



LINUX
SECURITY
SUMMIT

syzkaller

Update and Open Problems

@dvyukov

Agenda

- what is syzkaller
- new things
- open problems

syzkaller

OS kernel fuzzer:

- code-coverage-guided
- input-structure-aware
- multi-OS
- focus on automation

Syscall Descriptions

```
open(file ptr[in, filename], flags flags[open_flags]) fd  
read(fd fd, buf ptr[out, array[int8]], count len[buf])  
close(fd fd)
```

```
open_flags = O_RDONLY, O_WRONLY, ...
```

Programs

```
r0 = open(&(0x7f0000000000)="./file0", 0x3)  
read(r0, &(0x7f00000000100), 42)  
close(r0)
```

syzbot

syzkaller automation:

- continuous kernel/syzkaller update
- bug aggregation
- web UI
- bug reporting
- bug tracking

syzkaller.appspot.com

syzbot stats

Reported: 2281

Fixed: 1523 (66.7%)

Open: 758

syzbot stats

Reported: 2281

Fixed: 1523 (66.7%)

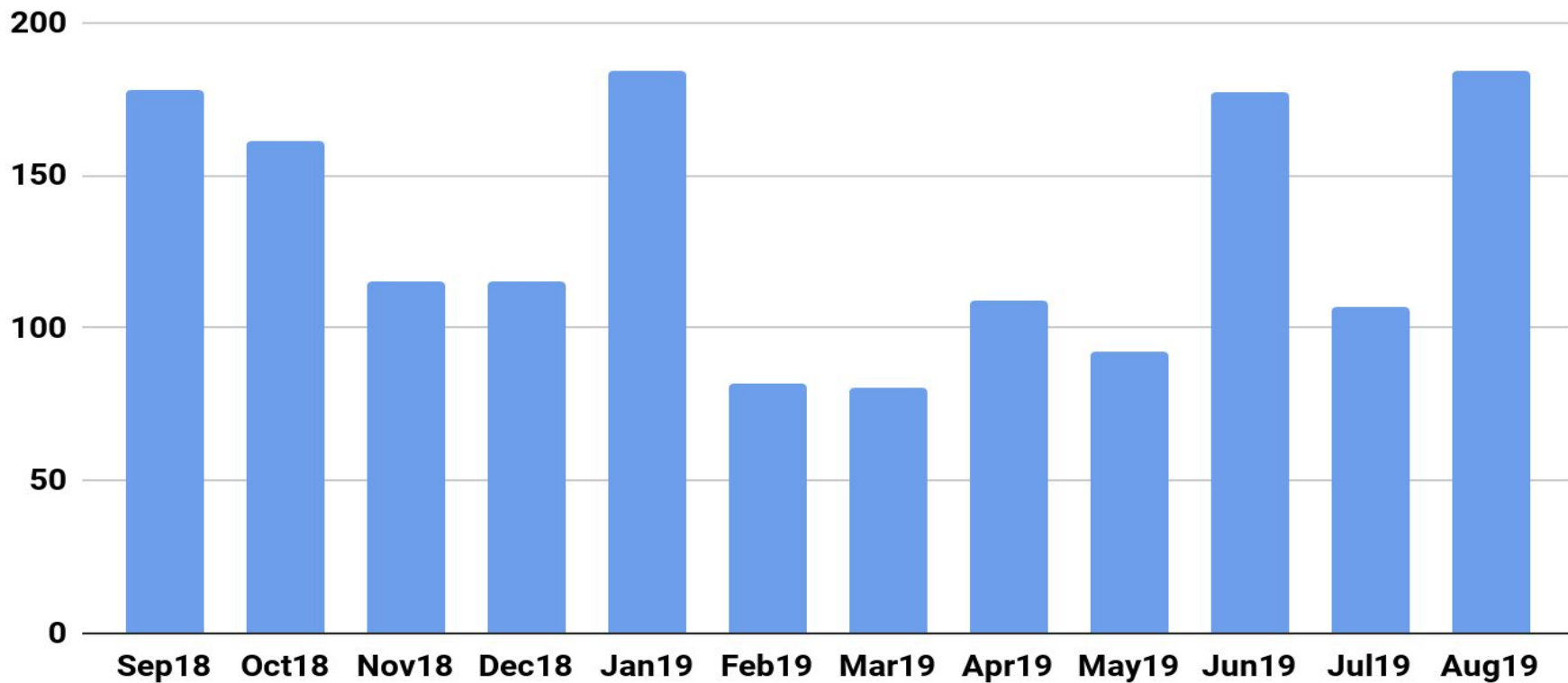
Open: 758

2 years: 3 bugs/day, 2 fixed

Open bugs

KASAN: invalid-free in iowarrior_disconnect			3	1d02h	1d11h
KASAN: invalid-free in rsi_91x_deinit	C		84	3d08h	119d
KASAN: slab-out-of-bounds Read in bacpy	C	cause	15	4h33m	225d
KASAN: slab-out-of-bounds Read in class_equal	syz	cause	79	13d	87d
KASAN: slab-out-of-bounds Read in hci_event_packet	C	cause	9	30d	225d
KASAN: slab-out-of-bounds Read in hidraw_ioctl	C		48	14h42m	21d
KASAN: slab-out-of-bounds Read in mceusb_dev_recv	C		2	6d00h	8d13h
KASAN: slab-out-of-bounds Write in ax_probe	C		4	2d23h	8d13h
KASAN: slab-out-of-bounds Write in check_noncircular	syz	cause	3	26d	32d
KASAN: slab-out-of-bounds Write in lg4ff_init	C		1	17d	15d
KASAN: use-after-free Read in adu_disconnect	C		236	57m	15d
KASAN: use-after-free Read in blkdev_bio_end_io	C	cause	13	3d16h	15d
KASAN: use-after-free Read in blkdev_direct_IO	C	cause	8	13d	18d
KASAN: use-after-free Read in ccid2_hc_tx_packet_recv	C		77	8d20h	505d
KASAN: use-after-free Read in debugfs_remove(3)	C	cause	77	2d10h	314d
KASAN: use-after-free Read in dvb_usb_device_exit(2)	C		64	2h28m	14d
KASAN: use-after-free Read in hidraw_ioctl	C		1394	1h10m	28d
KASAN: use-after-free Read in iowarrior_disconnect	C		299	7m	1d11h
