

Анатомия фаззинга для ядра Линукса

Пелевин Максим

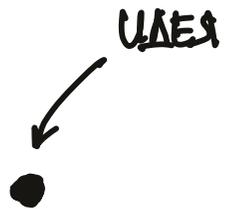
Пелевин Максим

- Разработчик более 10 лет
- Область работы:
динамический анализ кода
- Хобби: преподаватель
в университете

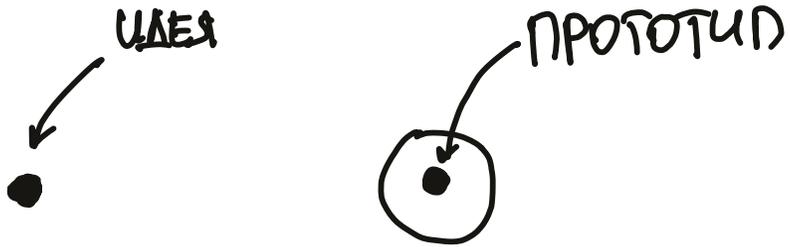


Преамбула

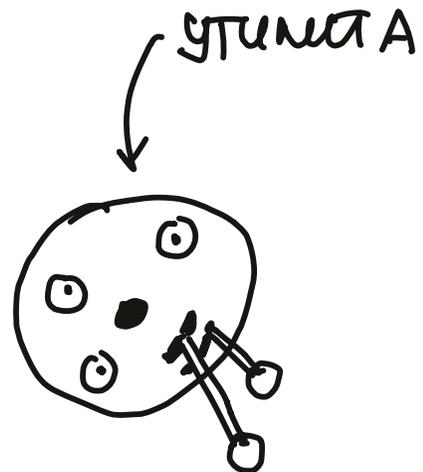
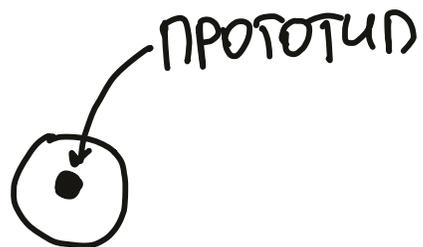
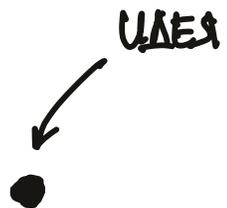
Преамбула



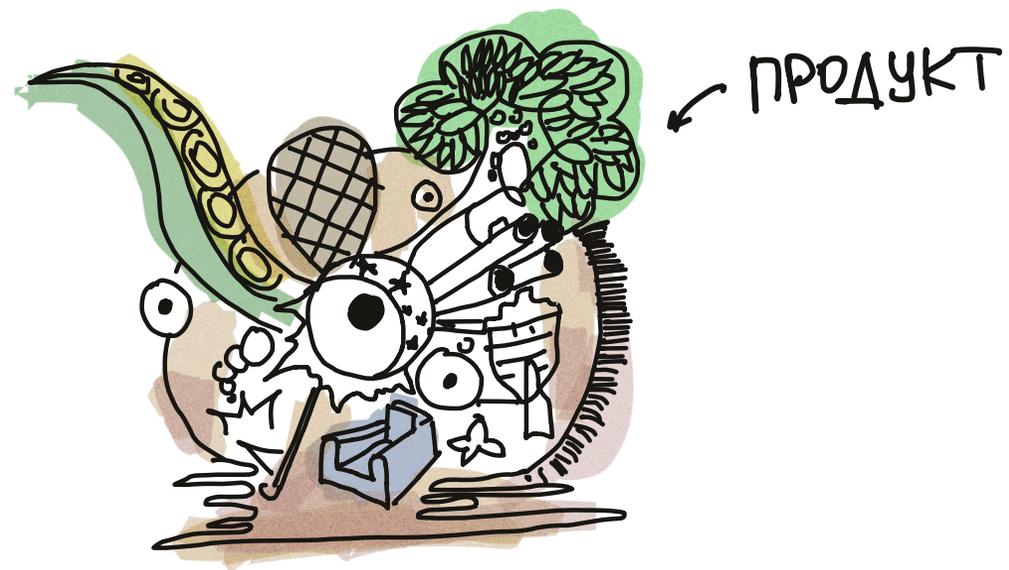
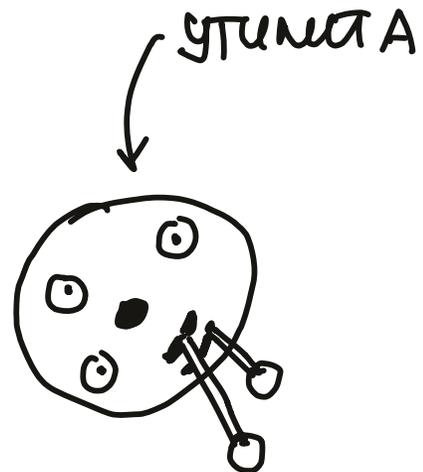
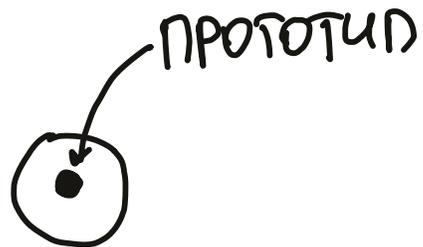
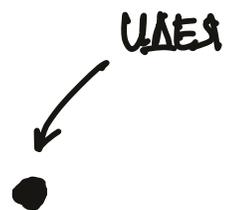
Преамбула



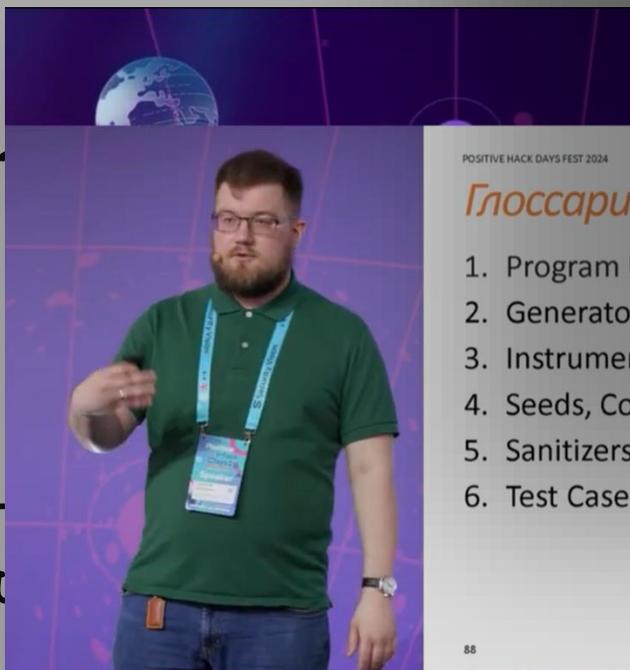
Преамбула



Преамбула



Преамбула



POSITIVE HACK DAYS FEST 2024

Глоссарий

1. Program
2. Generator
3. Instrument
4. Seeds, Co
5. Sanitizers, Exceptions, Oracles
6. Test Cases, Unit Test

88

```
80 int right = main.mergeSort(left + 4 * size - 1, (length - 1) / 2);
81 // perform merge sort
82 merge(arr, left, mid, right);
83 }
84 }
85 }
86 }
87 public static void main(String[] args) {
88     int[] arr = { 10, 3, 2, 19, 7, 15, 23, 13, 1 };
89     System.out.println(Arrays.toString(arr));
90     timSort(arr);
91     System.out.println(Arrays.toString(arr));
92 }
93 }
94 }
```

Максим Пелевин

Знакомство с фаззерами

Joker<?>

СБЕР

Яндекс

+T1

Сфера

РоссельхозБанк

X5 Tech

росбанк

hh

ПРОДУКТ

Фазза

SYZKALLER

Устройство фаззера

Фаззинг — это...

техника тестирования, при которой тестируемая программа запускается со сгенерированными случайными входными значениями. Работа программы впоследствии анализируется на наличие ошибок, нарушений проверок и потенциальных утечек памяти.

Однажды в 1988 году

**COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN-MADISON**

**CS 736
Fall 1988**

Bart Miller

Project List

(Brief Description Due: Wednesday, October 26)

(Midway Interview: Friday, November 18)

(Final Report Due: Thursday, December 15)

General Comments

The projects are intended to give you an opportunity to study a particular area related to operating systems. Your project may require a test implementation, measurement study, simulation, literature search,

Однажды в 1988 году

Projects

- (1) *Operating System Utility Program Reliability – The Fuzz Generator*: The goal of this project is to evaluate the robustness of various UNIX utility programs, given an unpredictable input stream. This project has two parts. First, you will build a *fuzz* generator. This is a program that will output a random character stream. Second, you will take the fuzz generator and use it to attack as many UNIX utilities as possible, with the goal of trying to break them. For the utilities that break, you will try to determine what type of input cause the break.

The fuzz generator will generate an output stream of random characters. It will need several options to give you flexibility to test different programs. Below is the start for a list of options for features that *fuzz* will support. It is important when writing this program to use good C and UNIX style, and good structure, as we hope to distribute this program to others.

- p only the printable ASCII characters
- a all ASCII characters
- 0 include the null (0 byte) character
- l generate random length lines (\n terminated strings)

Однажды в 1988 году

Название проекта: Operating System Utility Program Reliability - The Fuzz Generator.

Цель проекта: evaluate the robustness of various UNIX utility programs, given an unpredictable input stream.

Задачи проекта:

1. Build a *fuzz* generator.
2. Attack as many UNIX utilities as possible, with the goal of trying to break them.

Однажды в 1988 году

Название проекта: Operating System Utility Program Reliability - The Fuzz Generator.

Цель проекта: evaluate the robustness of various UNIX utility programs, given an unpredictable input stream.

Задачи проекта:

1. Build a *fuzz* generator.
2. Attack as many UNIX utilities as possible, with the goal of trying to break them.

Однажды в 1988 году

Название проекта: Operating System Utility Program Reliability - The Fuzz Generator.

Цель проекта: evaluate the robustness of various UNIX utility programs, given an unpredictable input stream.

