syzkaller

Mark Johnston
markj@FreeBSD.org

FreeBSD

FreeBSD Bay Area Vendor Summit
October 12, 2019
System Call Fuzzing: What?

- Common syscall usage patterns cover a small space
  - Why would you ever call `send(2)` after `listen(2)`?
- Increase coverage by generating and executing programs
- Look for crashes, hangs, sanitizer reports, etc.
- Cannot easily validate positive results

```c
for (;;) {
    p = generate_prog();
    execute(p);
}
```
System Call Fuzzing: Why?

- Kernel is part of the TCB
- System calls present a huge attack surface
- Jails and Capsicum help but are not sufficient
- FreeBSD has 500 system calls
  - Plus COMPAT_FREEBSD32, COMPAT_LINUX...
  - Plus de-muxing via ioctl(2), fcntl(2), setsockopt(2)...
- Fine-grained parallelism makes things much worse
System Call Fuzzing: How?

- Naive fuzzing mostly catches input validation bugs
- Can do better with semantic knowledge of syscall params
- Idea: use code coverage as input to test case generation

```c
for (cov = NULL;;) {
    p = generate_prog(corpus);
    cov1 = execute(p);
    if (!cov.contains(cov1)) {
        cov.add(cov1);
        corpus.add(p);
    }
}
```
Introduction to syzkaller

- “Unsupervised, coverage-guided kernel fuzzer”
- By Dmitry Vyukov at Google, initially for Linux
- [https://github.com/google/syzkaller/docs](https://github.com/google/syzkaller/docs)
- Kitchen sink approach:
  - Manages VMs running target kernels
  - Generates minimal reproducibles
  - Can inject network, USB, etc. packets
  - Collects, summarizes and deduplicates crash reports
  - Collects kernel code coverage info
  - Presents crash reports and test cases in a web dashboard
  - syz-ci periodically rebuilds kernel and syzkaller itself
  - Checks for regressions
  - Bisects new crashes
  - ...
syzkaller on FreeBSD

- syz-manager
- corpus, crash reports
- ssh, scp
- syz-fuzzer
- syz-executor
- VMs: bhyve, ZFS
- syscalls: /dev/kcov
- syz-prog2c
- .c files
- netdumpd
- vmcores

Build process:
- buildkernel
- gmake

-flowchart:
KCOV

- Thin user interface around LLVM SanitizerCoverage for kernel
- Initial implementation by mhorne®, finished by andrew®
- Open /dev/kcov and mmap to create shared buffer
- KIOENABLE ioctl enables tracing for the calling thread
- Buffer entries generated for every edge and comparison

```
include "./GENERIC"

ident SYZKALLER
options COVERAGE
options KCOV
```
System Call Descriptions

- syzkaller defines a syscall description grammar
- Supports “enhanced” types: flags, file descriptors, ...
- Implements compound types
- Each system call needs to be described - lots of work
- Some system calls have multiple flavours, e.g. `connect(2)`

```c
#include <fcntl.h>

open(file ptr[in, filename], flags flags[open_flags], mode flags[open_mode]) fd
open_flags = O_RDONLY, O_WRONLY, O_RDWR, O_APPEND, ...
open_mode = S_IRUSR, S_IWUSR, ...

stat {
    dev    int64
    ino    int64
    nlink  int64
    mode   int16
    __pad0 const[0, int16]
    uid    uid
    gid    gid
    ...
}
```
Sample Reproducer

#{"threaded":true,"collide":true,"repeat":true,"procs":4,"sandbox":"none","fault_call":-1,"tmpdir":true,"segv":true}

r0 = socket(0x2, 0x10000001, 0x84)
connect$unix(r0, &(0x7f0000000000)=@file={0xbd5699bc1ec0282, './file0\x00'}, 0x10)
getsockopt$inet6_sctp_SCTP_ENABLE_STREAM_RESET(r0, 0x84, 0x900,
   &(0x7f0000000080)={<r1=>0x0, 0x4},
   &(0x7f00000000c0)=0x8)
getsockopt$inet6_sctp_SCTP_DELAYED_SACK(r0, 0x84, 0xf, &(0x7f0000000180)={r1, 0x9, 0x6},
   &(0x7f00000001c0)=0xc)
listen(r0, 0x9)
setsockopt$inet6_sctp_SCTP_EVENTS(r0, 0x84, 0xc, &(0x7f0000000040)={0x0, 0x0, 0x0, 0x6}, 0xb)
setsockopt$inet6_sctp_SCTP_RTOINFO(r0, 0x84, 0x1, &(0x7f0000000100)={0x0, 0x0, 0x80000001}, 0x10)
shutdown(r0, 0x1)

Run with sudo syz-execprog ./repro.syz
Hosted CI for syzkaller, on GCE

https://syzkaller.appspot.com

Fuzzes many different operating systems

Thousands of bugs found

Mails syzkaller-freebsd-bugs@googlegroups.com when a new crash is found

Resolve reports automatically using a Reported-by tag:

commit fb4ce630e036f6b73bef06c3c4b9c7bf363a9b23
Author: markj <markj@FreeBSD.org>
Date:   Mon Mar 25 21:38:58 2019 +0000

Reject F_SETLK_REMOTE commands when sysid == 0.

A sysid of 0 denotes the local system, and some handlers for remote locking commands do not attempt to deal with local locks. Note that F_SETLK_REMOTE is only available to privileged users as it is intended to be used as a testing interface.

Reviewed by:   kib
Reported by:   syzbots9c457a6ae014a3281eb8@syzkaller.appspotmail.com
MFC after:     2 weeks
Sponsored by:  The FreeBSD Foundation
Differential Revision:  https://reviews.freebsd.org/D19702
syzkaller does not do a perfect job generating reproducers:
  - Some panics happen asynchronously (e.g., in a callout)
  - Some reproducers do not work (race conditions)
  - Reproducer minimization is not perfect or reliable

- VM disk image is discarded during reboot
- netdump(4) to the rescue
FreeBSD and syzkaller

Why is it worth investing time into syzkaller?
What do we need?
- Bug triage and analysis
- More system call descriptions
- Fuzzing ZFS, NFS-based images
- Fuzzing non-amd64 kernels
- syzkaller jail image
- Sanitizer support