KASAN: use-after-free Read in iowarrior_release	C		3	3d12h	1d11h
KASAN: use-after-free Read in kfree_skb(3)	C	cause	90	13h55m	105d
KASAN: use-after-free Read in nr_rx_frame(2)	C	cause	3	20d	28d

syzbot stats



The image features a teal abstract background with a white border. The background has a textured, wavy appearance, resembling water or a close-up of a rock surface. The white border is a thin line that frames the entire image. The text "new things" is centered in the white area.

new things

KMEMLEAK

- 3 months
 - 82 bugs reported
 - 44 fixed
- challenges:
 - slow
 - false positives
 - frequent bugs
 - reproducers

KMEMLEAK

- scan after a batch of tests
- report only if reproducible
- scanning is embed in C reproducers
- ignore leaks after first hit
- scanning:

KMEMLEAK

- scan after a batch of tests
- report only if reproducible
- scanning is embed in C reproducers
- ignore leaks after first hit
- scanning:

```
scan()  
sleep(4)  
scan()  
if (leaks) {  
    sleep(1)  
    scan()  
}
```


KMEMLEAK

All testing should use KMEMLEAK too!

KMSAN

KMSAN (KernelMemorySANitizer) -
detects **uses** of uninit **values**

16 months: 219 bugs reported, 105 fixed

not upstream (github.com/google/kmsan)

KMSAN bugs

KMSAN: kernel-infoleak in copy_siginfo_to_user

KMSAN: kernel-infoleak in video_usercopy

KMSAN: kernel-usb-infoleak in usbnet_write_cmd

KMSAN: uninit-value in tcp_create_openreq_child

KMSAN: uninit-value in aa_fqlookupn_profile

USB Fuzzing

- stress kernel from "external" size
- both external and userspace
- ~250 bugs reported, ~100 fixed
- 8400 device IDs

Bisection

Yay!

Bisection

Yay!

~50% success rate :(

Going back in time...

- v4.11: no gcc 7 ("undefined reference to ____ilog2_NaN")
- v4.1: no compiler-gcc5.h
- v3.17: no compiler-gcc4.h
- v3.8: modern perl fails ("Can't use defined(@array)")
 - but old perl fails on later kernels(Can't locate strict.pm in @INC)
- v3.6: no make olddefconfig
- v2.6.28: binutils fails (elf_x86_64: No such file or directory)
- v2.6.28: modern make fails ("mixed implicit and normal rules")

Going back in time...