Задачи проекта:

1. Build a *fuzz* generator.
2. Attack as many UNIX utilities as possible, with the goal of trying to break them.

Общая схема

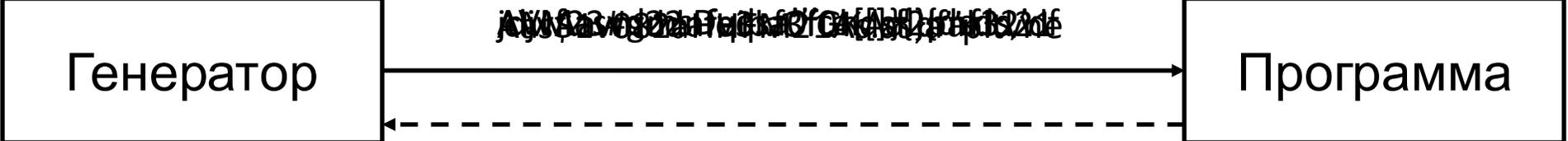
```
fuzz 100000 -o outfile | eqn
```

Общая схема

```
fuzz 100000 -o outfile | eqn
```

```
cat outfile | eqn
```

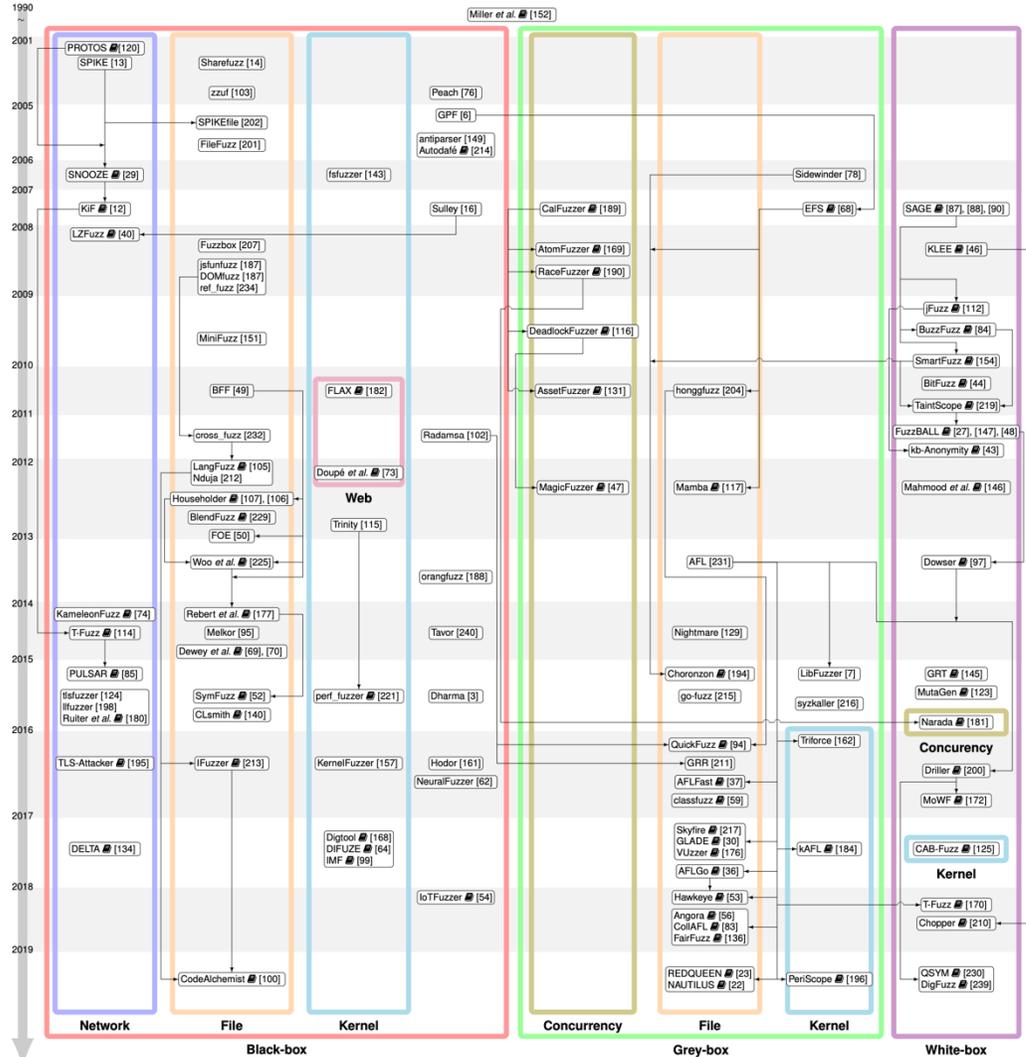
Общая схема



Общая схема

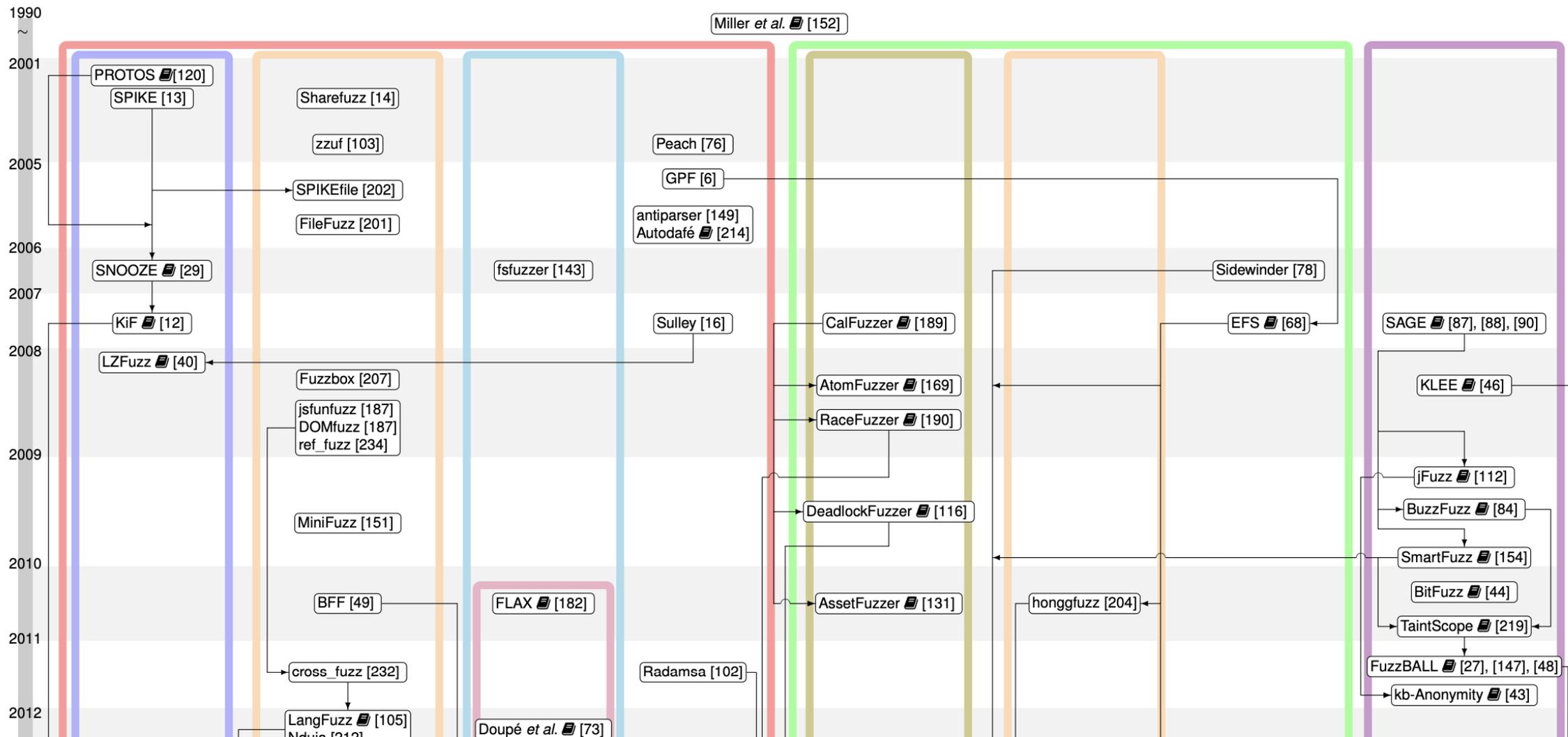


Эволюция фаззеров

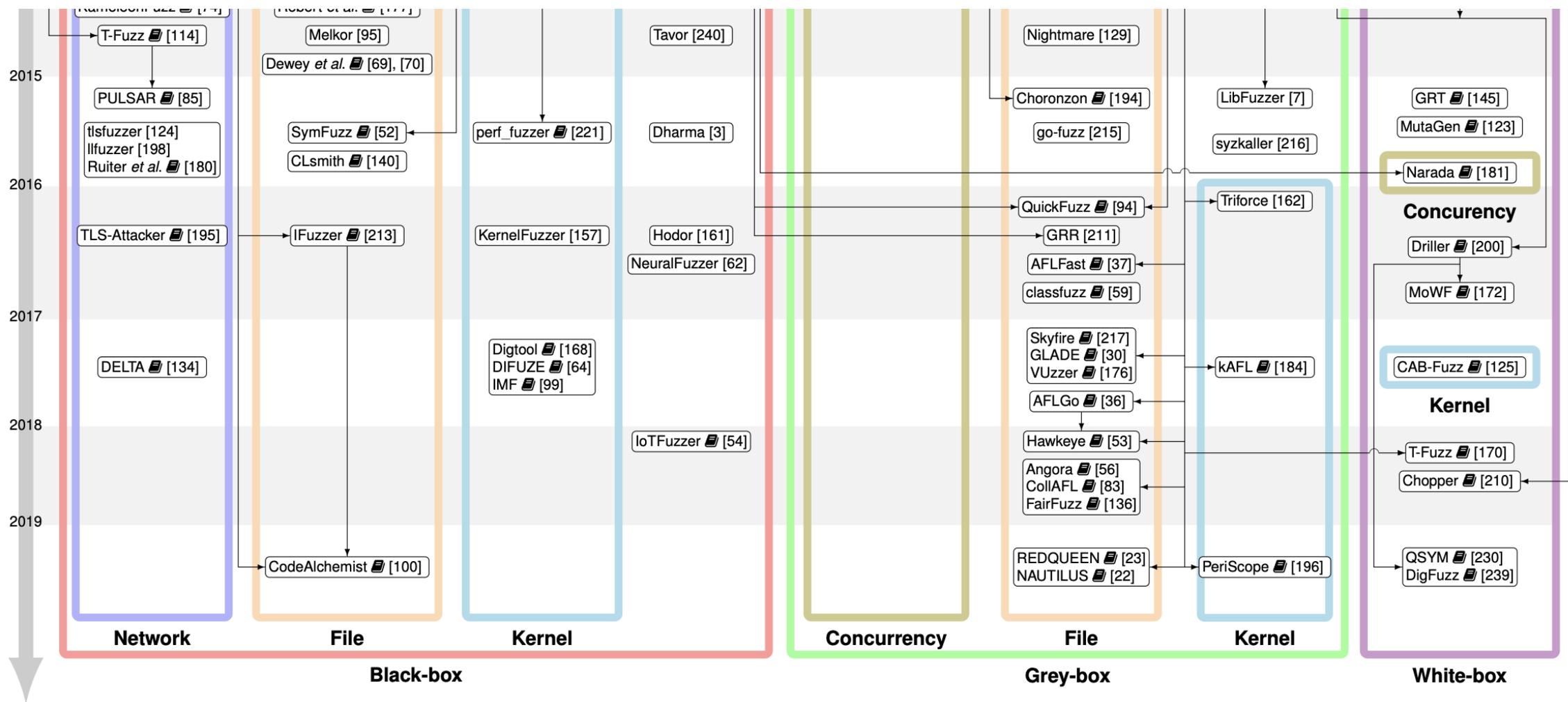


The Art, Science, and Engineering of Fuzzing: A Survey
 by Valentin J.M. Mañes, HyungSeok Han,
 Choongwoo Han, Sang Kil Cha, Manuel Egele,
 Edward J. Schwartz, and Maverick Woo

