Designing subsystems for **FUZZ-ability**

Dmitry Vyukov, dvyukov@

LPC 2022

syzkaller + syzbot

syzkaller:

- kernel fuzzer (randomized testing)
- executes infinite stream of random test programs
- code coverage-guided
- uses UAPI descriptions

syzbot:

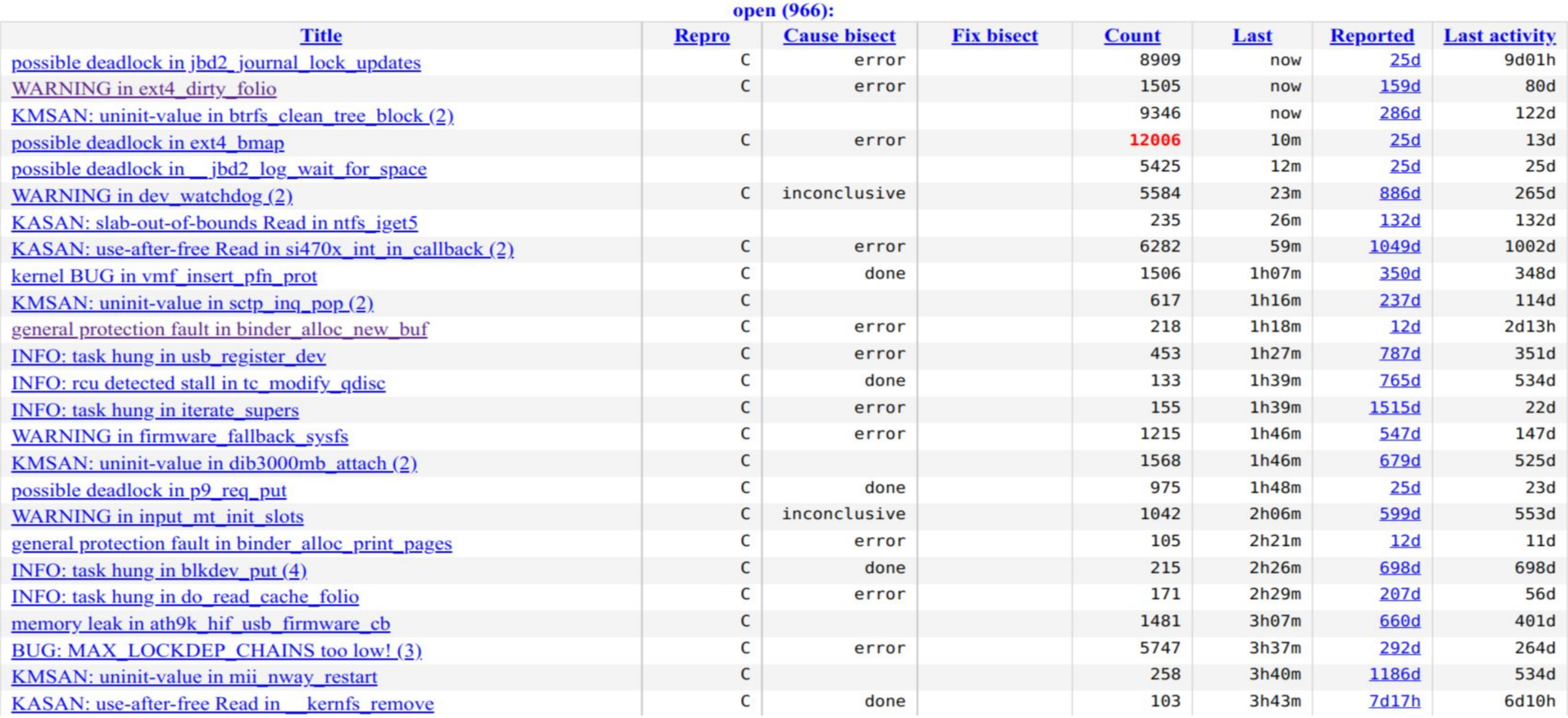
- continuous builds
- bug reporting/tracking
- cause/fix bisection, patch testing
- dashboard

>5000 bugs reported

>3000 bugs fixed

>7000 LTS backports

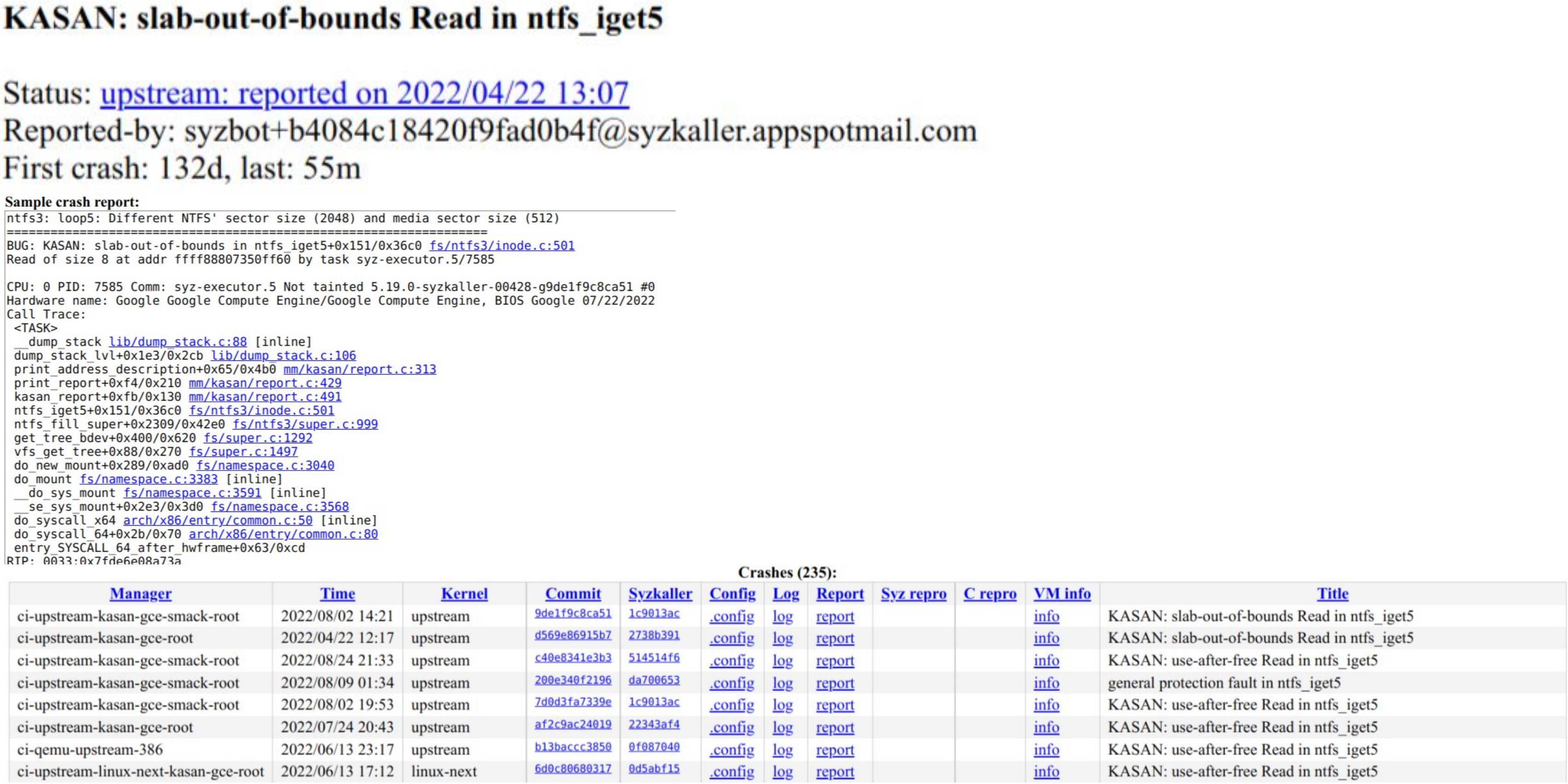
syzkaller.appspot.com



open (966):