- v4.10: no compat socket syscalls
- v4.9: no KASAN for atomicops
- v4.0: no KASAN
- ???: no LOCKDEP, FAULT_INJECT, etc

Going back in time...

- v5.2: boot broken SECURITY_TOMOYO_OMIT_...
- v4.15: boot broken USBIP_VUDC
- v4.13: boot broken CAN
- v4.12: runtime broken HSR & SMC
- v4.10: boot broken USBIP_VHCI & BT_HCIVHCI
- v4.7: runtime broken NET_TEAM
- v4.5: runtime broken BATMAN_ADV
- v4.0: random memory corruptions
- ...
- v2.6.28: build broken KVM

Bisection analysis

- 118 bisections
- ~50% success rate
 - 46% racy/flaky
 - 66% unrelated crashes
 - 55% have multiple manifestations
 - 14% broken build/boot
 - 8% disabled configs
- 70% success rate for latest releases



"Why don't you **just** bisect?"

Fix Bisection

- WIP
- fix bisect if no crashes for X days
 - no crash -> suggest to close the bug
 - crash -> ping

Lots of other work

- more OSes (gVisor, *BSD)
- more archs (PPC, ARM)
- description language improvements
 - offsetof
 - complex targets (len of parent_struct.foo.bar)
- special pointers (0xffffffff81000000, 0xffffffffffffffff)
- better sandboxing
- better OOPS parsing

Lots of other work

- CI improvements
- more tests
- test deflaking
- coverage reports
- static code analysis pre-commit
- fuzzing
- continuous fuzzing

Lots of other work

- split dashboard by kernel (fetching all bugs causes DB timeouts)
- better coverage reports
- auto-closing obsolete bugs
- auto-upstreaming bugs
- link to fixing commits
- more descriptions
- more configs enabled
- support quilt patch format (back to the future!)

The image features a teal abstract background on the left side, which transitions into a white area on the right. A thin white border is visible on the left side of the teal area. The word "Coverage" is written in a large, bold, black sans-serif font in the center of the white area.

Coverage

[fixed bugs \(1457\)](#)

Instances:

Name	Active	Uptime	Corpus	Coverage	Crashes	Execs	Kernel build	
							Commit	Freshness
ci-upstream-bpf-kasan-gce	now	5h00m	12495	333997	374	4765380	d34b0440	4d07h
ci-upstream-bpf-next-kasan-gce	now	5h00m	11911	349653	1223	3884894	1f726723	2d00h
ci-upstream-gce-leak	now	19m	32870	718698	165	1791489	06821504	5h11m
ci-upstream-kasan-gce	now	5h00m	38452	766481	79	9915553	d1abaeb3	1d00h
ci-upstream-kasan-gce-386	now	26m	27500	463469	27	4543471	06821504	5h11m
ci-upstream-kasan-gce-root	now	4h57m	38427	832759	70	8421211	06821504	5h11m
ci-upstream-kasan-gce-selinux-root	now	7m	37417	815585	76	6899496	06821504	5h11m
ci-upstream-kasan-gce-smack-root	now	5h00m	52248	597981	72	12315172	d1abaeb3	1d00h
ci-upstream-kmsan-gce	now	4h59m	47688	416927	1476	1157667	61ccdad1	12d
ci-upstream-linux-next-kasan-gce-root	now	5h00m	39650	879574	244	3590038	da657043	13h08m
ci-upstream-net-kasan-gce	now	5h00m	19327	438305	520	5831175	20e79a0a	2d19h
ci-upstream-net-this-kasan-gce	now	5h00m	19750	421161	278	6134519	cfef46d6	3d06h
ci2-upstream-usb	now	5h17m	2054	62989	516	825916	e06ce4da	39d

Coverage report

▶ arch/x86	24%	of 56169
▶ block	27%	of 17155
▶ certs	17%	of 48
▶ crypto	40%	of 11104
▶ drivers	5%	of 500972
▶ fs	14%	of 286044
▶ include	19%	of 38866
▶ init	---	of 1026
▶ ipc	62%	of 3212
▶ kernel	35%	of 65824
▶ lib	20%	of 24405
▶ mm	38%	of 40063
▶ net	25%	of 385603
▶ security	20%	of 29049
▶ sound	21%	of 32521
▶ virt	61%	of 2558