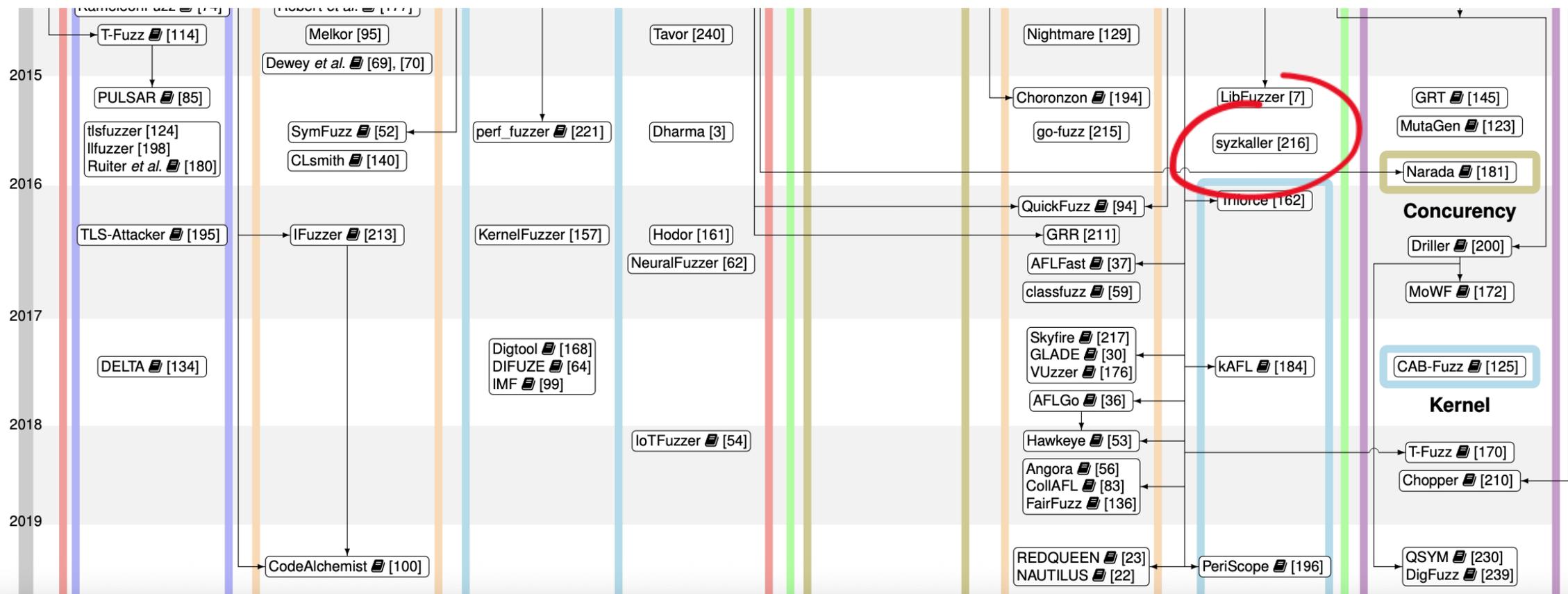
Эволюция фаззеров



Эволюция фаззеров



Эволюция фаззеров



syzkaller.appspot.com/upstream

syzbot

Linux

[sign-in](#) | [mailing list](#) | [source](#) | [docs](#)



Open [1664]

Subsystems



Fixed [6088]



Invalid [15125]



Missing Backports [159]



Crashes



Graphs



Coverage



Send us feedback

🐛 **Open [1664]**
≡ Subsystems
🐛 **Fixed [6088]**
🐛 **Invalid [15125]**
⬇️ **Missing Backports [159]**
≡ Crashes
📈 **Graphs**
📈 **Coverage**
💬 **Send us feedback**

Instances [tested repos]:

Name	Last active	Uptime	Corpus	Coverage <input type="checkbox"/>	Crashes	Execs	Kernel build				syzkaller build			Bu
							Commit	Config	Freshness	Status	Commit	Freshness	Status	
ci-qemu-gce-upstream-auto	now	4h00m	25027	363015	138	592865	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all
ci-qemu-native-arm64-kvm	now	19h29m	1268	20596		71135	29281a76709c	.config	23d		163f510d	20h04m		all
ci-qemu-upstream	now	4h38m	22563	359280	710	1193457	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all
ci-qemu-upstream-386	now	4h22m	45124	631076	388	1293644	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all
ci-qemu2-arm32	103d					broken	28eb75e178d3	.config	107d	failing	163f510d	20h04m		all
ci-qemu2-arm64	now	4h02m	79460	92545	31	401163	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all
ci-qemu2-arm64-compat	now	4h18m	48630	55397		232308	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all
ci-qemu2-arm64-mte	now	4h48m	177701	194779	83	1111980	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all
ci-qemu2-riscv64	now	19h33m	28977	401000	104	168131	245aece3750d	.config	22d		163f510d	20h04m		all
ci-snapshot-upstream-root	now	4h20m	66488	354218	1220	1529809	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all
ci-upstream-bpf-kasan-gce	now	19h31m	22063	208212	999	856374	6ccf6adb05d0	.config	5d19h		163f510d	20h04m		all
ci-upstream-bpf-next-kasan-gce	now	19h31m	21521	209087	1427	1424707	f28214603dc6	.config	1d08h		163f510d	20h04m		all
ci-upstream-gce-arm64	now	19h31m	76146	569661	1747	786654	77c95b8c7a16	.config	1d23h		163f510d	20h04m		all
ci-upstream-gce-leak	now	3h55m	52446	864335	970	2594057	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all
ci-upstream-kasan-badwrites-root	now	2h04m	46759	819314	351	639837	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all
ci-upstream-kasan-gce	now	3h36m	53664	409505	474	1674359	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all
ci-upstream-kasan-gce-386	now	2h49m	50835	400899	170	920429	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all
ci-upstream-kasan-gce-root	now	3h03m	78393	696331	1113	1103694	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all
ci-upstream-kasan-gce-selinux-root	now	3h17m	43608	703294	519	1576697	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all
ci-upstream-kasan-gce-smack-root	now	4h26m	74040	608144	720	1607694	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all
ci-upstream-kmsan-gce-386-root	now	2h23m	63399	478483	187	697550	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all
ci-upstream-kmsan-gce-root	now	3h57m	73676	552655	569	1170025	1110ce6a1e34	.config	17h29m		163f510d	20h04m		all

open (1401):

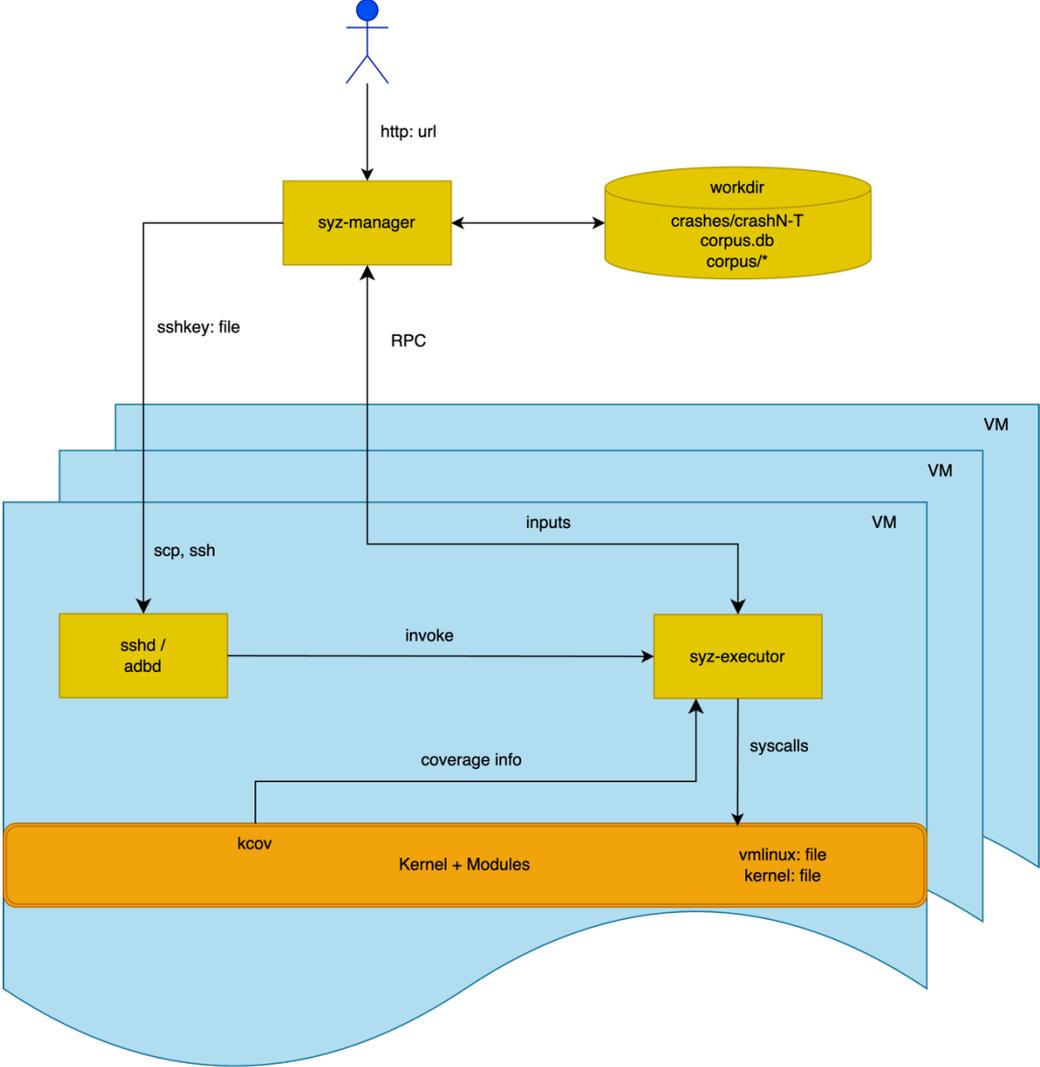
Title	Repro	Cause bisect	Fix bisect	Count	Last	Reported	Discussions
KASAN: slab-out-of-bounds							

ci-upstream-kasan-gce-smack-root	now	4h26m	74040	608144	720	1607694	1110ce6a1e34	.config	17h29m	163f510d	20h04m	all
ci-upstream-kmsan-gce-386-root	now	2h23m	63399	478483	187	697550	1110ce6a1e34	.config	17h29m	163f510d	20h04m	all
ci-upstream-kmsan-gce-root	now	3h57m	73676	552655	569	1170025	1110ce6a1e34	.config	17h29m	163f510d	20h04m	all

open (1401):

