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

![Bar chart showing the number of bugs per month from Sep 2018 to Aug 2019. The highest number of bugs was in Jan 19, and the lowest was in Mar 19.](chart_url)
## Open bugs

<table>
<thead>
<tr>
<th>Description</th>
<th>Language</th>
<th>Cause</th>
<th>Days Open</th>
<th>Days Fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>KASAN: invalid-free in iowarrior_disconnect</td>
<td></td>
<td></td>
<td>3</td>
<td>1d02h</td>
</tr>
<tr>
<td>KASAN: invalid-free in rsi_91x_deinit</td>
<td>C</td>
<td></td>
<td>84</td>
<td>3d08h</td>
</tr>
<tr>
<td>KASAN: slab-out-of-bounds Read in bactpy</td>
<td>C</td>
<td>cause</td>
<td>15</td>
<td>4h33m</td>
</tr>
<tr>
<td>KASAN: slab-out-of-bounds Read in class_equal</td>
<td>syz</td>
<td>cause</td>
<td>79</td>
<td>13d</td>
</tr>
<tr>
<td>KASAN: slab-out-of-bounds Read in hci_event_packet</td>
<td>C</td>
<td>cause</td>
<td>9</td>
<td>30d</td>
</tr>
<tr>
<td>KASAN: slab-out-of-bounds Read in hidraw_ioctl</td>
<td>C</td>
<td></td>
<td>48</td>
<td>14h42m</td>
</tr>
<tr>
<td>KASAN: slab-out-of-bounds Read in mceusb_dev_recv</td>
<td>C</td>
<td></td>
<td>2</td>
<td>6d00h</td>
</tr>
<tr>
<td>KASAN: slab-out-of-bounds Write in ax_probe</td>
<td>C</td>
<td></td>
<td>4</td>
<td>2d23h</td>
</tr>
<tr>
<td>KASAN: slab-out-of-bounds Write in check_noncircular</td>
<td>syz</td>
<td>cause</td>
<td>3</td>
<td>26d</td>
</tr>
<tr>
<td>KASAN: slab-out-of-bounds Write in lg4ff_init</td>
<td>C</td>
<td></td>
<td>1</td>
<td>17d</td>
</tr>
<tr>
<td>KASAN: use-after-free Read in adu_disconnect</td>
<td>C</td>
<td></td>
<td>236</td>
<td>57m</td>
</tr>
<tr>
<td>KASAN: use-after-free Read in blkdev_bio_end_io</td>
<td>C</td>
<td>cause</td>
<td>13</td>
<td>3d16h</td>
</tr>
<tr>
<td>KASAN: use-after-free Read in blkdev_direct_IO</td>
<td>C</td>
<td>cause</td>
<td>8</td>
<td>13d</td>
</tr>
<tr>
<td>KASAN: use-after-free Read in ccid2_hc_tx_packet_recv</td>
<td>C</td>
<td>cause</td>
<td>77</td>
<td>8d20h</td>
</tr>
<tr>
<td>KASAN: use-after-free Read in debugfs_remove (3)</td>
<td>C</td>
<td>cause</td>
<td>77</td>
<td>2d10h</td>
</tr>
<tr>
<td>KASAN: use-after-free Read in dvb_usb_device_exit (2)</td>
<td>C</td>
<td></td>
<td>64</td>
<td>2h28m</td>
</tr>
<tr>
<td>KASAN: use-after-free Read in hidraw_ioctl</td>
<td>C</td>
<td></td>
<td>1394</td>
<td>1h10m</td>
</tr>
<tr>
<td>KASAN: use-after-free Read in iowarrior_disconnect</td>
<td>C</td>
<td></td>
<td>299</td>
<td>7m</td>
</tr>
<tr>
<td>KASAN: use-after-free Read in iowarrior_release</td>
<td>C</td>
<td></td>
<td>3</td>
<td>3d12h</td>
</tr>
<tr>
<td>KASAN: use-after-free Read in kfree_skb (3)</td>
<td>C</td>
<td>cause</td>
<td>90</td>
<td>13h55m</td>
</tr>
<tr>
<td>KASAN: use-after-free Read in nr_rx_frame (2)</td>
<td>C</td>
<td>cause</td>
<td>3</td>
<td>20d</td>
</tr>
</tbody>
</table>
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:

```c
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