Coverage report

▶ arch/x86	24%	of 56169
▶ block	27%	of 17155
▶ certs	17%	of 48
▶ crypto	40%	of 11104
▶ drivers	5%	of 500972
▶ fs	14%	of 286044
▶ include	19%	of 38866
▶ init	---	of 1026
▶ ipc	62%	of 3212
▶ kernel	35%	of 65824
▶ lib	20%	of 24405
▶ mm	38%	of 40063
▶ net	25%	of 385603
▶ security	20%	of 29049
▶ sound	21%	of 32521
▶ virt	61%	of 2558

Coverage report

▶ arch/x86	24%	of 56169
▶ block	27%	of 17155
▶ certs	17%	of 48
▶ crypto	40%	of 11104
▶ drivers	5%	of 500972
▶ fs	14%	of 286044
▶ include	19%	of 38866
▶ init	---	of 1026
▶ ipc	62%	of 3212
▶ kernel	35%	of 65824
▶ lib	20%	of 24405
▶ mm	38%	of 40063
▶ net	25%	of 385603
▶ security	20%	of 29049
▶ sound	21%	of 32521
▶ virt	61%	of 2558

Coverage report

▼ security	20%	of 29049
▶ apparmor	12%	of 8940
▶ integrity	24%	of 2102
▶ keys	55%	of 3101
▼ safesetid	9%	of 151
lsm.c	25%	of 53
securityfs.c	---	of 98
▶ selinux	---	of 7128
▶ smack	---	of 1831
▶ tomoyo	42%	of 3271
▶ yama	46%	of 259
commoncap.c	83%	of 421
device_cgroup.c	7%	of 311
inode.c	4%	of 53
lsm_audit.c	7%	of 172
min_addr.c	---	of 10
security.c	51%	of 1299

security/apparmor/ipc.c

```
static int profile_tracer_perm(struct aa_profile *tracer,  
                               struct aa_label *tracee, u32 request,  
                               struct common_audit_data *sa)  
{  
    if (profile_unconfined(tracer))  
        return 0;  
  
    if (PROFILE_MEDIATES(tracer, AA_CLASS_PTRACE))  
        return profile_ptrace_perm(tracer, tracee, request, sa);  
  
    /* profile uses the old style capability check for ptrace */  
    if (&tracer->label == tracee)  
        return 0;  
  
    aad(sa)->label = &tracer->label;  
    aad(sa)->peer = tracee;  
    aad(sa)->request = 0;  
    aad(sa)->error = aa_capable(&tracer->label, CAP_SYS_PTRACE,  
                                CAP_OPT_NONE);  
  
    return aa_audit(AUDIT_APPARMOR_AUTO, tracer, sa, audit_ptrace_cb);  
}
```


security/apparmor/audit.c

```
27     if (KILL_MODE(profile) && type == AUDIT_APPARMOR_DENIED)
        type = AUDIT_APPARMOR_KILL;

27     aad(sa)->label = &profile->label;

    aa_audit_msg(type, sa, cb);

    if (aad(sa)->type == AUDIT_APPARMOR_KILL)
        (void)send_sig_info(SIGKILL, NULL,
            sa->type == LSM_AUDIT_DATA_TASK && sa->u.tsk ?
            sa->u.tsk : current);

17     if (aad(sa)->type == AUDIT_APPARMOR_ALLOWED)
        return complain_error(aad(sa)->error);

    return aad(sa)->error;
```

security/apparmor/path.c

```
/* Handle two cases:
 * 1. A deleted dentry && profile is not allowing mediation of deleted
 * 2. On some filesystems, newly allocated dentries appear to the
 *    security_path hooks as a deleted dentry except without an inode
 *    allocated.
 */
171 if (d_unlinked(path->dentry) && d_is_positive(path->dentry) &&
3    !(flags & (PATH_MEDIATE_DELETED | PATH_DELEGATE_DELETED))) {
    error = -ENOENT;
    goto out;
}
```

Things may be confusing

```
if (aad(sa)->peer) {  
    audit_log_format(ab, " target=");  
    aa_label_xaudit(ab, labels_ns(aad(sa)->label), aad(sa)->peer,  
                    FLAG_VIEW_SUBNS, GFP_ATOMIC);  
}
```

Things may be confusing

```
rcu_read_lock();  
tracer = ptrace_parent(current);
```

Things may be confusing

```
27     if (KILL_MODE(profile) && type == AUDIT_APPARMOR_DENIED)
        type = AUDIT_APPARMOR_KILL;

27     aad(sa)->label = &profile->label;

    aa_audit_msg(type, sa, cb);

    if (aad(sa)->type == AUDIT_APPARMOR_KILL)
        (void)send_sig_info(SIGKILL, NULL,
            sa->type == LSM_AUDIT_DATA_TASK && sa->u.tsk ?
            sa->u.tsk : current);

17     if (aad(sa)->type == AUDIT_APPARMOR_ALLOWED)
        return complain_error(aad(sa)->error);

    return aad(sa)->error;
```


