syzbot

update and open problems

Dmitry Vyukov, dvyukov@
Linux Plumbers 2019

syzkaller + syzbot

syzkaller - OS kernel fuzzer:

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

syzbot - syzkaller automation:

- continuous kernel/syzkaller update
- bug reporting / tracking
- web dashboard

syzkaller.appspot.com

syzbot stats

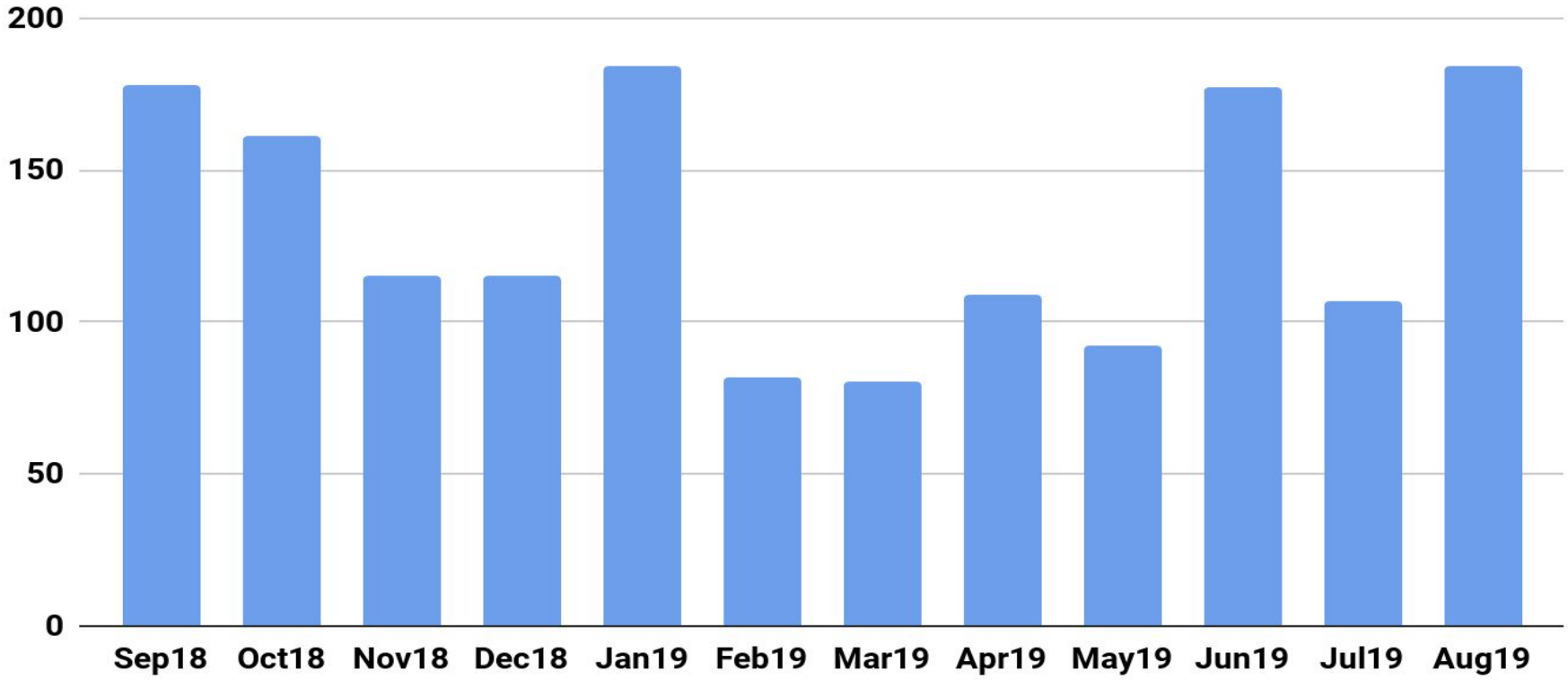
Reported:	2281
Fixed:	1523 (66.7%)
Open:	758

syzbot stats

Reported: 2281
Fixed: 1523 (66.7%)
Open: 758

2 years: 3 bugs/day, 2 fixed

Bugs / month


200
150
100
50
0
Sep18
Oct18
Nov18
Dec18
Jan19
Feb19
Mar19
Apr19
May19
Jun19
Jul19
Aug19