```c
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
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 (0xfffffffff81000000, 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
<table>
<thead>
<tr>
<th>Name</th>
<th>Active</th>
<th>Uptime</th>
<th>Corpus</th>
<th>Coverage</th>
<th>Crashes</th>
<th>Execs</th>
<th>Kernel build</th>
<th>Freshness</th>
</tr>
</thead>
<tbody>
<tr>
<td>ci-upstream-bpf-kasan-gce</td>
<td>now</td>
<td>5h00m</td>
<td>12495</td>
<td>333997</td>
<td>374</td>
<td>4765380</td>
<td>d34b0440</td>
<td>4d07h</td>
</tr>
<tr>
<td>ci-upstream-bpf-next-kasan-gce</td>
<td>now</td>
<td>5h00m</td>
<td>11911</td>
<td>349653</td>
<td>1223</td>
<td>3884894</td>
<td>1f726723</td>
<td>2d00h</td>
</tr>
<tr>
<td>ci-upstream-gce-leak</td>
<td>now</td>
<td>19m</td>
<td>32870</td>
<td>718698</td>
<td>165</td>
<td>1791489</td>
<td>06821504</td>
<td>5h11m</td>
</tr>
<tr>
<td>ci-upstream-kasan-gce</td>
<td>now</td>
<td>5h00m</td>
<td>38452</td>
<td>766481</td>
<td>79</td>
<td>9915553</td>
<td>d1abaeb3</td>
<td>1d00h</td>
</tr>
<tr>
<td>ci-upstream-kasan-gce-386</td>
<td>now</td>
<td>26m</td>
<td>27500</td>
<td>463469</td>
<td>27</td>
<td>4543471</td>
<td>06821504</td>
<td>5h11m</td>
</tr>
<tr>
<td>ci-upstream-kasan-gce-root</td>
<td>now</td>
<td>4h57m</td>
<td>38427</td>
<td>832759</td>
<td>70</td>
<td>8421211</td>
<td>06821504</td>
<td>5h11m</td>
</tr>
<tr>
<td>ci-upstream-kasan-gce-selinux-root</td>
<td>now</td>
<td>7m</td>
<td>37417</td>
<td>815585</td>
<td>76</td>
<td>6899946</td>
<td>06821504</td>
<td>5h11m</td>
</tr>
<tr>
<td>ci-upstream-kasan-gce-smack-root</td>
<td>now</td>
<td>5h00m</td>
<td>52248</td>
<td>597981</td>
<td>72</td>
<td>12315172</td>
<td>d1abaeb3</td>
<td>1d00h</td>
</tr>
<tr>
<td>ci-upstream-kmsan-gce</td>
<td>now</td>
<td>4h59m</td>
<td>47688</td>
<td>416927</td>
<td>1476</td>
<td>1157667</td>
<td>61ccdad1</td>
<td>12d</td>
</tr>
<tr>
<td>ci-upstream-linux-next-kasan-gce-root</td>
<td>now</td>
<td>5h00m</td>
<td>39650</td>
<td>879574</td>
<td>244</td>
<td>3590038</td>
<td>da657043</td>
<td>13h08m</td>
</tr>
<tr>
<td>ci-upstream-net-kasan-gce</td>
<td>now</td>
<td>5h00m</td>
<td>19327</td>
<td>438305</td>
<td>520</td>
<td>5831175</td>
<td>20e79a0a</td>
<td>2d19h</td>
</tr>
<tr>
<td>ci-upstream-net-this-kasan-gce</td>
<td>now</td>
<td>5h00m</td>
<td>19750</td>
<td>421161</td>
<td>278</td>
<td>6134519</td>
<td>cef46d6</td>
<td>3d06h</td>
</tr>
<tr>
<td>ci2-upstream-usb</td>
<td>now</td>
<td>5h17m</td>
<td>2054</td>
<td>62989</td>
<td>516</td>
<td>825916</td>
<td>e06ce4da</td>
<td>39d</td>
</tr>
</tbody>
</table>
### Coverage report

<table>
<thead>
<tr>
<th>Directory</th>
<th>Coverage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch/x86</td>
<td>24%</td>
<td>56169</td>
</tr>
<tr>
<td>block</td>
<td>27%</td>
<td>17155</td>
</tr>
<tr>
<td>certs</td>
<td>17%</td>
<td>48</td>
</tr>
<tr>
<td>crypto</td>
<td>40%</td>
<td>11104</td>
</tr>
<tr>
<td>drivers</td>
<td>5%</td>
<td>500972</td>
</tr>
<tr>
<td>fs</td>
<td>14%</td>
<td>286044</td>
</tr>
<tr>
<td>include</td>
<td>19%</td>
<td>38866</td>
</tr>
<tr>
<td>init</td>
<td>---</td>
<td>1026</td>
</tr>
<tr>
<td>ipc</td>
<td>62%</td>
<td>3212</td>
</tr>
<tr>
<td>kernel</td>
<td>35%</td>
<td>65824</td>
</tr>
<tr>
<td>lib</td>
<td>20%</td>
<td>24405</td>
</tr>
<tr>
<td>mm</td>
<td>38%</td>
<td>40063</td>
</tr>
<tr>
<td>net</td>
<td>25%</td>
<td>385603</td>
</tr>
<tr>
<td>security</td>
<td>20%</td>
<td>29049</td>
</tr>
<tr>
<td>sound</td>
<td>21%</td>
<td>32521</td>
</tr>
<tr>
<td>virt</td>
<td>61%</td>
<td>2558</td>
</tr>
</tbody>
</table>
# Coverage report