Coverage caveats

- only synchronous syscall code
- no background threads
- no interrupts
- no init code

Coverage report

▶ arch/x86	24%	of 56169
▶ block	27%	of 17155
▶ certs	17%	of 48
▶ crypto	40%	of 11104
▶ drivers	5%	of 500972
▶ fs	14%	of 286044
▶ include	19%	of 38866
▶ init	---	of 1026
▶ ipc	62%	of 3212
▶ kernel	35%	of 65824
▶ lib	20%	of 24405
▶ mm	38%	of 40063
▶ net	25%	of 385603
▶ security	20%	of 29049
▶ sound	21%	of 32521
▶ virt	61%	of 2558

Coverage report

▶ arch/x86	24%	of 56169
▶ block	27%	of 17155
▶ certs	17%	of 48
▶ crypto	40%	of 11104
▶ drivers	5%	of 500972
▶ fs	14%	of 286044
▶ include	19%	of 38866
▶ init	---	of 1026
▶ ipc	62%	of 3212
▶ kernel	35%	of 65824
▶ lib	20%	of 24405
▶ mm	38%	of 40063
▶ net	25%	of 385603
▶ security	20%	of 29049
▶ sound	21%	of 32521
▶ virt	61%	of 2558

Coverage report

▶ arch/x86	24%	of 56169
▶ block	27%	of 17155
▶ certs	17%	of 48
▶ crypto	40%	of 11104
▶ drivers	5%	of 500972
▶ fs	14%	of 286044
▶ include	19%	of 38866
▶ init	---	of 1026
▶ ipc	62%	of 3212
▶ kernel	35%	of 65824
▶ lib	20%	of 24405
▶ mm	38%	of 40063
▶ net	25%	of 385603
▶ security	20%	of 29049
▶ sound	21%	of 32521
▶ virt	61%	of 2558

▶ ecryptfs	2%	of 1904
▶ efivarfs	---	of 141
▶ exportfs	50%	of 166
▶ ext4	46%	of 19326
▶ f2fs	3%	of 18005
▶ fat	57%	of 2860
▶ fscache	8%	of 2504
▶ fuse	32%	of 3567
▶ gfs2	2%	of 10877
▶ hfs	10%	of 1361
▶ hfsplus	5%	of 2249
▶ hugetlbfs	45%	of 400

net/

▶ ife	---	of 30
▶ ipv4	51%	of 40105
▶ ipv6	46%	of 28958
▶ kcm	54%	of 894
▶ key	75%	of 1359
▶ l2tp	44%	of 2227
▶ l3mdev	49%	of 96
▶ lapb	9%	of 404
▶ llc	29%	of 1683
▶ mac80211	2%	of 31461
▶ mac802154	1%	of 2206
▶ mpls	25%	of 1270

Total Coverage

251'405 out of 1'497'611

17%*

~1 MLOC

The image features a teal abstract background with a white border. The text "Increasing coverage" is centered in a bold, black, sans-serif font.

Increasing coverage

More descriptions

```
resource fd_floppy[fd]
```

```
open(dev ptr[in, string["/dev/fd0"]], ...) fd_floppy
```

```
ioctl(fd fd_floppy, cmd const[FDEJECT])
```

```
ioctl(fd fd_floppy, cmd const[FDSETPRM],  
      arg ptr[in, floppy_struct])
```

```
floppy_struct {  
    size    int32  
    sect    int32  
    ... }  
}
```


More descriptions

resource **fd_floppy**[fd]

```
open(dev ptr[in, string["/dev/fd0"]], ...) fd_floppy
```

```
ioctl(fd fd_floppy, cmd const[FDEJECT])
```

```
ioctl(fd fd_floppy, cmd const[FDSETPRM],  
      arg ptr[in, floppy_struct])
```

```
floppy_struct {  
    size    int32  
    sect    int32  
    ...}
```

More descriptions

```
resource fd_floppy[fd]
```

```
open(dev ptr[in, string["/dev/fd0"]], ...) fd_floppy
```

```
ioctl(fd fd_floppy, cmd const[FDEJECT])
```

```
ioctl(fd fd_floppy, cmd const[FDSETPRM],  
      arg ptr[in, floppy_struct])
```

```
floppy_struct {  
    size    int32  
    sect    int32  
    ... }  
}
```

