof(struct frag_hdr));
}
```

Object-driven Kernel Fuzzing

- Instrument critical objects related statement
- Corpus: unseen basic block with critical object operation
- Corpus: system call covers more mode and the same system call has critical object operation
- Seed: Not use new system call
- Mutation: new system call in seed corpus

Object-driven Kernel Fuzzing

- Resource and arguments that system calls operate are necessary for successfully triggering a target kernel bug

```
r0 = openat(...,
'/dev/dsp1\x00');
ioctl(r0, ...);
write(r0, ...);
read(r0, ...);
```

(a) 7022420

```
// initial PoC: max = -1
bpf$MAP_CREATE(...,
    @max=0xffffffffffffffff);
// exit triggers GFP
exit(0);
```

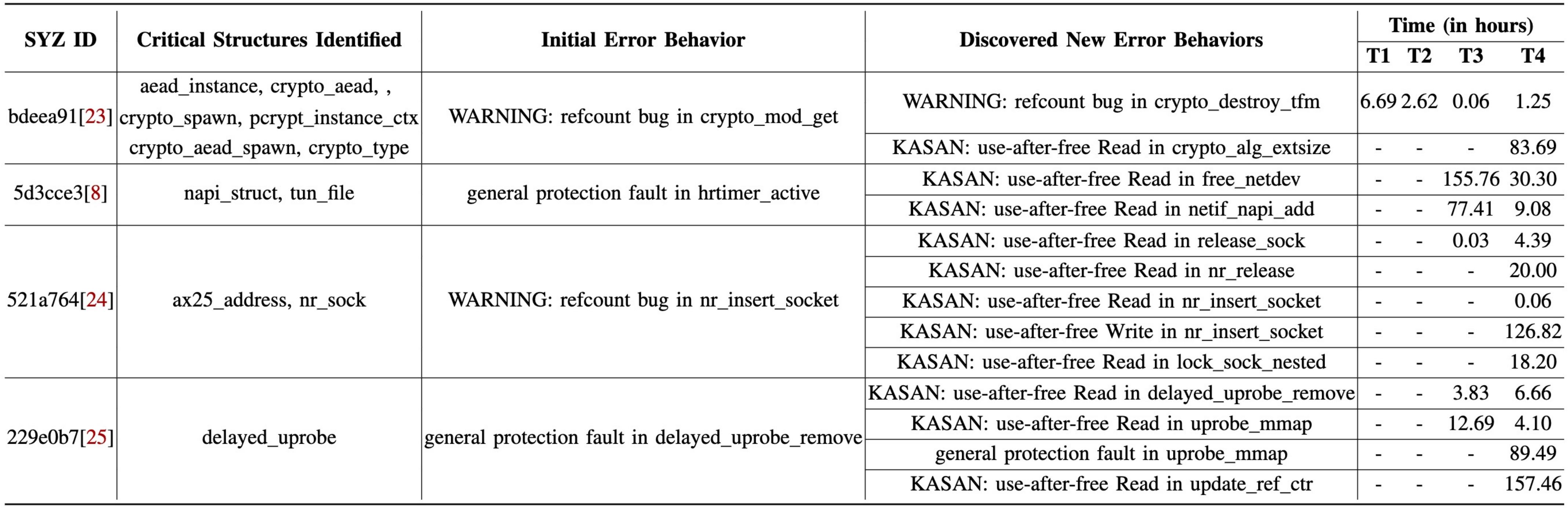
(b) 692a8c2

- Group the system call specification templates (resource/usage)
- Only Changed: constant, pointer referencing a memory region, checksum, and resource

Evaluation

- Bug source: Syzbot -> 50 bugs
  - With PoC program
  - No KMSAN
- Five version kernel (5.6 – 5.10) -> 10 bugs
  - Two recently reproducible bug
- Four VM (7 days)
  - Two GREBE + Syzkaller
  - Two without mutation optimization

Evaluation



SYZ ID	Critical Structures Identified	Initial Error Behavior	Discovered New Error Behaviors	Time (in hours)			
				T1	T2	T3	T4
bdeea91[23]	aead_instance, crypto_aead, , crypto_spawn, pcrypt_instance_ctx crypto_aead_spawn, crypto_type	WARNING: refcount bug in crypto_mod_get	WARNING: refcount bug in crypto_destroy_tfm	6.69	2.62	0.06	1.25
			KASAN: use-after-free Read in crypto_alg_extsize	-	-	-	83.69
5d3cce3[8]	napi_struct, tun_file	general protection fault in hrtimer_active	KASAN: use-after-free Read in free_netdev	-	-	155.76	30.30
			KASAN: use-after-free Read in netif_napi_add	-	-	77.41	9.08
521a764[24]	ax25_address, nr_sock	WARNING: refcount bug in nr_insert_socket	KASAN: use-after-free Read in release_sock	-	-	0.03	4.39