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

KMEMLEAK

- 3 months
  - 82 bugs reported
  - 44 fixed

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()
}
```

All testing should use KMEMLEAK too!

Systematic fault injection

`/proc/thread-self/fail-nth`

`write N: fails N-th fault site (kmalloc) in the task`

`read: check if the fault was injected`

Systematic fault injection

```
for (int i = 0;; i++) {
    write("/proc/thread-self/fail-nth", i);
    execute_syscall();
    if (read("/proc/thread-self/fail-nth") != "0") break;
}
```

All testing should use fault injection!*

*with ptrace
**inject faults, but ignore exit status

The virgin lands of error paths.

KMSAN

KMSAN - detects **uses** of uninit **values**

16 months: 219 bugs reported, 105 fixed

```
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
- `/sys/kernel/debug/usb-fuzzer`
- ~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, 0x9999999999999999)
- 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!)

Coverage

syzbot

admin | mailing list

Linux

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

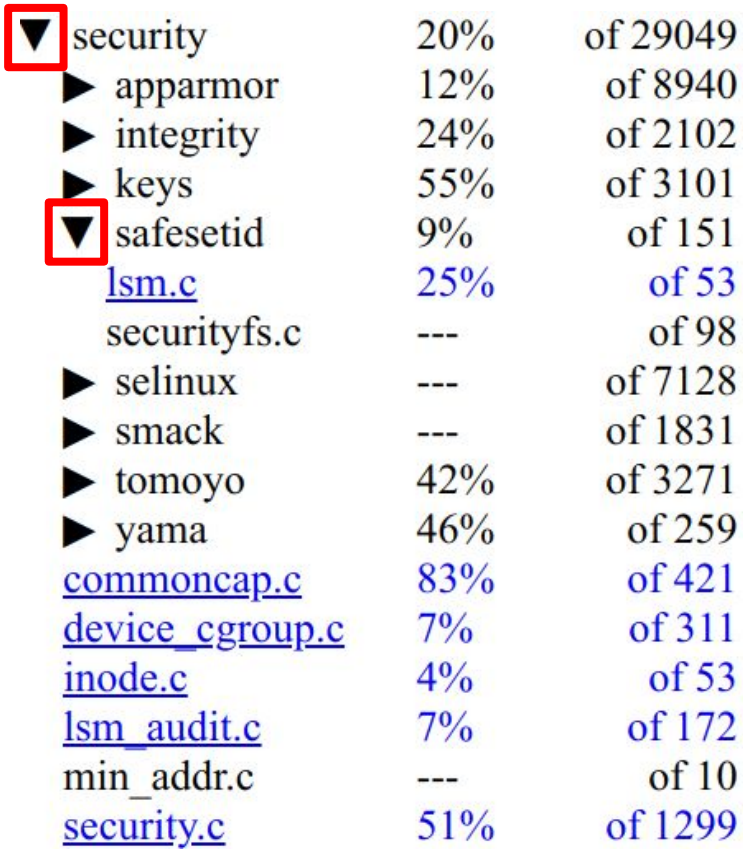
Coverage report

▶ 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/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.
 */
if (d_unlinked(path->dentry) && d_is_positive(path->dentry) &&
    !(flags & (PATH_MEDIATE_DELETED | PATH_DELEGATE_DELETED))) {
                error = -ENOENT;
                goto out;
}
```

171
3

Coverage caveats

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

Total Coverage

251'405 out of 3'454'974

7%

~1.8 MLOC

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
...}
```

Stub devices

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

Stub devices

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

Not just fuzzing, also **testing**!

Thanks!

Q&A

Dmitry Vyukov, dvyukov@