Title	Repro	Cause bisect	Fix bisect	Count	Last	Reported	Last activity
possible deadlock in jbd2_journal_lock_updates	C	error		8909	now	25d	9d01h
WARNING in ext4_dirty_folio	C	error		1505	now	159d	80d
KMSAN: uninit-value in btrfs_clean_tree_block (2)				9346	now	286d	122d
possible deadlock in ext4_bmap	C	error		12006	10m	25d	13d
possible deadlock in __jbd2_log_wait_for_space				5425	12m	25d	25d
WARNING in dev_watchdog (2)	C	inconclusive		5584	23m	886d	265d
KASAN: slab-out-of-bounds Read in ntfs_iget5				235	26m	132d	132d
KASAN: use-after-free Read in si470x_int_in_callback (2)	C	error		6282	59m	1049d	1002d
kernel BUG in vmf_insert_pfn_prot	C	done		1506	1h07m	350d	348d
KMSAN: uninit-value in sctp_inq_pop (2)	C			617	1h16m	237d	114d
general protection fault in binder_alloc_new_buf	C	error		218	1h18m	12d	2d13h
INFO: task hung in usb_register_dev	C	error		453	1h27m	787d	351d
INFO: rcu detected stall in tc_modify_qdisc	C	done		133	1h39m	765d	534d
INFO: task hung in iterate_supers	C	error		155	1h39m	1515d	22d
WARNING in firmware_fallback_sysfs	C	error		1215	1h46m	547d	147d
KMSAN: uninit-value in dib3000mb_attach (2)	C			1568	1h46m	679d	525d
possible deadlock in p9_req_put	C	done		975	1h48m	25d	23d
WARNING in input_mt_init_slots	C	inconclusive		1042	2h06m	599d	553d
general protection fault in binder_alloc_print_pages	C	error		105	2h21m	12d	11d
INFO: task hung in blkdev_put (4)	C	done		215	2h26m	698d	698d
INFO: task hung in do_read_cache_folio	C	error		171	2h29m	207d	56d
memory leak in ath9k_hif_usb_firmware_cb	C			1481	3h07m	660d	401d
BUG: MAX_LOCKDEP_CHAINS too low! (3)	C	error		5747	3h37m	292d	264d
KMSAN: uninit-value in mii_nway_restart	C			258	3h40m	1186d	534d
KASAN: use-after-free Read in __kernfs_remove	C	done		103	3h43m	7d17h	6d10h

Bug info



KASAN: slab-out-of-bounds Read in ntfs_iget5

Status: upstream: reported on 2022/04/22 13:07
Reported-by: syzbot+b4084c18420f9fad0b4f@syzkaller.appspotmail.com
First crash: 132d, last: 55m

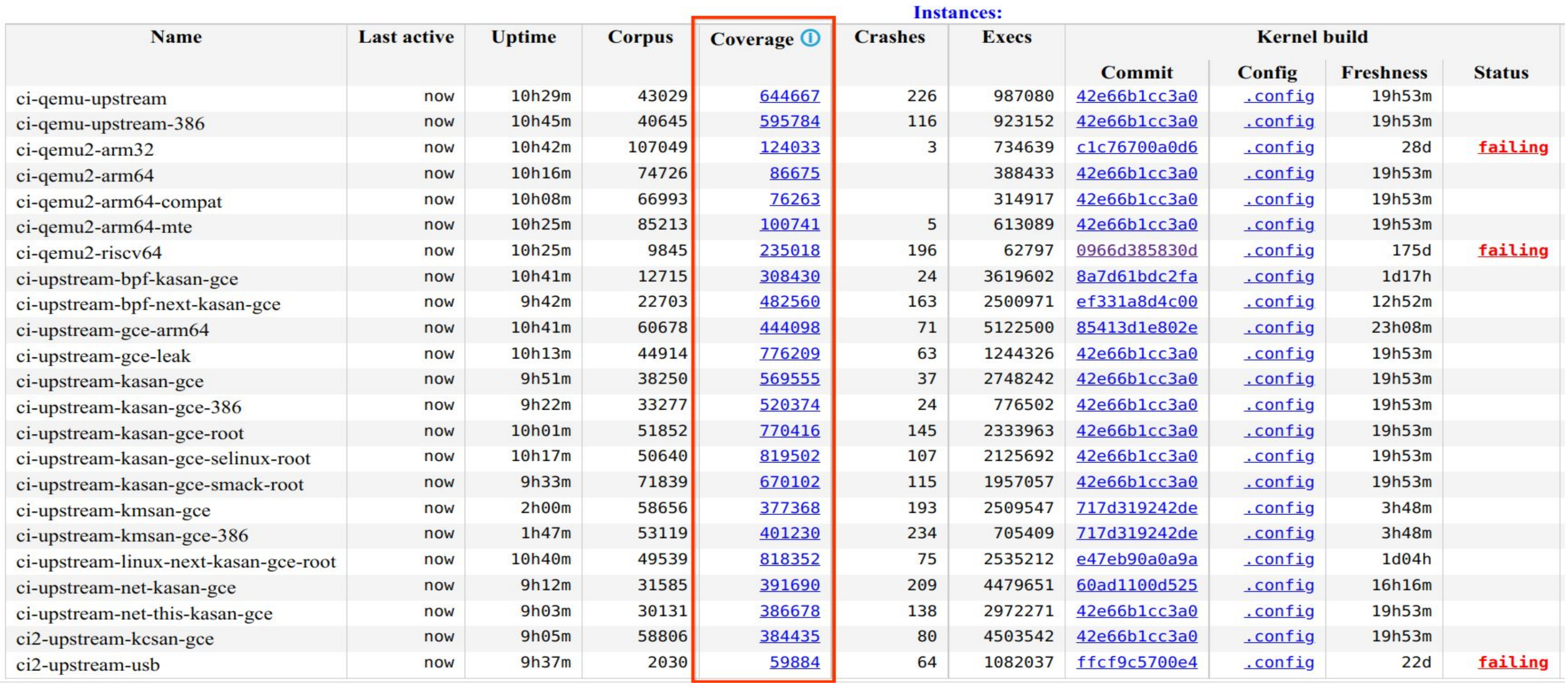
Sample crash report:

```
ntfs3: loop5: Different NTFS' sector size (2048) and media sector size (512)
==================================================================
BUG: KASAN: slab-out-of-bounds in ntfs_iget5+0x151/0x36c0 fs/ntfs3/inode.c:501
Read of size 8 at addr ffff88807350ff60 by task syz-executor.5/7585

CPU: 0 PID: 7585 Comm: syz-executor.5 Not tainted 5.19.0-syzkaller-00428-g9de1f9c8ca51 #0
Hardware name: Google Google Compute Engine/Google Compute Engine, BIOS Google 07/22/2022
Call Trace:
 <TASK>
 __dump_stack lib/dump_stack.c:88 [inline]
 dump_stack_lvl+0x1e3/0x2cb lib/dump_stack.c:106
 print_address_description+0x65/0x4b0 mm/kasan/report.c:313
 print_report+0xf4/0x210 mm/kasan/report.c:429
 kasan_report+0xfb/0x130 mm/kasan/report.c:491
 ntfs_iget5+0x151/0x36c0 fs/ntfs3/inode.c:501
 ntfs_fill_super+0x2309/0x42e0 fs/ntfs3/super.c:999
 get_tree_bdev+0x400/0x620 fs/super.c:1292
 vfs_get_tree+0x88/0x270 fs/super.c:1497
 do_new_mount+0x289/0xad0 fs/namespace.c:3040
 do_mount fs/namespace.c:3383 [inline]
 __do_sys_mount fs/namespace.c:3591 [inline]
 __se_sys_mount+0x2e3/0x3d0 fs/namespace.c:3568
 do_syscall_x64 arch/x86/entry/common.c:50 [inline]
 do_syscall_64+0x2b/0x70 arch/x86/entry/common.c:80
 entry_SYSCALL_64_after_hwframe+0x63/0xcd
RIP: 0033:0x7fde6e08a73a
```