			KASAN: use-after-free Read in nr_release	-	-	-	20.00
			KASAN: use-after-free Read in nr_insert_socket	-	-	-	0.06
			KASAN: use-after-free Write in nr_insert_socket	-	-	-	126.82
			KASAN: use-after-free Read in lock_sock_nested	-	-	-	18.20
229e0b7[25]	delayed_uprobe	general protection fault in delayed_uprobe_remove	KASAN: use-after-free Read in delayed_uprobe_remove	-	-	3.83	6.66
			KASAN: use-after-free Read in uprobe_mmap	-	-	12.69	4.10
			general protection fault in uprobe_mmap	-	-	-	89.49
			KASAN: use-after-free Read in update_ref_ctr	-	-	-	157.46

SYZ ID	Exploitability Change	SYZ ID	Exploitability Change
d1baeb1 [27]	LL $\rightarrow$ L (2) $\star$	de28cb0 [28]	LL $\rightarrow$ L (5)
8eceaff [29]	LL $\rightarrow$ L (2) $\star$	f56bbe6 [30]	LL $\rightarrow$ L (1)
bb7fa48 [31]	LL $\rightarrow$ L (1)	f0ec9a3 [32]	LL $\rightarrow$ L (1)
d767177 [33]	LL $\rightarrow$ L (2)	5d3cce3 [8]	LL $\rightarrow$ L (2) $\star$
460cc94 [34]	LL $\rightarrow$ L (1)	692a8c2 [12]	LL $\rightarrow$ L (12) $\star$
0df4c1a [35]	LL $\rightarrow$ L (3)	4cf5ee7 [36]	LL $\rightarrow$ L (2)
229e0b7 [25]	LL $\rightarrow$ L (3)	502c872 [37]	LL $\rightarrow$ L (1)
163388d [38]	LL $\rightarrow$ L (1)	b36d7e4 [39]	LL $\rightarrow$ L (1)
bdeea91 [23]	LL $\rightarrow$ L (1)	1fd1d44 [40]	LL $\rightarrow$ L (1)
b9b37a7 [41]	LL $\rightarrow$ L (4)	695527b [42]	LL $\rightarrow$ L (1)
0d93140 [43]	LL $\rightarrow$ L (1)	85fd017 [44]	LL $\rightarrow$ L (4) $\star$
b0e30ab [45]	LL $\rightarrow$ L (1)	6a03985 [46]	LL $\rightarrow$ L (3) $\star$
d5222b3 [47]	LL $\rightarrow$ L (1)	575a090 [48]	LL $\rightarrow$ L (1)
3a6c997 [49]	L $\rightarrow$ L (10)	27ae1ae [50]	L $\rightarrow$ L (1)
cbb2898 [51]	L $\rightarrow$ L (1)	4bf11aa [52]	L $\rightarrow$ L (1)
e4be308 [53]	L $\rightarrow$ L (11)	7022420 [11]	L $\rightarrow$ L (1)
3b7409f [54]	L $\rightarrow$ L (1)	ddaf58b [55]	L $\rightarrow$ L (2)