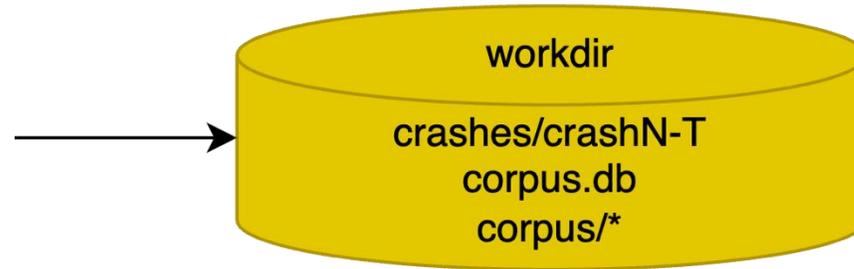
Title	Repro	Cause bisect	Fix bisect	Count	Last	Reported	Discussions
KCSAN: data-race in can_send / can_send (5) <code>can</code>				52	2d15h	7h18m	
WARNING: locking bug in alloc_pid <code>kernel</code>				1	4d07h	7h45m	
possible deadlock in lapbeth_device_event <code>x25</code>	<code>syz</code>	<code>error</code>		593	now	8h28m	PATCH [21m]
KMSAN: uninit-value in bch2_extent_crc_append (2) <code>bcachefs</code>	<code>C</code>			4	4d11h	13h44m	
KMSAN: uninit-value in __nf_conncount_add <code>netfilter</code>				1	4d15h	15h01m	PATCH [8h22m]
possible deadlock in efivarfs_actor <code>efi</code> <code>fs</code>	<code>C</code>			3	4d11h	15h12m	1 [8h14m]
WARNING in btrfs_create_new_inode (2) <code>btrfs</code>				1	4d22h	22h16m	
KCSAN: data-race in selinux_socket_post_create / selinux_socket_so...				1	2d06h	1d18h	PATCH [13h10m]
KASAN: slab-out-of-bounds Read in vc_uniscr_check <code>serial</code>				1	6d05h	1d18h	
WARNING: locking bug in d_set_mounted <code>bcachefs</code>				1	6d07h	1d18h	
KASAN: slab-use-after-free Read in nla_put (2) <code>rdma</code>				1	6d19h	1d18h	
WARNING in tracepoint_add_func (2) <code>trace</code>				2	6d03h	1d18h	1 [1d16h]
WARNING in dev_setup_tc <code>netfilter</code>				34	1d12h	1d18h	2 [1d00h]
KASAN: slab-use-after-free Read in ip_skb_dst_mtu (2) <code>net</code> <code>bcachefs</code>	<code>syz</code>	<code>done</code>		1	7d23h	1d18h	
possible deadlock in register_netdevice <code>net</code>	<code>syz</code>	<code>error</code>		1035	1d00h	1d18h	
UBSAN: shift-out-of-bounds in dbAllocAG (2) <code>jfs</code>	<code>C</code>	<code>done</code>		4	5d23h	1d18h	
BUG: unable to handle kernel paging request in pipe_write <code>bcachefs</code>	<code>syz</code>			1	2d20h	2d10h	1 [2d06h]
general protection fault in proc_sys_compare <code>bcachefs</code>	<code>syz</code>	<code>done</code>		1	6d15h	2d15h	6 [20h06m]
KASAN: slab-out-of-bounds Read in __bch2_bkey_unpack_key ...	<code>C</code>			11	3d19h	3d00h	1 [2d00h]
BUG: unable to handle kernel paging request in handle_softirqs <code>bpf</code> ...	<code>syz</code>			1	8d12h	4d12h	
WARNING in depot_fetch_stack <code>mm</code>				3	8d07h	4d12h	
KCSAN: data-race in virtqueue_disable_cb / virtqueue_enable_cb_de...				1	5d01h	4d20h	
KASAN: slab-out-of-bounds Read in xlog_cksum <code>xf</code>	<code>C</code>			4	5d01h	5d01h	3 [4d05h]
WARNING in taprio_get_start_time <code>net</code>				1	5d12h	5d08h	
INFO: trying to register non-static key in sock_def_readable <code>net</code> ...	<code>syz</code>			1	5d11h	5d08h	
BUG: corrupted list in __sk_destruct (2) <code>net</code> <code>ext4</code>	<code>syz</code>	<code>done</code>		1	5d22h	5d08h	
BUG: sleeping function called from invalid context in __bio_release ...				3	9h35m	5d10h	
KASAN: slab-out-of-bounds Read in validate_bset_keys <code>bcachefs</code>	<code>C</code>			5	6d19h	6d03h	1 [2d09h]
WARNING: bad unlock balance in __mm_populate <code>xf</code> <code>mm</code>	<code>C</code>			30	12m	7d00h	11 [14h12m]
KASAN: global-out-of-bounds Read in get_mem_cgroup_from_mm ...	<code>C</code>			1	7d19h	7d06h	
kernel BUG in try_to_unmap_one <code>mm</code>				1	11d	7d19h	3 [6d01h]