Crashes (235):

Manager	Time	Kernel	Commit	Syzkaller	Config	Log	Report	Syz repro	C repro	VM info	Title
ci-upstream-kasan-gce-smack-root	2022/08/02 14:21	upstream	9de1f9c8ca51	1c9013ac	.config	log	report			info	KASAN: slab-out-of-bounds Read in ntfs_iget5
ci-upstream-kasan-gce-root	2022/04/22 12:17	upstream	d569e86915b7	2738b391	.config	log	report			info	KASAN: slab-out-of-bounds Read in ntfs_iget5
ci-upstream-kasan-gce-smack-root	2022/08/24 21:33	upstream	c40e8341e3b3	514514f6	.config	log	report			info	KASAN: use-after-free Read in ntfs_iget5
ci-upstream-kasan-gce-smack-root	2022/08/09 01:34	upstream	200e340f2196	da700653	.config	log	report			info	general protection fault in ntfs_iget5
ci-upstream-kasan-gce-smack-root	2022/08/02 19:53	upstream	7d0d3fa7339e	1c9013ac	.config	log	report			info	KASAN: use-after-free Read in ntfs_iget5
ci-upstream-kasan-gce-root	2022/07/24 20:43	upstream	af2c9ac24019	22343af4	.config	log	report			info	KASAN: use-after-free Read in ntfs_iget5
ci-qemu-upstream-386	2022/06/13 23:17	upstream	b13baccc3850	0f087040	.config	log	report			info	KASAN: use-after-free Read in ntfs_iget5
ci-upstream-linux-next-kasan-gce-root	2022/06/13 17:12	linux-next	6d0c80680317	0d5abf15	.config	log	report			info	KASAN: use-after-free Read in ntfs_iget5

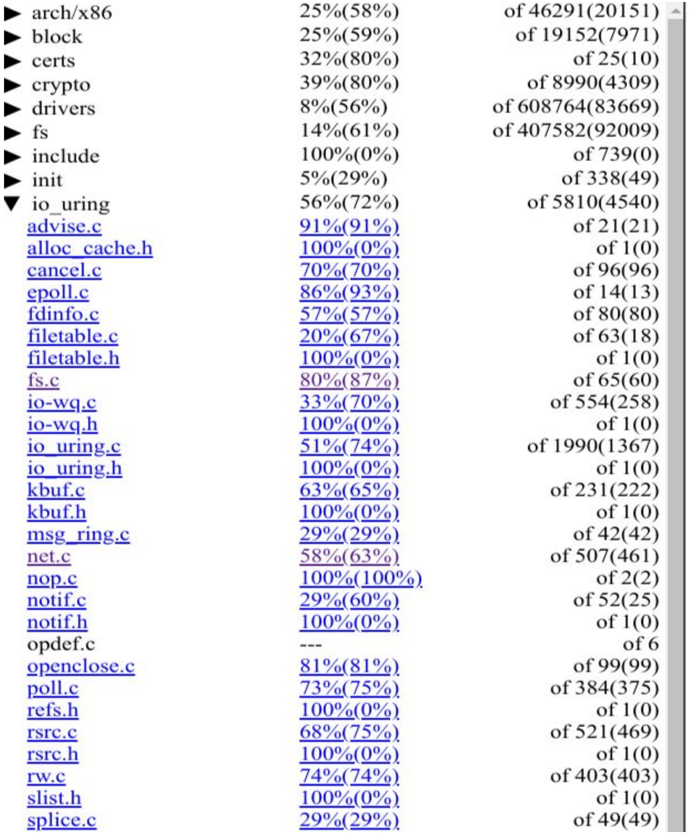
Code coverage reports



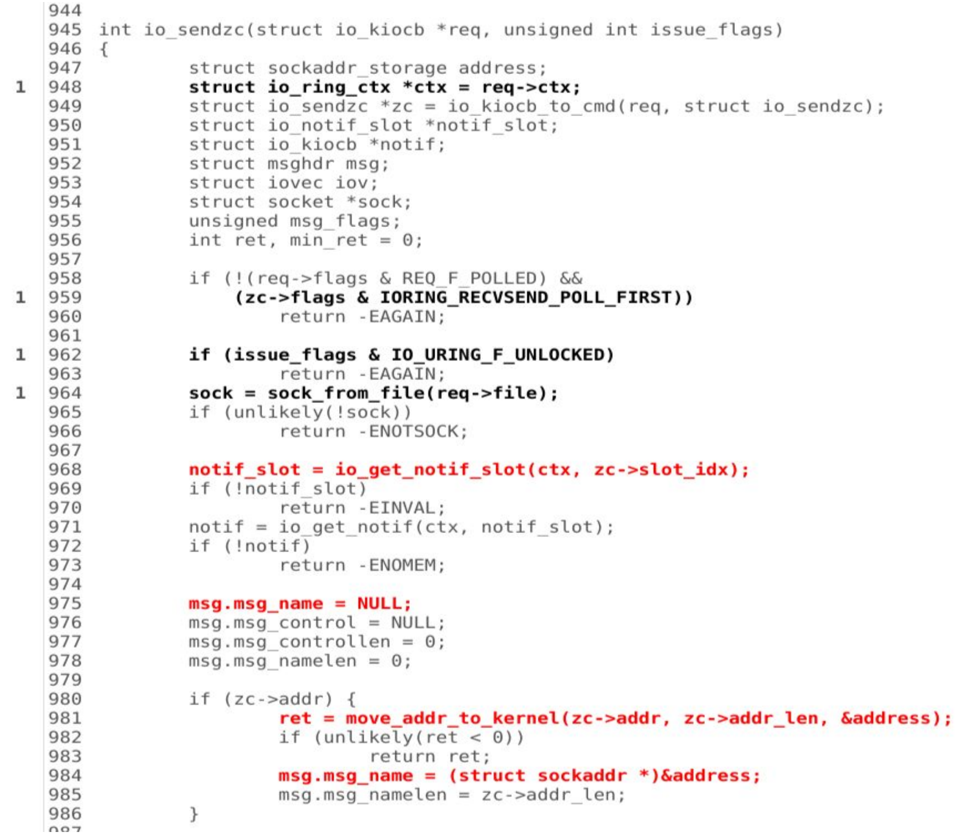
Instances:

Name	Last active	Uptime	Corpus	Coverage ⓘ	Crashes	Execs	Kernel build			
							Commit	Config	Freshness	Status
ci-qemu-upstream	now	10h29m	43029	644667	226	987080	42e66b1cc3a0	.config	19h53m	
ci-qemu-upstream-386	now	10h45m	40645	595784	116	923152	42e66b1cc3a0	.config	19h53m	
ci-qemu2-arm32	now	10h42m	107049	124033	3	734639	c1c76700a0d6	.config	28d	failing
ci-qemu2-arm64	now	10h16m	74726	86675		388433	42e66b1cc3a0	.config	19h53m	
ci-qemu2-arm64-compat	now	10h08m	66993	76263		314917	42e66b1cc3a0	.config	19h53m	
ci-qemu2-arm64-mte	now	10h25m	85213	100741	5	613089	42e66b1cc3a0	.config	19h53m	
ci-qemu2-riscv64	now	10h25m	9845	235018	196	62797	0966d385830d	.config	175d	failing
ci-upstream-bpf-kasan-gce	now	10h41m	12715	308430	24	3619602	8a7d61bdc2fa	.config	1d17h	
ci-upstream-bpf-next-kasan-gce	now	9h42m	22703	482560	163	2500971	ef331a8d4c00	.config	12h52m	
ci-upstream-gce-arm64	now	10h41m	60678	444098	71	5122500	85413d1e802e	.config	23h08m	
ci-upstream-gce-leak	now	10h13m	44914	776209	63	1244326	42e66b1cc3a0	.config	19h53m	
ci-upstream-kasan-gce	now	9h51m	38250	569555	37	2748242	42e66b1cc3a0	.config	19h53m	
ci-upstream-kasan-gce-386	now	9h22m	33277	520374	24	776502	42e66b1cc3a0	.config	19h53m	
ci-upstream-kasan-gce-root	now	10h01m	51852	770416	145	2333963	42e66b1cc3a0	.config	19h53m	
ci-upstream-kasan-gce-selinux-root	now	10h17m	50640	819502	107	2125692	42e66b1cc3a0	.config	19h53m	
ci-upstream-kasan-gce-smack-root	now	9h33m	71839	670102	115	1957057	42e66b1cc3a0	.config	19h53m	
ci-upstream-kmsan-gce	now	2h00m	58656	377368	193	2509547	717d319242de	.config	3h48m	
ci-upstream-kmsan-gce-386	now	1h47m	53119	401230	234	705409	717d319242de	.config	3h48m	
ci-upstream-linux-next-kasan-gce-root	now	10h40m	49539	818352	75	2535212	e47eb90a0a9a	.config	1d04h	
ci-upstream-net-kasan-gce	now	9h12m	31585	391690	209	4479651	60ad1100d525	.config	16h16m	
ci-upstream-net-this-kasan-gce	now	9h03m	30131	386678	138	2972271	42e66b1cc3a0	.config	19h53m	
ci2-upstream-kcsan-gce	now	9h05m	58806	384435	80	4503542	42e66b1cc3a0	.config	19h53m	
ci2-upstream-usb	now	9h37m	2030	59884	64	1082037	ffcf9c5700e4	.config	22d	failing

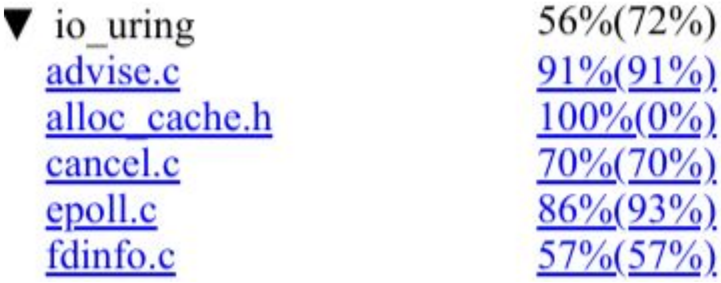
Code coverage reports



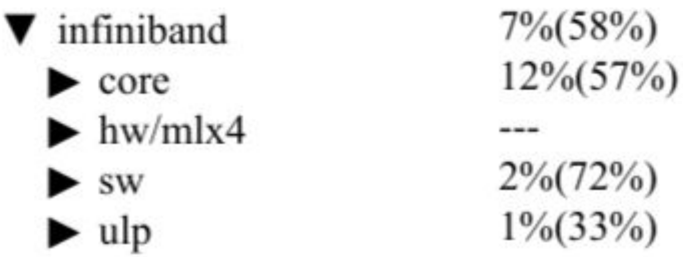
▶ arch/x86	25%(58%)	of 46291(20151)
▶ block	25%(59%)	of 19152(7971)
▶ certs	32%(80%)	of 25(10)
▶ crypto	39%(80%)	of 8990(4309)
▶ drivers	8%(56%)	of 608764(83669)
▶ fs	14%(61%)	of 407582(92009)
▶ include	100%(0%)	of 739(0)
▶ init	5%(29%)	of 338(49)
▼ io_uring	56%(72%)	of 5810(4540)
advise.c	91%(91%)	of 21(21)
alloc_cache.h	100%(0%)	of 1(0)
cancel.c	70%(70%)	of 96(96)
epoll.c	86%(93%)	of 14(13)
fdinfo.c	57%(57%)	of 80(80)
filetable.c	20%(67%)	of 63(18)
filetable.h	100%(0%)	of 1(0)
fs.c	80%(87%)	of 65(60)
io-wq.c	33%(70%)	of 554(258)
io-wq.h	100%(0%)	of 1(0)
io_uring.c	51%(74%)	of 1990(1367)
io_uring.h	100%(0%)	of 1(0)
kbuf.c	63%(65%)	of 231(222)
kbuf.h	100%(0%)	of 1(0)
msg_ring.c	29%(29%)	of 42(42)
net.c	58%(63%)	of 507(461)
nop.c	100%(100%)	of 2(2)
notif.c	29%(60%)	of 52(25)
notif.h	100%(0%)	of 1(0)
opdef.c	---	of 6
openclose.c	81%(81%)	of 99(99)
poll.c	73%(75%)	of 384(375)
refs.h	100%(0%)	of 1(0)
rsrc.c	68%(75%)	of 521(469)
rsrc.h	100%(0%)	of 1(0)
rw.c	74%(74%)	of 403(403)
slist.h	100%(0%)	of 1(0)
splice.c	29%(29%)	of 49(49)