More descriptions

```
resource fd_floppy[fd]
```

```
open(dev ptr[in, string["/dev/fd0"]], ...) fd_floppy
```

```
ioctl(fd fd_floppy, cmd const[FDEJECT])
```

```
ioctl(fd fd_floppy, cmd const[FDSETPRM],  
      arg ptr[in, floppy_struct])
```

```
floppy_struct {  
    size    int32  
    sect    int32  
    ... }
```

More descriptions

```
resource fd_floppy[fd]
```

```
open(dev ptr[in, string["/dev/fd0"]], ...) fd_floppy
```

```
ioctl(fd fd_floppy, cmd const[FDEJECT])
```

```
ioctl(fd fd_floppy, cmd const[FDSETPRM],  
arg ptr[in, floppy_struct])
```

```
floppy_struct {  
    size    int32  
    sect    int32  
    ...}
```

More descriptions

```
resource fd_floppy[fd]
```

```
open(dev ptr[in, string["/dev/fd0"]], ...) fd_floppy
```

```
ioctl(fd fd_floppy, cmd const[FDEJECT])
```

```
ioctl(fd fd_floppy, cmd const[FDSETPRM],  
      arg ptr[in, floppy_struct])
```

```
floppy_struct {  
    size    int32  
    sect    int32  
    ... }  
}
```

More surface

- CONFIGs
- sysctl's
- per-test setup (netdev, cgroups)
- ???

Interface WTFs

Linked lists?

1. `set_robust_list`

Interface WTFs

Linked lists?

1. `set_robust_list`

```
2. struct v4l2_clip {  
    struct v4l2_rect      c;  
    struct v4l2_clip__user *next;  
};
```

The image features a teal abstract background on the left side, which transitions into a white area on the right. A thin white border is visible around the white area. The text "video4linux, WHY?" is centered in the white area.

video4linux, WHY?

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;
}
```

Dealing with linked lists

... 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.

WTF: user pointers as IDs

Need to pass the same pointer **value**:

```
io_submit(..., struct iocb __user *iocbp);  
io_cancel(..., struct iocb __user *iocbp);
```

WTF: user pointers as IDs

```
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);  
}
```


WTF: user pointers as IDs

"How do I describe a kernel mutex?"

WTF: variable-size-elem array length

WTF: variable-size-elem array length

```
snd_ctl_elem_info {  
    names_ptr      ptr64[in, array[string]]  
    names_length   bytesize[names_ptr, int32]  
    items          len[names_ptr, int32]  
}
```

WTF: variable-size-elem array length

```
snd_ctl_elem_info {  
    names_ptr      ptr64[in, array[string]]  
    names_length   bytesize[names_ptr, int32]  
    items          len[names_ptr, int32]  
}
```

WTF: variable-size-elem array length

```
snd_ctl_elem_info {  
    names_ptr      ptr64[in, array[string]]  
    names_length   bytesize[names_ptr, int32]  
    items          len[names_ptr, int32]  
}
```

WTF: variable-size-elem array length

NETFILTER

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_counters
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

```
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  
...
```

Stick to existing practices

- no linked lists (arrays)
- no pointers as IDs (fd/idr)
- no secondary info
- ...

Stick to existing practices

- no linked lists (arrays)
- no pointers as IDs (fd/idr)
- no secondary info
- ...

require upfront formal descriptions?

A vertical teal abstract background with a textured, wavy pattern is located on the left side of the slide.

Testable code

Stub devices

- tun
- vcan
- veth
- vivid/vimc/vim2m/vicodec

Stub devices

- tun
- vcan
- veth
- vivid/vimc/vim2m/vicodec

Not just fuzzing, also **testing!**

Best practices

- namespace-friendly (not just init ns)
- dynamic creation (not one/fixed number)
- isolation (brand new instance)
- handy programmatic interface
- raw interface
 - no sanity checks
 - no second guessing
 - no hard-coded behavior

SIGKILL

Can you SIGKILL any process?

Yes.

Except for 1 case: FUSE.

Parsing kernel output

WARN for non-bugs

```
pr_err("WARNING: ...")
```

```
pr_err("my custom error type here: ...")
```

Wishes

- fix bugs
- more coverage
- standard interfaces
- testable code



Thank you!

Q&A

Dmitry Vyukov, @dvyukov



LINUX SECURITY SUMMIT