Архитектура Syzkaller



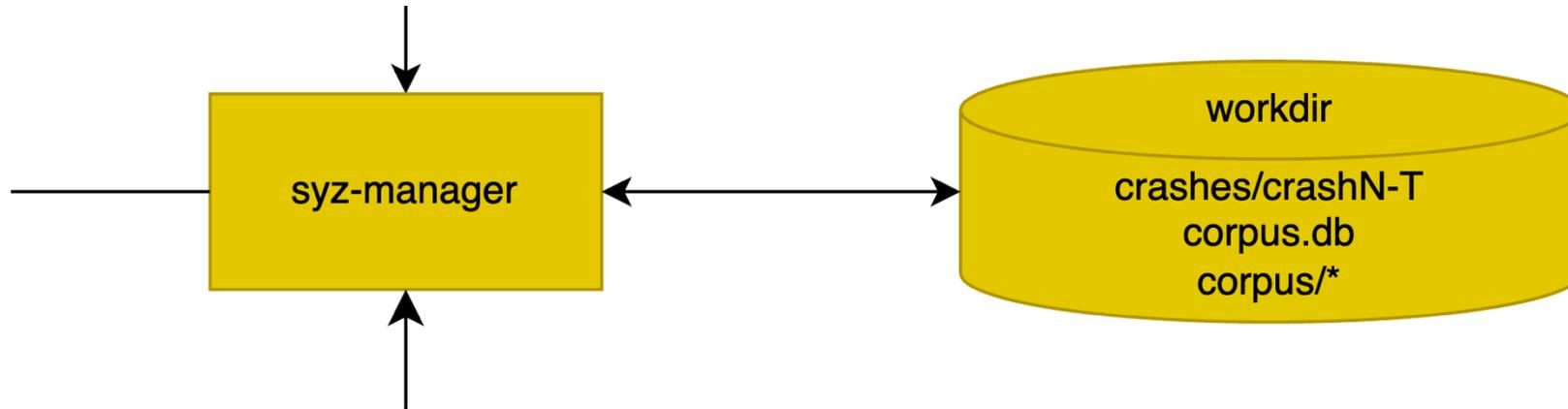
Архитектура Syzkaller

Хранилище значений

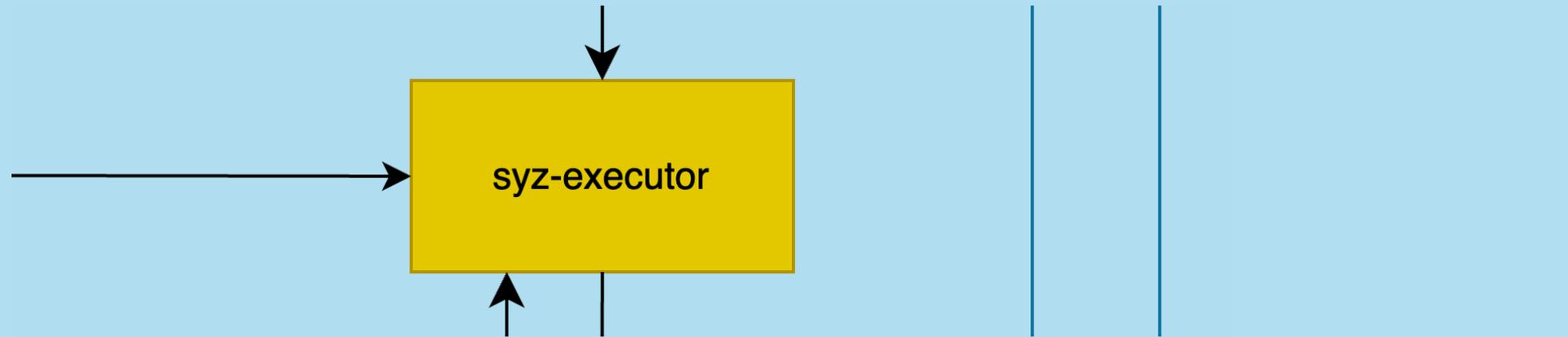


Архитектура Syzkaller

Генератор

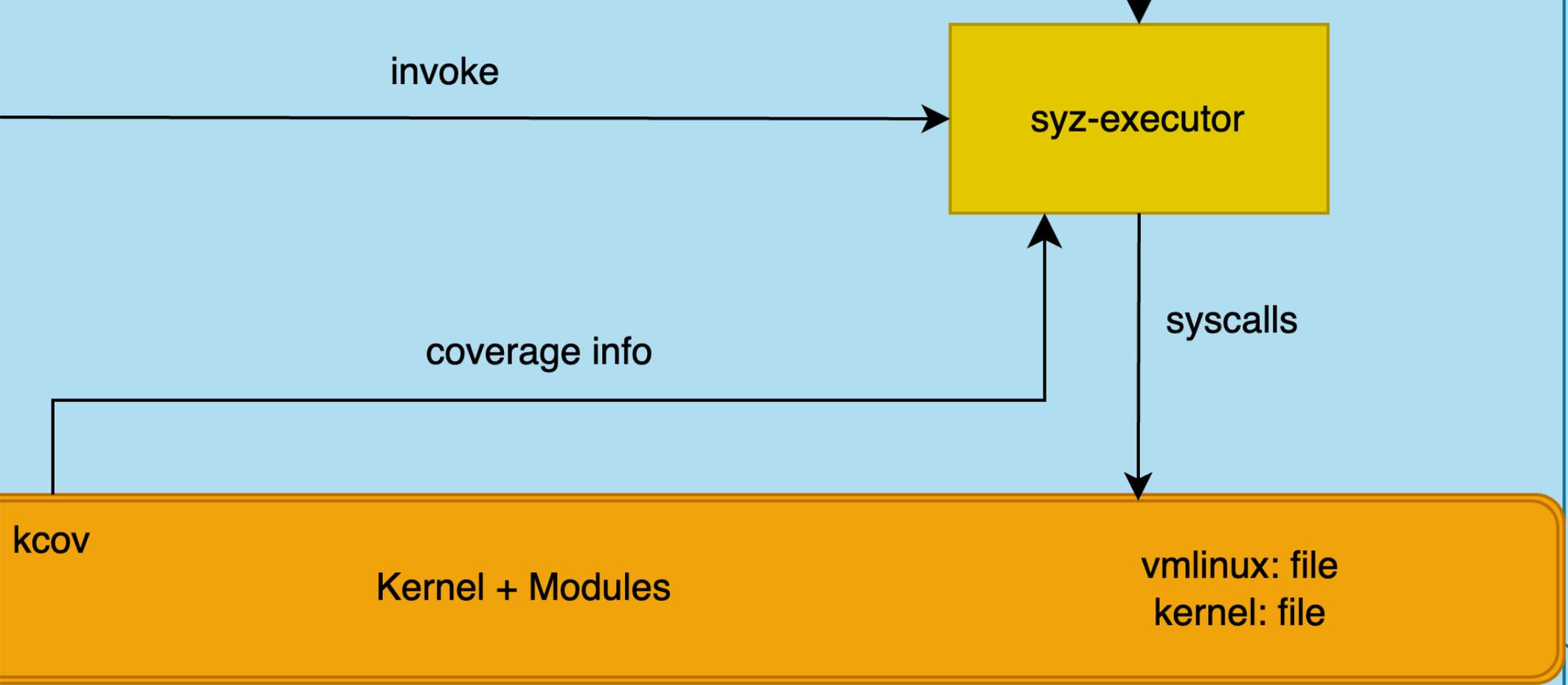


Архитектура Syzkaller

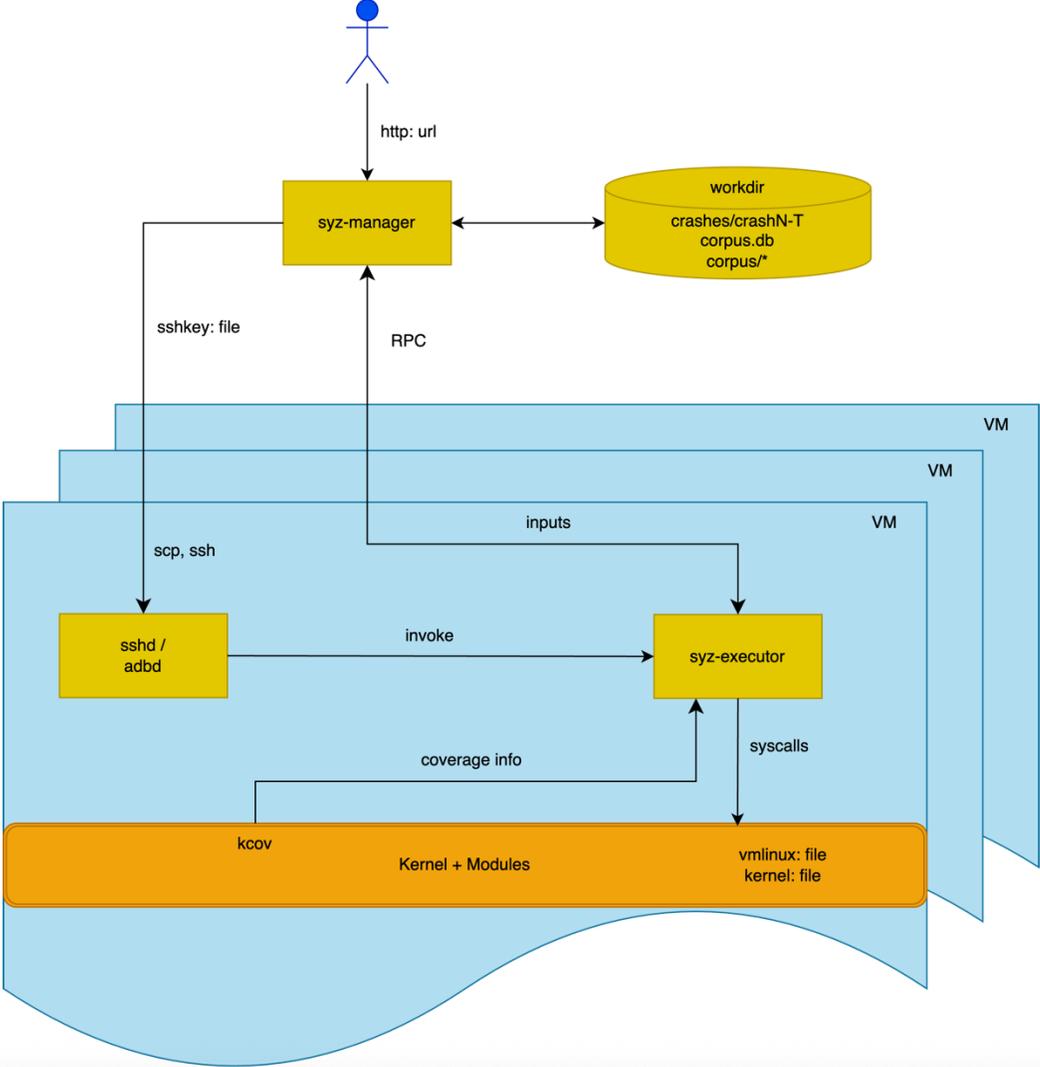


Архитектура Syzkaller

Инструментация



Архитектура Syzkaller



- corpus
- cover
- coveragedb
- covermerger
- csource
- db
- debugtracer
- delextract
- email
- flatrpc
- fuzzer
 - queue
 - cover.go
 - fuzzer.go
 - fuzzer_test.go
 - job.go
 - job_test.go
 - stats.go
- gce
- gcs
- hash
- html
- ifaceprobe
- ifuzz
- image
- instance
- kcidb

```
← → manager.go runner.go qemu.go execprog.go local.go executor.cc fuzzer.go snapshot.go dashapi.go + □ ↗
pkg/fuzzer/fuzzer.go
267 func (fuzzer *Fuzzer) genFuzz() *queue.Request {
268     // Either generate a new input or mutate an existing one.
269     mutateRate := 0.95
270     if !fuzzer.Config.Coverage {
271         // If we don't have real coverage signal, generate programs
272         // more frequently because fallback signal is weak.
273         mutateRate = 0.5
274     }
275     var req *queue.Request
276     rnd := fuzzer.rand()
277     if rnd.Float64() < mutateRate {
278         req = mutateProgRequest(fuzzer, rnd)
279     }
280     if req == nil {
281         req = genProgRequest(fuzzer, rnd)
282     }
283     if fuzzer.Config.Collide && rnd.Intn(3) == 0 {
284         req = &queue.Request{
285             Prog: randomCollide(req.Prog, rnd),
286             Stat: fuzzer.statExecCollide,
287         }
288     }
289     fuzzer.prepare(req, 0, 0)
290     return req
291 }
```

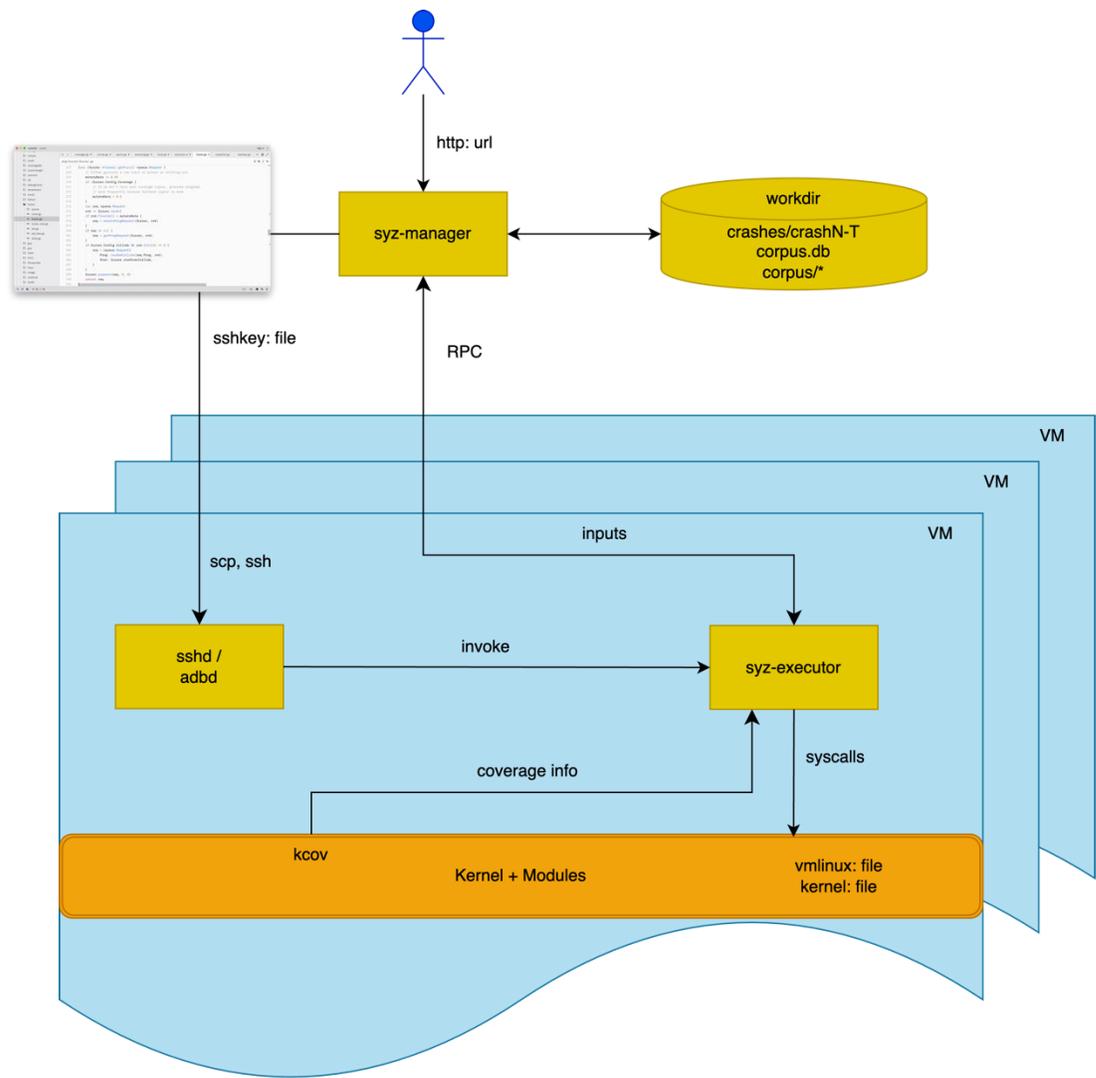
- corpus
- cover
- coveragedb
- covermerger
- csource
- db
- debugtracer
- declextract
- email
- flatrpc
- fuzzer
 - queue
 - cover.go
 - fuzzer.go
 - fuzzer_test.go
 - job.go
 - job_test.go
 - stats.go
- gce
- gcs
- hash
- html
- ifaceprobe
- ifuzz
- image
- instance
- kcidb

```
← → manager.go runner.go qemu.go execprog.go local.go executor.cc fuzzer.go snapshot.go dashapi.go + □ ↗
pkg/fuzzer/fuzzer.go
267 func (fuzzer *Fuzzer) genFuzz() *queue.Request {
268     // Either generate a new input or mutate an existing one.
269     mutateRate := 0.95
270     if !fuzzer.Config.Coverage {
271         // If we don't have real coverage signal, generate programs
272         // more frequently because fallback signal is weak.
273         mutateRate = 0.5
274     }
275     var req *queue.Request
276     rnd := fuzzer.rand()
277     if rnd.Float64() < mutateRate {
278         req = mutateProgRequest(fuzzer, rnd)
279     }
280     if req == nil {
281         req = genProgRequest(fuzzer, rnd)
282     }
283     if fuzzer.Config.Collide && rnd.Intn(3) == 0 {
284         req = &queue.Request{
285             Prog: randomCollide(req.Prog, rnd),
286             Stat: fuzzer.statExecCollide,
287         }
288     }
289     fuzzer.prepare(req, 0, 0)
290     return req
291 }
```