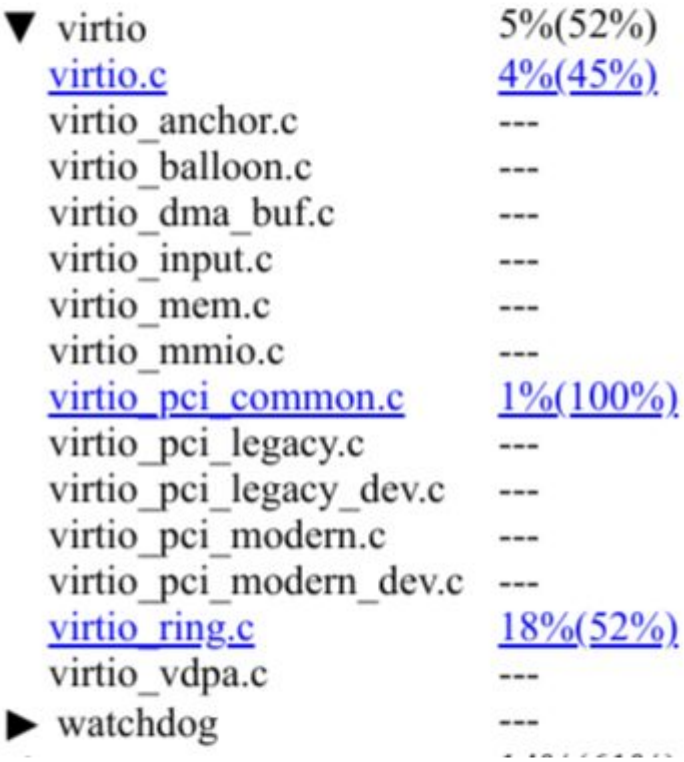
```

    int io_sendzc(struct io_kiocb *req, unsigned int issue_flags)
    {
            struct sockaddr_storage address;
1           struct io_ring_ctx *ctx = req->ctx;
            struct io_sendzc *zc = io_kiocb_to_cmd(req, struct io_sendzc);
            struct io_notif_slot *notif_slot;
            struct io_kiocb *notif;
            struct msghdr msg;
            struct iovec iov;
            struct socket *sock;
            unsigned msg_flags;
            int ret, min_ret = 0;

            if (!(req->flags & REQ_F_POLLED) &&
1               (zc->flags & IORING_RECVSEND_POLL_FIRST))
                    return -EAGAIN;

1           if (issue_flags & IO_URING_F_UNLOCKED)
                    return -EAGAIN;
1           sock = sock_from_file(req->file);
            if (unlikely(!sock))
                    return -ENOTSOCK;

            notif_slot = io_get_notif_slot(ctx, zc->slot_idx);
            if (!notif_slot)
                    return -EINVAL;
            notif = io_get_notif(ctx, notif_slot);
            if (!notif)
                    return -ENOMEM;

            msg.msg_name = NULL;
            msg.msg_control = NULL;
            msg.msg_controllen = 0;
            msg.msg_namelen = 0;

            if (zc->addr) {
                    ret = move_addr_to_kernel(zc->addr, zc->addr_len, &address);
                    if (unlikely(ret < 0))
                            return ret;
                    msg.msg_name = (struct sockaddr *)&address;
                    msg.msg_namelen = zc->addr_len;
            }
```



```
▼ io_uring            56%(72%)
  advise.c            91%(91%)
  alloc_cache.h       100%(0%)
  cancel.c            70%(70%)
  epoll.c             86%(93%)
  fdinfo.c            57%(57%)
```



```
▼ infiniband          7%(58%)
  ▶ core              12%(57%)
  ▶ hw/mlx4           ---
  ▶ sw                2%(72%)
  ▶ ulp               1%(33%)
```



```
▼ virtio                      5%(52%)
  virtio.c                    4%(45%)
  virtio_anchor.c             ---
  virtio_balloon.c            ---
  virtio_dma_buf.c            ---
  virtio_input.c              ---
  virtio_mem.c                ---
  virtio_mmio.c               ---
  virtio_pci_common.c         1%(100%)
  virtio_pci_legacy.c         ---
  virtio_pci_legacy_dev.c     ---
  virtio_pci_modern.c         ---
  virtio_pci_modern_dev.c     ---
  virtio_ring.c               18%(52%)
  virtio_vdpa.c               ---
▶ watchdog                    ---
```

Make your code well covered!

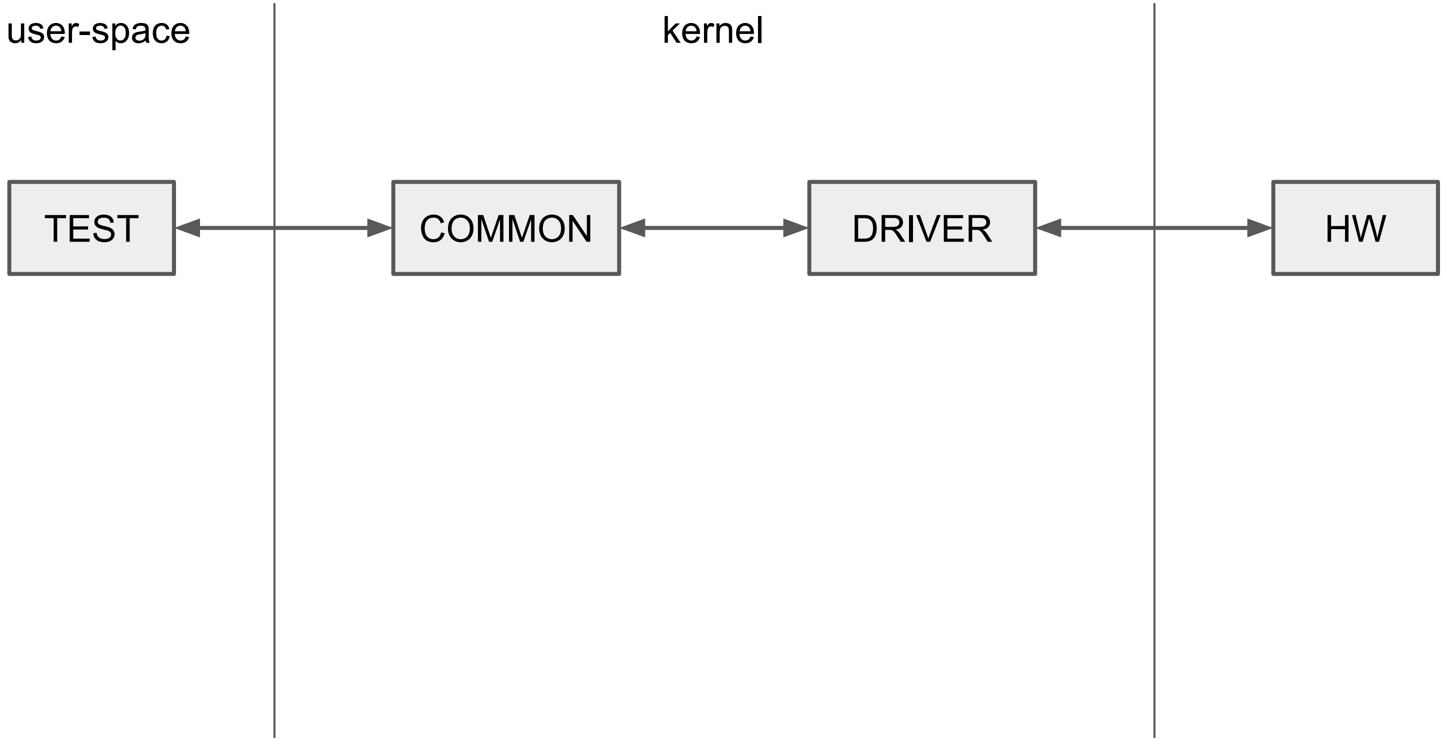
Fuzz-ability ≈≈ Test-ability

- Test-ability:
  - user-space test
  - runs in VM
  - isolated/reproducible
  - automated
  - single kernel build