<table>
<thead>
<tr>
<th>Directory</th>
<th>Coverage</th>
<th>Total Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch/x86</td>
<td>24%</td>
<td>56169</td>
</tr>
<tr>
<td>block</td>
<td>27%</td>
<td>17155</td>
</tr>
<tr>
<td>certs</td>
<td>17%</td>
<td>48</td>
</tr>
<tr>
<td>crypto</td>
<td>40%</td>
<td>11104</td>
</tr>
<tr>
<td>drivers</td>
<td>5%</td>
<td>500972</td>
</tr>
<tr>
<td>fs</td>
<td>14%</td>
<td>286044</td>
</tr>
<tr>
<td>include</td>
<td>19%</td>
<td>38866</td>
</tr>
<tr>
<td>init</td>
<td>---</td>
<td>1026</td>
</tr>
<tr>
<td>ipc</td>
<td>62%</td>
<td>3212</td>
</tr>
<tr>
<td>kernel</td>
<td>35%</td>
<td>65824</td>
</tr>
<tr>
<td>lib</td>
<td>20%</td>
<td>24405</td>
</tr>
<tr>
<td>mm</td>
<td>38%</td>
<td>40063</td>
</tr>
<tr>
<td>net</td>
<td>25%</td>
<td>385603</td>
</tr>
<tr>
<td>security</td>
<td>20%</td>
<td>29049</td>
</tr>
<tr>
<td>sound</td>
<td>21%</td>
<td>32521</td>
</tr>
<tr>
<td>virt</td>
<td>61%</td>
<td>2558</td>
</tr>
</tbody>
</table>
## Coverage report

<table>
<thead>
<tr>
<th>Directory</th>
<th>Coverage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch/x86</td>
<td>24%</td>
<td>56169</td>
</tr>
<tr>
<td>block</td>
<td>27%</td>
<td>17155</td>
</tr>
<tr>
<td>certs</td>
<td>17%</td>
<td>48</td>
</tr>
<tr>
<td>crypto</td>
<td>40%</td>
<td>11104</td>
</tr>
<tr>
<td>drivers</td>
<td>5%</td>
<td>500972</td>
</tr>
<tr>
<td>fs</td>
<td>14%</td>
<td>286044</td>
</tr>
<tr>
<td>include</td>
<td>19%</td>
<td>38866</td>
</tr>
<tr>
<td>init</td>
<td>---</td>
<td>1026</td>
</tr>
<tr>
<td>ipc</td>
<td>62%</td>
<td>3212</td>
</tr>
<tr>
<td>kernel</td>
<td>35%</td>
<td>65824</td>
</tr>
<tr>
<td>lib</td>
<td>20%</td>
<td>24405</td>
</tr>
<tr>
<td>mm</td>
<td>38%</td>
<td>40063</td>
</tr>
<tr>
<td>net</td>
<td>25%</td>
<td>385603</td>
</tr>
<tr>
<td>security</td>
<td>20%</td>
<td>29049</td>
</tr>
<tr>
<td>sound</td>
<td>21%</td>
<td>32521</td>
</tr>
<tr>
<td>virt</td>
<td>61%</td>
<td>2558</td>
</tr>
</tbody>
</table>
## Coverage report

<table>
<thead>
<tr>
<th>Module</th>
<th>Coverage</th>
<th>Total Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>security</td>
<td>20%</td>
<td>29049</td>
</tr>
<tr>
<td>apparmor</td>
<td>12%</td>
<td>8940</td>
</tr>
<tr>
<td>integrity</td>
<td>24%</td>
<td>2102</td>
</tr>
<tr>
<td>keys</td>
<td>55%</td>
<td>3101</td>
</tr>
<tr>
<td>safesetid</td>
<td>9%</td>
<td>151</td>
</tr>
<tr>
<td>lsm.c</td>
<td>25%</td>
<td>53</td>
</tr>
<tr>
<td>securityfs.c</td>
<td>---</td>
<td>98</td>
</tr>
<tr>
<td>selinux</td>
<td>---</td>
<td>7128</td>
</tr>
<tr>
<td>smack</td>
<td>---</td>
<td>1831</td>
</tr>
<tr>
<td>tomoyo</td>
<td>42%</td>
<td>3271</td>
</tr>
<tr>
<td>yama</td>
<td>46%</td>
<td>259</td>
</tr>
<tr>
<td>commoncap.c</td>
<td>83%</td>
<td>421</td>
</tr>
<tr>
<td>device_cgroup.c</td>
<td>7%</td>
<td>311</td>
</tr>
<tr>
<td>inode.c</td>
<td>4%</td>
<td>53</td>
</tr>
<tr>
<td>lsm_audit.c</td>
<td>7%</td>
<td>172</td>
</tr>
<tr>
<td>min_addr.c</td>
<td>---</td>
<td>10</td>
</tr>
<tr>
<td>security.c</td>
<td>51%</td>
<td>1299</td>
</tr>
</tbody>
</table>
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);
}
if (KILL_MODE(profile) && type == AUDIT_APPARMOR_DENIED)
    type = AUDIT_APPARMOR_KILL;

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);

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

return aad(sa)->error;
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.

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

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

ioctl(fd fd_floppy, cmd const[FDEJECT])
ioctl(fd fd_floppy, cmd const[FDSERPRM],
       arg ptr[in, floppy_struct])

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

resource fd_floppy

```
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[FDSERETPRM],
        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@