- corpus
- cover
- coveragedb
- covermerger
- csource
- db
- debugtracer
- declextract
- email
- flatrpc
- fuzzer
 - queue
 - cover.go
 - fuzzer.go
 - fuzzer_test.go
 - job.go
 - job_test.go
 - stats.go
- gce
- gcs
- hash
- html
- ifaceprobe
- ifuzz
- image
- instance
- kcidb

```
← → manager.go runner.go qemu.go execprog.go local.go executor.cc fuzzer.go job.go prio.go prog + [ ↗
pkg/fuzzer/job.go
44 func genProgRequest(fuzzer *Fuzzer, rnd *rand.Rand) *queue.Request {
45     p := fuzzer.target.Generate(rnd,
46         prog.RecommendedCalls,
47         fuzzer.ChoiceTable())
48     return &queue.Request{
49         Prog:    p,
50         ExecOpts: setFlags(flatrpc.ExecFlagCollectSignal),
51         Stat:    fuzzer.statExecGenerate,
52     }
53 }
54
55 func mutateProgRequest(fuzzer *Fuzzer, rnd *rand.Rand) *queue.Request {
56     p := fuzzer.Config.Corpus.ChooseProgram(rnd)
57     if p == nil {
58         return nil
59     }
60     newP := p.Clone()
61     newP.Mutate(rnd,
62         prog.RecommendedCalls,
63         fuzzer.ChoiceTable(),
64         fuzzer.Config.NoMutateCalls,
65         fuzzer.Config.Corpus.Programs(),
66     )
67     return &queue.Request{
68         Prog:    newP,
```

Архитектура Syzkaller



СИСТЕМНЫЕ ВЫЗОВЫ В Syzkaller

```
static intptr_t execute_syscall(const call_t* c, intptr_t a[kMaxArgs])
{
    if (c->call)
        return c->call(a[0], a[1], a[2], a[3], a[4], a[5], a[6], a[7], a[8]);
    return syscall(c->sys_nr, a[0], a[1], a[2], a[3], a[4], a[5]);
}
```

syscall(2)

System Calls Manual

syscall(2)

NAME [top](#)

syscall - indirect system call

LIBRARY [top](#)

Standard C library (*libc*, *-lc*)

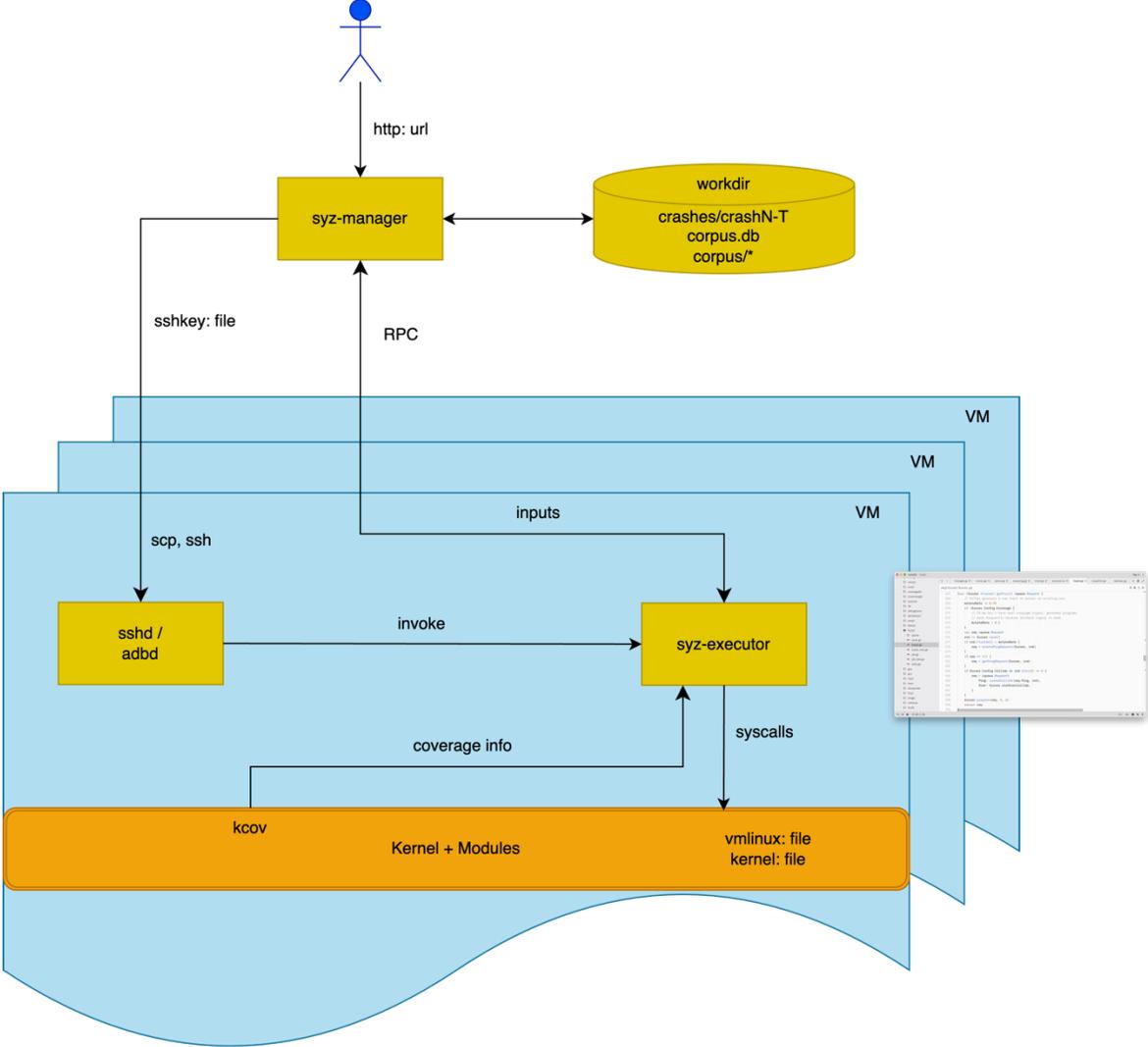
- common_kvm_arm64_syz
- common_kvm_ppc64.h
- common_linux.h
- common_openbsd.h
- common_test.h
- common_usb.h
- common_usb_linux.h
- common_usb_netbsd.h
- common_windows.h
- common_zlib.h
- conn.h
- cover_filter.h
- embed.go
- executor.cc
- executor_bsd.h
- executor_darwin.h
- executor_fuchsia.h
- executor_linux.h
- executor_runner.h
- executor_test.h
- executor_windows.h
- files.h
- gen_linux_amd64.go
- gen_linux_ppc64le.go
- kvm.h
- kvm_amd64.S
- kvm_amd64.S.h
- kvm_gen.cc

```
manager.go runner.go qemu.go execprog.go local.go executor.cc fuzzer.go job.go on_linux.h executor + [ ] ↗
executor/executor_linux.h
86         break;
87     }
88     pkeys[npkey] = pk;
89 }
90 while (npkey--)
91     pkey_free(pkeys[npkey]);
92 }
93
94 static intptr_t execute_syscall(const call_t* c, intptr_t a[kMaxArgs])
95 {
96     if (c->call)
97         return c->call(a[0], a[1], a[2], a[3], a[4], a[5], a[6], a[7], a[8]);
98     return syscall(c->sys_nr, a[0], a[1], a[2], a[3], a[4], a[5]);
99 }
100
101 static void cover_open(cover_t* cov, bool extra)
102 {
103     int fd = open("/sys/kernel/debug/kcov", O_RDWR);
104     if (fd == -1)
105         fail("open of /sys/kernel/debug/kcov failed");
106     if (dup2(fd, cov->fd) < 0)
107         failmsg("failed to dup cover fd", "from=%d, to=%d", fd, cov->fd);
108     close(fd);
109     const int kcov_init_trace = is_kernel_64_bit ? KCOV_INIT_TRACE64 : KCOV_INIT_TRACE32;
110     const int cover_size = extra ? kExtraCoverSize : flag_snapshot ? kSnapshotCoverSize
```

```
642 |         if (pid == 0) {
643 |             #if SYZ_EXECUTOR || SYZ_USE_TMP_DIR
644 |                 if (chdir(cwdbuf))
645 |                     fail("failed to chdir");
646 |             #endif
647 |             #if SYZ_HAVE_SETUP_TEST
648 |                 setup_test();
649 |             #endif
650 |             #if SYZ_HAVE_SETUP_EXT_TEST
651 |                 setup_ext_test();
652 |             #endif
653 |             #if SYZ_EXECUTOR
654 |                 close(kInPipeFd);
655 |             #endif
656 |             #if SYZ_EXECUTOR
657 |                 close(kOutPipeFd);
658 |             #endif
659 |                 execute_one();
660 |             #if !SYZ_EXECUTOR && SYZ_HAVE_CLOSE_FDS && !SYZ_THREADED
661 |                 // Executor's execute_one has already called close_fds.
662 |                 close_fds();
663 |             #endif
664 |                 doexit(0);
665 |         }
```

Что написано выше
по коду и для чего?