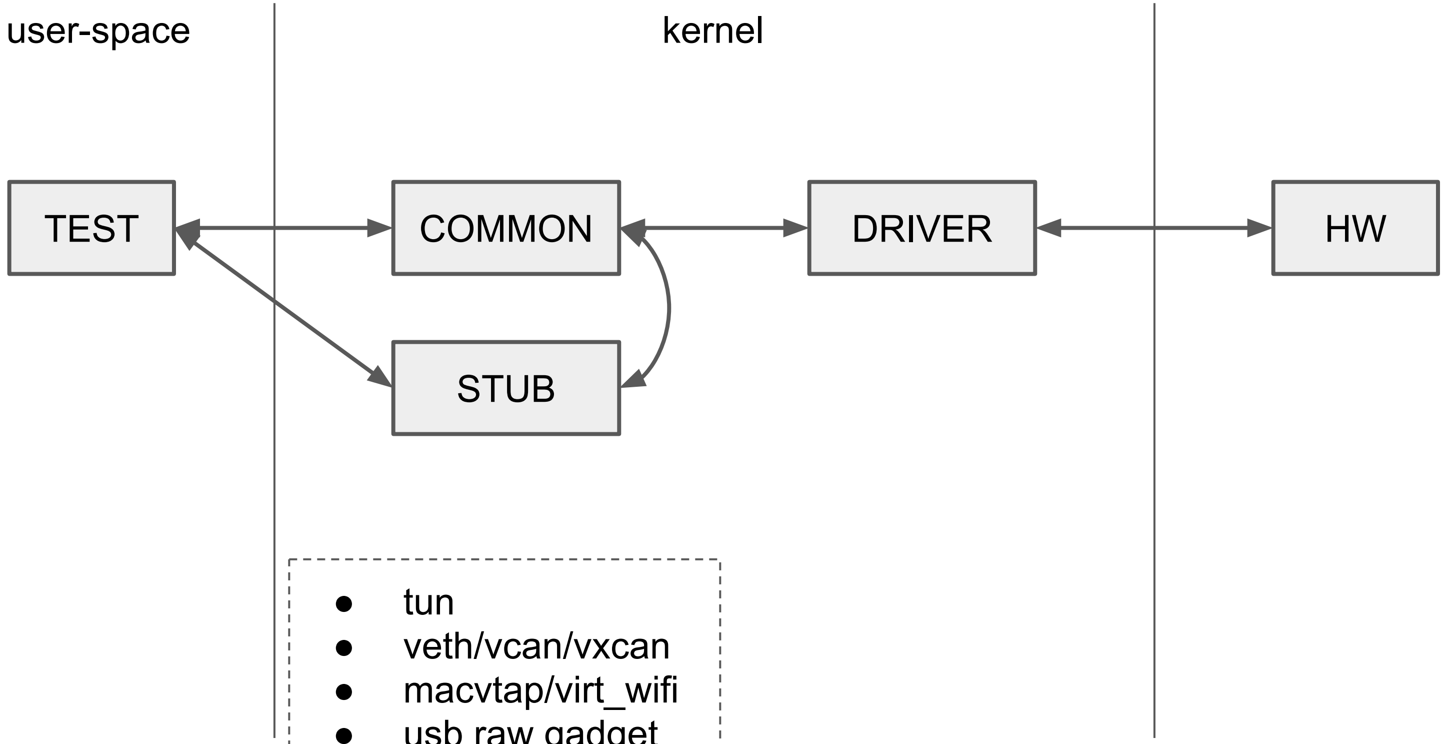
Fuzz-ability ≈≈ Test-ability

- Test-ability:
  - user-space test
  - runs in VM
  - isolated/reproducible
  - automated
  - single kernel build
- Fuzz-ability:
  - hardened against "crazy" inputs
  - isolated/reproducible with random inputs
  - reasonably easy to generate inputs automatically