Архитектура Syzkaller



- testutil
- tool
- tools
- validator
- vcs
- vminfo
- prog
- sys
- syz-ci
- syz-hub
- syz-manager
 - hub.go
 - hub_test.go
 - manager.go
 - snapshot.go
 - stats.go
- syz-verifier
- tools
- vendor
- vm
- .clang-format
- .gitattributes
- .gitignore
- .golangci.yml
- AUTHORS

```
manager.go runner.go qemu.go execprog.go local.go executor.cc executor_linux.h  
syz-manager/manager.go  
61 type Manager struct {  
62     cfg          *mgrconfig.Config  
63     mode         *Mode  
64     vmPool       *vm.Pool  
65     pool         *vm.Dispatcher  
66     target       *prog.Target  
67     sysTarget    *targets.Target  
68     reporter     *report.Reporter  
69     crashStore   *manager.CrashStore  
70     serv         rpcserver.Server  
71     http         *manager.HTTPServer  
72     servStats    rpcserver.Stats  
73     corpus       *corpus.Corpus  
74     corpusDB     *db.DB  
75     corpusDBMu   sync.Mutex // for concurrent operations on corpusDB  
76     corpusPreload chan []fuzzer.Candidate  
77     firstConnect atomic.Int64 // unix time, or 0 if not connected  
78     crashTypes   map[string]bool  
79     enabledFeatures flatrpc.Feature  
80     checkDone    atomic.Bool  
81     reportGenerator *manager.ReportGeneratorWrapper  
82     fresh        bool  
83     coverFilters  manager.CoverageFilters
```

syz-manager/manager.go

```
61  type Manager struct {
62      cfg          *mgrconfig.Config
63      mode        *Mode
64      vmPool      *vm.Pool
65      pool        *vm.Dispatcher
66      target      *prog.Target
67      sysTarget   *targets.Target
68      reporter    *report.Reporter
69      crashStore  *manager.CrashStore
70      serv        rpcserver.Server
71      http        *manager.HTTPServer
72      servStats   rpcserver.Stats
73      corpus      *corpus.Corpus
74      corpusDB    *db.DB
75      corpusDBMu  sync.Mutex // for concurrent operations on corpusDB
76      corpusPreload chan []fuzzer.Candidate
77      firstConnect atomic.Int64 // unix time, or 0 if not connected
78      crashTypes  map[string]bool
79      enabledFeatures flatrpc.Feature
80      checkDone   atomic.Bool
81      reportGenerator *manager.ReportGeneratorWrapper
82      fresh       bool
83      coverFilters  manager.CoverageFilters
```

```
61 type Manager struct {
62     cfg          *mgrconfig.Config
63     mode        *Mode
64     vmPool      *vm.Pool
65     pool        *vm.Dispatcher
66     target      *prog.Target
67     sysTarget   *targets.Target
68     reporter    *report.Reporter
69     crashStore  *manager.CrashStore
70     serv        rpcserver.Server
71     http        *manager.HTTPServer
72     servStats   rpcserver.Stats
73     corpus      *corpus.Corpus
74     corpusDB    *db.DB
75     corpusDBMu  sync.Mutex // for concurrent operations on corpusDB
76     corpusPreload chan []fuzzer.Candidate
77     firstConnect atomic.Int64 // unix time, or 0 if not connected
78     crashTypes  map[string]bool
79     enabledFeatures flatrpc.Feature
80     checkDone   atomic.Bool
81     reportGenerator *manager.ReportGeneratorWrapper
82     fresh       bool
83     coverFilters manager.CoverageFilters
```

Конфигурация фаззера

Передаются в виде JSON во время запуска

```
61 type Manager struct {
62     cfg          *mgrconfig.Config
63     mode         *Mode
64     vmPool       *vm.Pool
65     pool         *vm.Dispatcher
66     target       *prog.Target
67     sysTarget    *targets.Target
68     reporter     *report.Reporter
69     crashStore   *manager.CrashStore
70     serv         rpcserver.Server
71     http         *manager.HTTPServer
72     servStats    rpcserver.Stats
73     corpus       *corpus.Corpus
74     corpusDB     *db.DB
75     corpusDBMu   sync.Mutex // for concurrent operations on corpusDB
76     corpusPreload chan []fuzzer.Candidate
77     firstConnect atomic.Int64 // unix time, or 0 if not connected
78     crashTypes   map[string]bool
79     enabledFeatures flatrpc.Feature
80     checkDone    atomic.Bool
81     reportGenerator *manager.ReportGeneratorWrapper
82     fresh        bool
83     coverFilters  manager.CoverageFilters
```

Режим работы

Например:

1. fuzzing
2. smoke-test
3. corpus-triage
4. corpus-run
5. run-tests

```

61  type Manager struct {
62      cfg          *mgrconfig.Config
63      mode        *Mode
64      vmPool       *vm.Pool
65      pool        *vm.Dispatcher
66      target      *prog.Target
67      sysTarget   *targets.Target
68      reporter    *report.Reporter
69      crashStore  *manager.CrashStore
70      serv        rpcserver.Server
71      http        *manager.HTTPServer
72      servStats   rpcserver.Stats
73      corpus      *corpus.Corpus
74      corpusDB    *db.DB
75      corpusDBMu  sync.Mutex // for concurrent operations on corpusDB
76      corpusPreload chan []fuzzer.Candidate
77      firstConnect atomic.Int64 // unix time, or 0 if not connected
78      crashTypes  map[string]bool
79      enabledFeatures flatrpc.Feature
80      checkDone   atomic.Bool
81      reportGenerator *manager.ReportGeneratorWrapper
82      fresh       bool
83      coverFilters  manager.CoverageFilters

```

Описание машины

Хранит информацию о том, какие машины (виртуальные или физические) используются, а также их количество и другую нужную для работы информацию

```
61 type Manager struct {
62     cfg          *mgrconfig.Config
63     mode        *Mode
64     vmPool      *vm.Pool
65     pool        *vm.Dispatcher
66     target      *prog.Target
67     sysTarget   *targets.Target
68     reporter    *report.Reporter
69     crashStore  *manager.CrashStore
70     serv        rpcserver.Server
71     http        *manager.HTTPServer
72     servStats   rpcserver.Stats
73     corpus      *corpus.Corpus
74     corpusDB    *db.DB
75     corpusDBMu  sync.Mutex // for concurrent operations on corpusDB
76     corpusPreload chan []fuzzer.Candidate
77     firstConnect atomic.Int64 // unix time, or 0 if not connected
78     crashTypes  map[string]bool
79     enabledFeatures flatrpc.Feature
80     checkDone   atomic.Bool
81     reportGenerator *manager.ReportGeneratorWrapper
82     fresh       bool
83     coverFilters manager.CoverageFilters
```

Пул тестовых машин

Хранит информацию обо всех созданных и используемых тестовых машинах (виртуальные или физические) на которых запускается тестовый набор системных вызовов

```
61 type Manager struct {
62     cfg          *mgrconfig.Config
63     mode        *Mode
64     vmPool      *vm.Pool
65     pool        *vm.Dispatcher
66     target      *prog.Target
67     sysTarget   *targets.Target
68     reporter    *report.Reporter
69     crashStore  *manager.CrashStore
70     serv        rpcserver.Server
71     http        *manager.HTTPServer
72     servStats   rpcserver.Stats
73     corpus      *corpus.Corpus
74     corpusDB    *db.DB
75     corpusDBMu  sync.Mutex // for concurrent operations on corpusDB
76     corpusPreload chan []fuzzer.Candidate
77     firstConnect atomic.Int64 // unix time, or 0 if not connected
78     crashTypes  map[string]bool
79     enabledFeatures flatrpc.Feature
80     checkDone   atomic.Bool
81     reportGenerator *manager.ReportGeneratorWrapper
82     fresh       bool
83     coverFilters manager.CoverageFilters
```

Описание

ОС/Архитектуры

Содержит информацию о паре ОС/Архитектура и дополнительную информацию, в том загруженный список системных вызовов, доступных для вызова.

```
61 type Manager struct {
62     cfg          *mgrconfig.Config
63     mode         *Mode
64     vmPool       *vm.Pool
65     pool         *vm.Dispatcher
66     target       *prog.Target
67     sysTarget    *targets.Target
68     reporter     *report.Reporter
69     crashStore   *manager.CrashStore
70     serv         rpcserver.Server
71     http         *manager.HTTPServer
72     servStats    rpcserver.Stats
73     corpus       *corpus.Corpus
74     corpusDB     *db.DB
75     corpusDBMu   sync.Mutex // for concurrent operations on corpusDB
76     corpusPreload chan []fuzzer.Candidate
77     firstConnect atomic.Int64 // unix time, or 0 if not connected
78     crashTypes   map[string]bool
79     enabledFeatures flatrpc.Feature
80     checkDone    atomic.Bool
81     reportGenerator *manager.ReportGeneratorWrapper
82     fresh        bool
83     coverFilters  manager.CoverageFilters
```

Отчёты о найденных
проблемах

```

61  type Manager struct {
62      cfg          *mgrconfig.Config
63      mode        *Mode
64      vmPool      *vm.Pool
65      pool        *vm.Dispatcher
66      target      *prog.Target
67      sysTarget   *targets.Target
68      reporter    *report.Reporter
69      crashStore  *manager.CrashStore
70      serv        rpcserver.Server
71      http        *manager.HTTPServer
72      servStats   rpcserver.Stats
73      corpus      *corpus.Corpus
74      corpusDB    *db.DB
75      corpusDBMu  sync.Mutex // for concurrent operations on corpusDB
76      corpusPreload chan []fuzzer.Candidate
77      firstConnect atomic.Int64 // unix time, or 0 if not connected
78      crashTypes  map[string]bool
79      enabledFeatures flatrpc.Feature
80      checkDone   atomic.Bool
81      reportGenerator *manager.ReportGeneratorWrapper
82      fresh       bool
83      coverFilters  manager.CoverageFilters

```

Канал для связи

Используется для связи и координации частей фаззера, запущенных на целевой машине.

```
61  type Manager struct {
62      cfg          *mgrconfig.Config
63      mode        *Mode
64      vmPool      *vm.Pool
65      pool        *vm.Dispatcher
66      target      *prog.Target
67      sysTarget   *targets.Target
68      reporter    *report.Reporter
69      crashStore  *manager.CrashStore
70      serv        rpcserver.Server
71      http        *manager.HTTPServer
72      servStats   rpcserver.Stats
73      corpus      *corpus.Corpus
74      corpusDB    *db.DB
75      corpusDBMu  sync.Mutex // for concurrent operations on corpusDB
76      corpusPreload chan []fuzzer.Candidate
77      firstConnect atomic.Int64 // unix time, or 0 if not connected
78      crashTypes  map[string]bool
79      enabledFeatures flatrpc.Feature
80      checkDone   atomic.Bool
81      reportGenerator *manager.ReportGeneratorWrapper
82      fresh       bool
83      coverFilters  manager.CoverageFilters
```

Для пользователя

Пользователь может взаимодействовать с фаззером через браузер, чтобы отслеживать результат работы

```
61  type Manager struct {
62      cfg          *mgrconfig.Config
63      mode        *Mode
64      vmPool      *vm.Pool
65      pool        *vm.Dispatcher
66      target      *prog.Target
67      sysTarget   *targets.Target
68      reporter    *report.Reporter
69      crashStore  *manager.CrashStore
70      serv        rpcserver.Server
71      http        *manager.HTTPServer
72      servStats   rpcserver.Stats
73      corpus      *corpus.Corpus
74      corpusDB    *db.DB
75      corpusDBMu  sync.Mutex // for concurrent operations on corpusDB
76      corpusPreload chan []fuzzer.Candidate
77      firstConnect atomic.Int64 // unix time, or 0 if not connected
78      crashTypes  map[string]bool
79      enabledFeatures flatrpc.Feature
80      checkDone   atomic.Bool
81      reportGenerator *manager.ReportGeneratorWrapper
82      fresh       bool
83      coverFilters  manager.CoverageFilters
```