Requiring HW

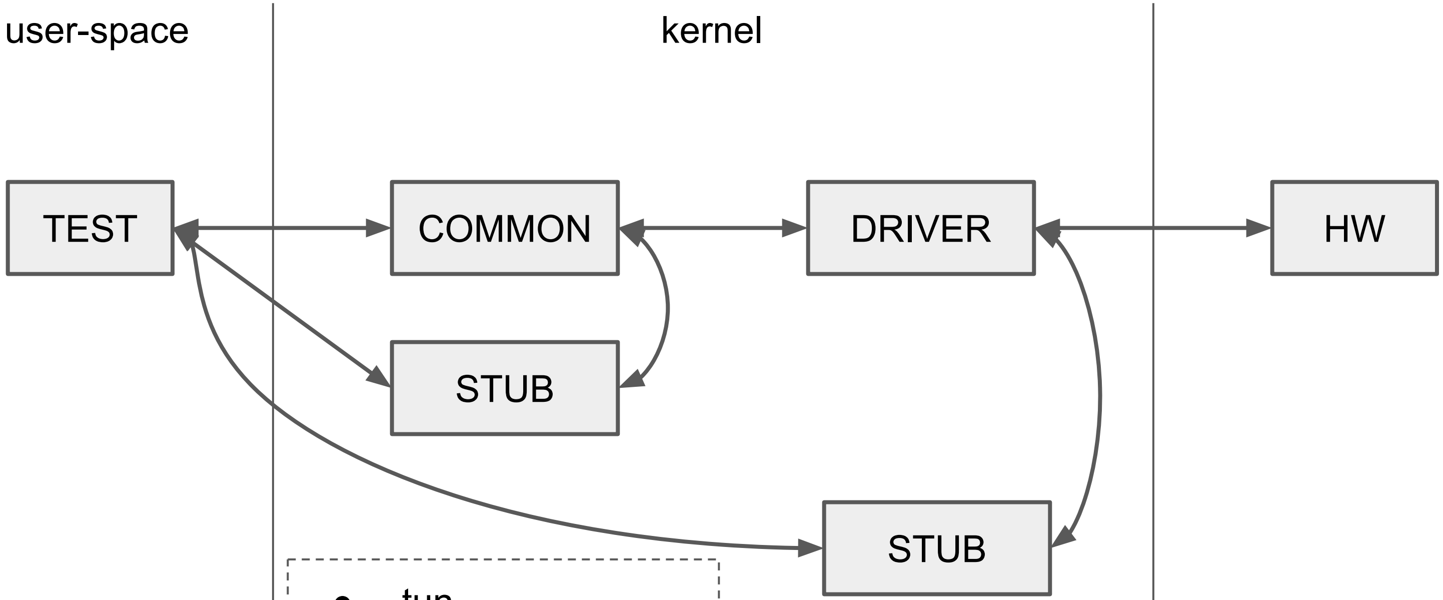


Requiring HW



- tun
- veth/vcan/vxcan
- macvtap/virt_wifi
- usb raw gadget
- vivid/vimc/vim2m

Requiring HW

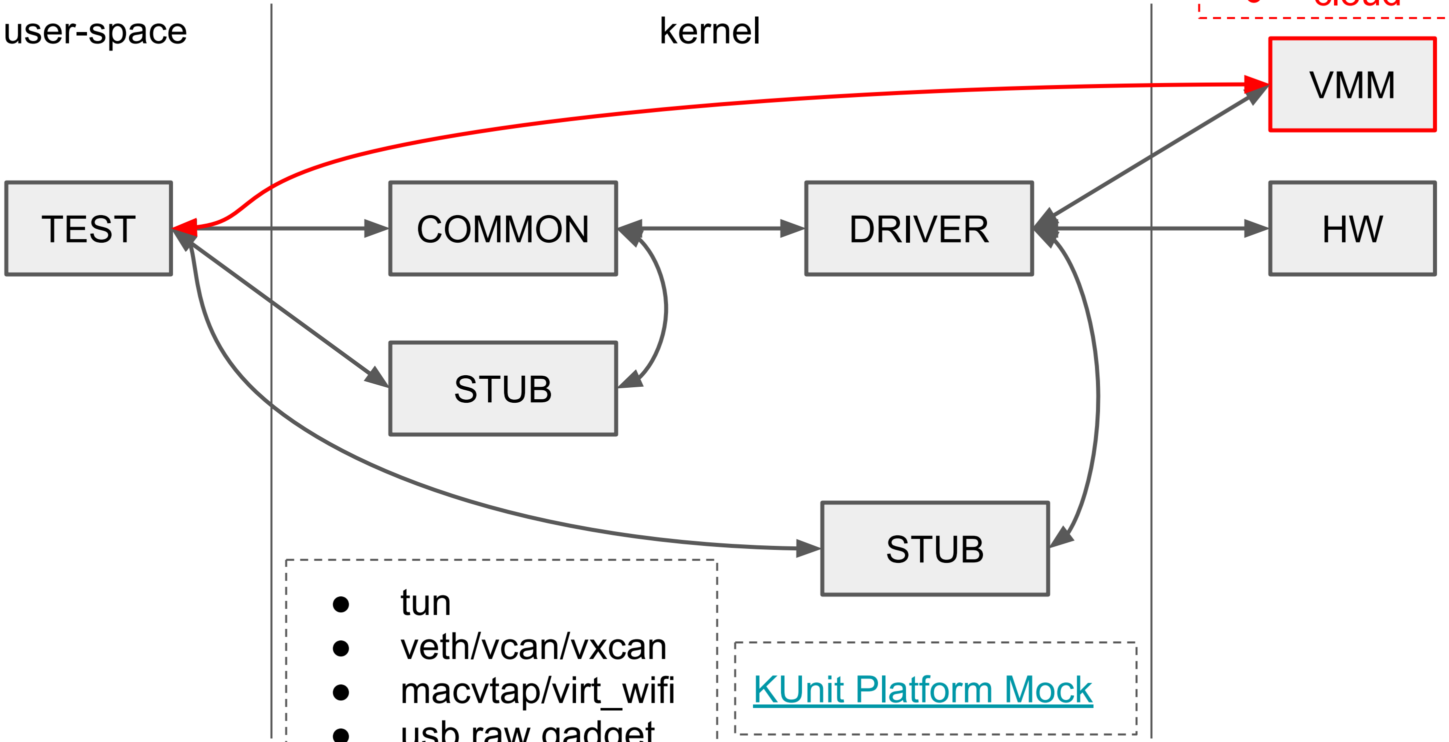


- tun
- veth/vcan/vxcan
- macvtap/virt_wifi
- usb raw gadget
- vivid/vimc/vim2m

KUnit Platform Mock

Requiring HW

- isolation
- reproducibility
- controllability
- cloud



user-space

kernel

VMM

TEST

COMMON

DRIVER

HW

STUB

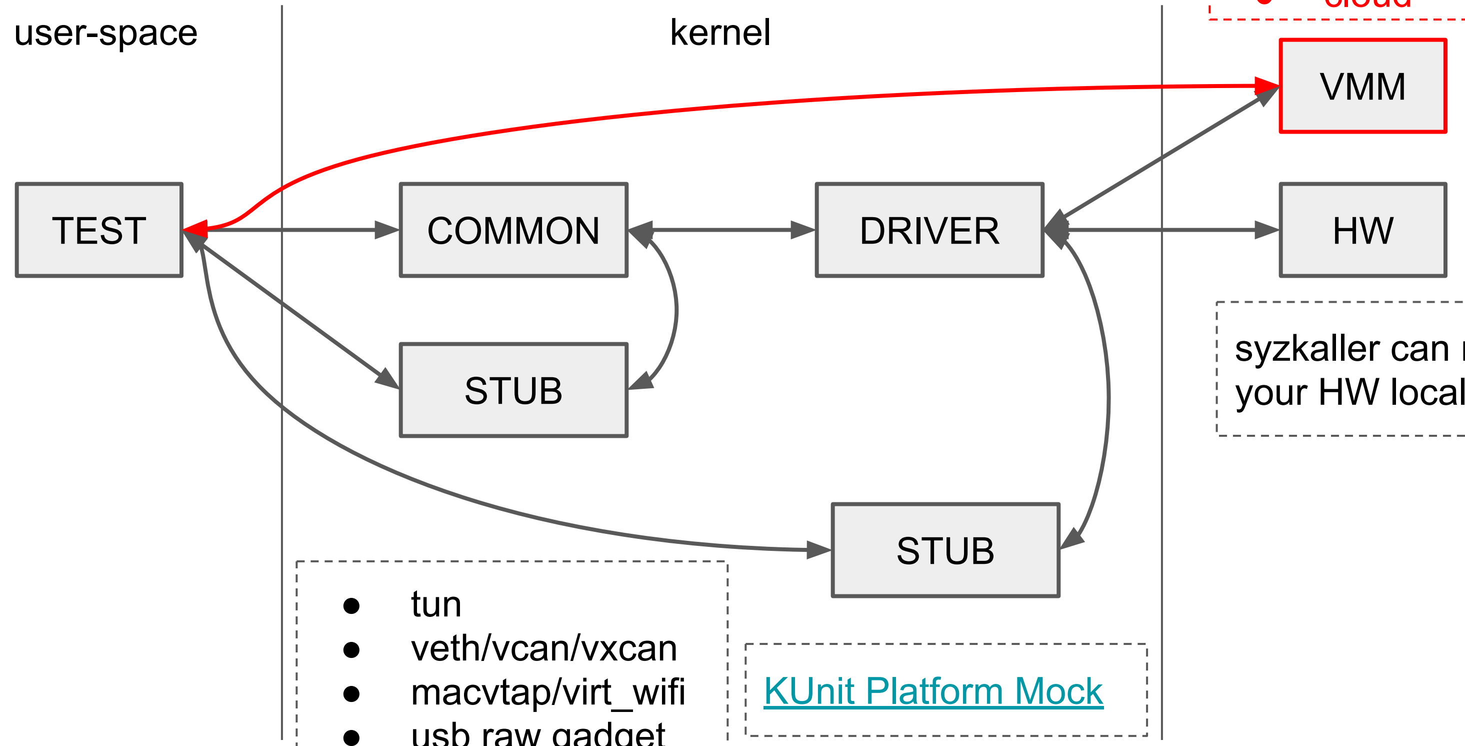
STUB

- tun
- veth/vcan/vxcan
- macvtap/virt_wifi
- usb raw gadget
- vivid/vimc/vim2m

KUnit Platform Mock

Requiring HW

- isolation
- reproducibility
- controllability
- cloud



syzkaller can run on your HW locally

- tun
- veth/vcan/vxcan
- macvtap/virt_wifi
- usb raw gadget
- vivid/vimc/vim2m

KUnit Platform Mock

Stub pitfalls

- only 1 device (CONFIG_NFC_SIM)
- fixed number of devices (CONFIG_VIDEO_VIVID)
- sticky global devices (CONFIG_WWAN_HWSIM)
- only init_net (CONFIG_EQUALIZER, CONFIG_BT)
- chatty configuration via debugfs (CONFIG_NFC_SIM, CONFIG_GPIO_SIM)

Stub pitfalls

- only 1 device (CONFIG_NFC_SIM)
- fixed number of devices (CONFIG_VIDEO_VIVID)
- sticky global devices (CONFIG_WWAN_HWSIM)
- only init_net (CONFIG_EQUALIZER, CONFIG_BT)
- chatty configuration via debugfs (CONFIG_NFC_SIM, CONFIG_GPIO_SIM)

Ideally:

- ioctl(/debugfs/your_sim, CREATE_STUB) -> (stub_fd, dev_fd)

Isolation, Reproducibility, Controllability, Automatability

- fresh transient instances (`memfd_create()` vs `shmem` id's)
- unreasonably restrictive assumptions (only init_net, only 1 instance)
- kill-ability (can't kill fuse)
- global settings (`net.core.bpf_jit_enable`)
- hardened against "crazy" inputs (user-space bugs! disabled in testing)

WARN_ON

- do add **`WARN_ON`** for invariants/assumptions!
- allows to catch logical bugs (found >1000)
- good for documentation (you can trust!)
- CONFIG_FOO_DEBUG for expensive checks (VM_BUG_ON)
- document CONFIG_FOO_DEBUG (checks vs logging vs stubs)
- DO NOT use WARN_ON for bad inputs

UAPI descriptions (syzlang)

```
open(file ptr[in, filename], flags flags[open_flags], mode flags[open_mode]) fd
pipe(pipefd ptr[out, pipefd])
close(fd fd)

open_flags = O_WRONLY, O_RDWR, ...

pipefd {
	rfd	fd
	wfd	fd
}
```

Stick to standard patterns!

Linked lists

```
struct v4l2_clip {
    struct v4l2_rect        c;
    struct v4l2_clip __user *next;
};
```

Compat handling

```
static int get_v4l2_window32(struct v4l2_window __user *p64,
                                      struct v4l2_window32 __user *p32,
                                      void __user *aux_buf, u32 aux_space)
{
           struct v4l2_clip32 __user *uclips;
           struct v4l2_clip __user *kclips;
           compat_caddr_t p;
           u32 clipcount;

           if (!access_ok(p32, sizeof(*p32)) ||
               copy_in_user(&p64->w, &p32->w, sizeof(p32->w)) ||
               assign_in_user(&p64->field, &p32->field) ||
               assign_in_user(&p64->chromakey, &p32->chromakey) ||
               assign_in_user(&p64->global_alpha, &p32->global_alpha) ||
               get_user(clipcount, &p32->clipcount) ||
               put_user(clipcount, &p64->clipcount))
                      return -EFAULT;
           if (clipcount > 2048)
                      return -EINVAL;
           if (!clipcount)
                      return put_user(NULL, &p64->clips);

           if (get_user(p, &p32->clips))
                      return -EFAULT;
           uclips = compat_ptr(p);
           if (aux_space < clipcount * sizeof(*kclips))
                      return -EFAULT;
           kclips = aux_buf;
           if (put_user(kclips, &p64->clips))
                      return -EFAULT;

           while (clipcount--) {
                      if (copy_in_user(&kclips->c, &uclips->c, sizeof(uclips->c)))
                                 return -EFAULT;
                      if (put_user(clipcount ? kclips + 1 : NULL, &kclips->next))
                                 return -EFAULT;
                      uclips++;
                      kclips++;
           }
           return 0;
}
```

```
static int put_v4l2_window32(struct v4l2_window __user *p64,
                                      struct v4l2_window32 __user *p32)
{
           struct v4l2_clip __user *kclips;
           struct v4l2_clip32 __user *uclips;
           compat_caddr_t p;
           u32 clipcount;

           if (copy_in_user(&p32->w, &p64->w, sizeof(p64->w)) ||
               assign_in_user(&p32->field, &p64->field) ||
               assign_in_user(&p32->chromakey, &p64->chromakey) ||
               assign_in_user(&p32->global_alpha, &p64->global_alpha) ||
               get_user(clipcount, &p64->clipcount) ||
               put_user(clipcount, &p32->clipcount))
                      return -EFAULT;
           if (!clipcount)
                      return 0;

           if (get_user(kclips, &p64->clips))
                      return -EFAULT;
           if (get_user(p, &p32->clips))
                      return -EFAULT;
           uclips = compat_ptr(p);
           while (clipcount--) {
                      if (copy_in_user(&uclips->c, &kclips->c, sizeof(uclips->c)))
                                 return -EFAULT;
                      uclips++;
                      kclips++;
           }
           return 0;
}
```

CVE-2017-13166

[PATCH] v4l2-compat-ioctl32.c: fix security bug in compat ioctl32

... the 64-bit ioctl expects all pointers to point to user space memory. As a workaround, **set_fs(KERNEL_DS)** is called to temporarily disable this extra safety check and allow kernel pointers. However, **this might introduce a security vulnerability:** The result of the 32-bit to 64-bit conversion is writeable by user space because the output buffer has been allocated via compat_alloc_user_space(). **A malicious user space** process could then manipulate pointers inside this output buffer, and due to the previous set_fs(KERNEL_DS) call, functions like get_user() or put_user() no longer prevent kernel memory access.

User pointers as IDs

```
io_submit(..., struct iocb __user * iocbp);
io_cancel(..., struct iocb __user * iocbp);  // must match io_submit pointer
```

TODO: this sucks

```
SYSCALL_DEFINE3(io_cancel,..., obj,...) {
    ...
    spin_lock_irq(&ctx->ctx_lock);
    /* TODO: use a hash or array, this sucks. */
    list_for_each_entry(kiocb, &ctx->active_reqs, ki_list) {
           if (kiocb->ki_res.obj == obj)
                  break;
    }
    spin_unlock_irq(&ctx->ctx_lock);
```

Consistency

```
struct mif6ctl {
...
    __u16     mif6c_pifi;        /* the index of the physical IF */
...
};


...
    dev = dev_get_by_index(net, vifc->mif6c_pifi);
```

Device index is `__u32` everywhere else!

NETFILTER

TL;DR; single blob in a very complex format:

- parallel arrays
- cross-references
- size of array with variable-size-elements
- derived info
- ...

NETFILTER

netfilter: add back stackpointer size **checks**
netfilter: x_tables: fix int **overflow** in xt_alloc_table_info()
netfilter: x_tables: avoid **out-of-bounds** reads in xt_request_find
netfilter: x_tables: initialise match/target **check** parameter struct
netfilter: x_tables: avoid **stack-out-of-bounds** read in xt_copy_counte
netfilter: x_tables: add and use xt_**check**_proc_name
netfilter: ebtables: CONFIG_COMPAT: **don't trust** userland offsets
netfilter: ebtables: handle string from userspace with **care**
netfilter: bridge: ebt_among: add missing match size **checks**
netfilter: bridge: ebt_among: add more missing match size **checks**
netfilter: x_tables: fix **4** pointer **leaks** to userspace
netfilter: ip_tables: fix **infoleak** to userspace
netfilter: arp_tables: fix **infoleak** to userspace
ipv4: netfilter: ip_tables: fix **information leak** to userland
ipv4: netfilter: arp_tables: fix **information leak** to userland
...

Human-oriented chatty protocols

Currently:

```
int fd1 = open("/dev/ptmx", O_RDWR);

int ptyno;
ioctl(fd1, TIOCGPTN, &ptyno);

char buf[128];
sprintf(buf, "/dev/pts/%d", ptyno);

int fd2 = open(buf, O_RDWR);
```

Human-oriented chatty protocols

Currently:

```
int fd1 = open("/dev/ptmx", O_RDWR);


int ptyno;

ioctl(fd1, TIOCGPTN, &ptyno);


char buf[128];

sprintf(buf, "/dev/pts/%d", ptyno);


int fd2 = open(buf, O_RDWR);
```

Ideally:

```
int fd1 = open("/dev/ptmx", O_RDWR);


int ptyfd[2]

ioctl(fd, TIOCGPTN, ptyfd);
```

UAPI

- stick to existing patterns
- be consistent with existing code
- don't use convoluted formats, derived info
- be nice to machines
- automatic interface extraction/inference
- write syzkaller descriptions with the code

Recap

- Allow testing in VMs
- Provide fresh transient instances
- Properly use WARN_ON
- Stick to standard UAPI patterns

matrix.lpc.events/#/room/#**syzkaller:**lpc.events

Thank you!

Q&A

Dmitry Vyukov, dvyukov@