Корпус программ

Хранит программы, которые сгенерировал фаззер и вместе покрывающих код ядра до текущих пределов

89

90

mu sync.Mutex

Логика фаззера

91

fuzzer atomic.Pointer[fuzzer.Fuzzer]

92

snapshotSource *queue.Distributor

93

phase int

94

95

disabledHashes map[string]struct{}

96

newRepros [][]byte

97

lastMinCorpus int

98

memoryLeakFrames map[string]bool

99

dataRaceFrames map[string]bool

100

saturatedCalls map[string]bool

101

102

externalReproQueue chan *manager.Crash

103

crashes chan *manager.Crash

104

105

benchMu sync.Mutex

106

benchFile *os.File

107

108

assetStorage *asset.Storage

109

110

reproLoop *manager.ReproLoop

111

- └─ windows
 - └─ gen
 - └─ empty.go
 - └─ init.go
 - └─ sys.txt
 - └─ sys_amd64.const
 - └─ windows.txt
 - └─ sys.go
- └─ syz-ci
- └─ syz-hub
- └─ syz-manager
 - └─ hub.go
 - └─ hub_test.go
 - └─ manager.go
 - └─ snapshot.go
 - └─ stats.go
- └─ syz-verifier
- └─ tools
 - └─ android
 - └─ jni
 - └─ Makefile
 - └─ arm64
 - └─ docker
 - └─ fops_probe
 - └─ kcovfuzzer
 - └─ kcovtrace
 - └─ syz-benchcmp

← → manager.go runner.go qemu.go execprog.go local.go executor.cc executor_linux.h ishapi.go References t + [] ↗

Q ✨ I ⚙

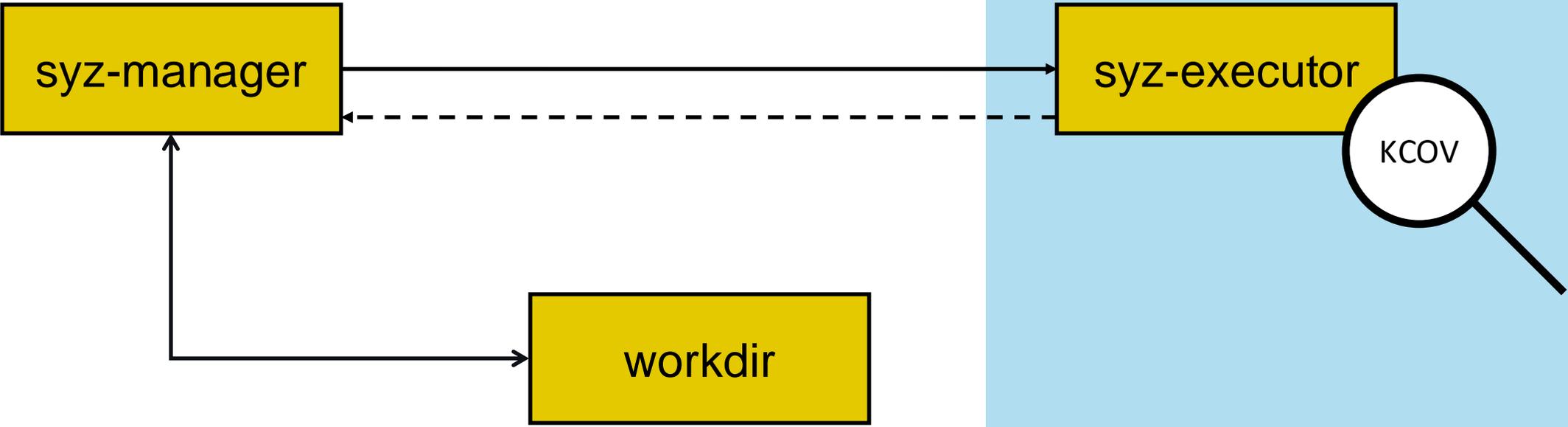
```

syz-manager/manager.go
89
90     mu          sync.Mutex
91     fuzzer      atomic.Pointer[fuzzer.Fuzzer]
92     snapshotSource *queue.Distributor
93     phase       int
94
95     disabledHashes map[string]struct{}
96     newRepros      [][]byte
97     lastMinCorpus  int
98     memoryLeakFrames map[string]bool
99     dataRaceFrames  map[string]bool
100    saturatedCalls  map[string]bool
101
102    externalReproQueue chan *manager.Crash
103    crashes            chan *manager.Crash
104
105    benchMu  sync.Mutex
106    benchFile *os.File
107
108    assetStorage *asset.Storage
109
110    reproLoop *manager.ReproLoop
111
112    Stats
113
  
```

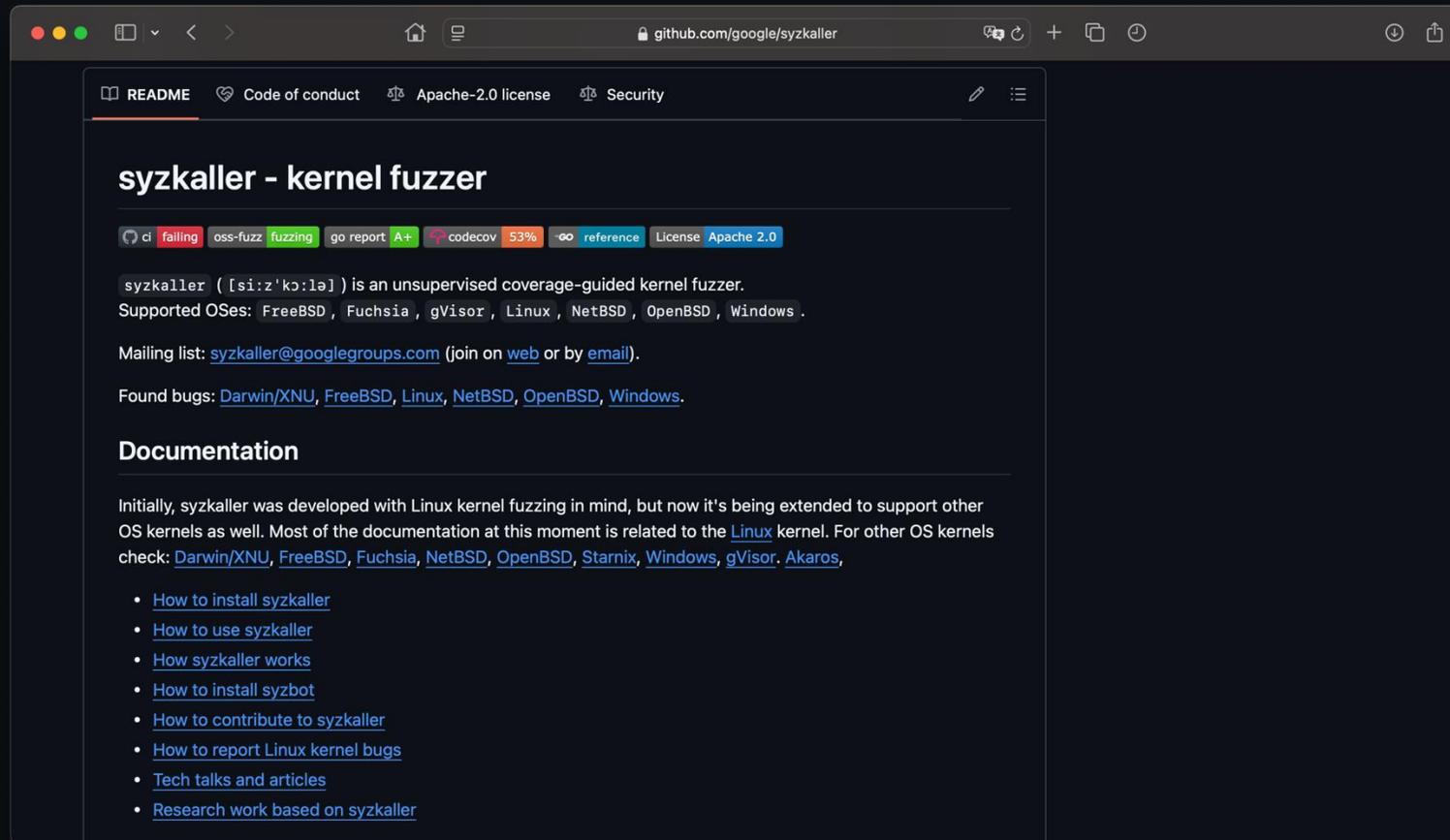
Общая схема



Схема Syzkaller



Установка и запуск



The screenshot shows the GitHub repository page for `syzkaller`. The browser address bar displays `github.com/google/syzkaller`. The repository navigation bar includes links for `README`, `Code of conduct`, `Apache-2.0 license`, and `Security`. The main content area features the title `syzkaller - kernel fuzzer` and a status bar with various metrics: `ci` (falling), `oss-fuzz` (fuzzing), `go report` (A+), `codecov` (53%), `reference`, and `License` (Apache 2.0).

`syzkaller` ([si:z'kɔ:lɚ]) is an unsupervised coverage-guided kernel fuzzer.
Supported OSes: FreeBSD, Fuchsia, gVisor, Linux, NetBSD, OpenBSD, Windows.

Mailing list: syzkaller@googlegroups.com (join on [web](#) or by [email](#)).

Found bugs: [Darwin/XNU](#), [FreeBSD](#), [Linux](#), [NetBSD](#), [OpenBSD](#), [Windows](#).

Documentation

Initially, syzkaller was developed with Linux kernel fuzzing in mind, but now it's being extended to support other OS kernels as well. Most of the documentation at this moment is related to the [Linux](#) kernel. For other OS kernels check: [Darwin/XNU](#), [FreeBSD](#), [Fuchsia](#), [NetBSD](#), [OpenBSD](#), [Starmix](#), [Windows](#), [gVisor](#), [Akaros](#).

- [How to install syzkaller](#)
- [How to use syzkaller](#)
- [How syzkaller works](#)
- [How to install syzbot](#)
- [How to contribute to syzkaller](#)
- [How to report Linux kernel bugs](#)
- [Tech talks and articles](#)
- [Research work based on syzkaller](#)

Установка и запуск

Documentation

Initially, syzkaller was developed with Linux kernel fuzzing in mind, but now it's being extended to support other OS kernels as well. Most of the documentation at this moment is related to the [Linux](#) kernel. For other OS kernels check: [Darwin/XNU](#), [FreeBSD](#), [Fuchsia](#), [NetBSD](#), [OpenBSD](#), [Starnix](#), [Windows](#), [gVisor](#). [Akaros](#),

- [How to install syzkaller](#)
- [How to use syzkaller](#)

Установка и запуск

Что нужно взять:

1. Машина, на которой всё будет работать
2. Ядро, которое нужно собрать
3. Образ, с которым придётся работать

Установка и запуск



> WSL

Windows 11 + WSL

QEMU

Debian Bullseye

Linux Kernel v.6.2



Kernel

Checkout Linux Kernel source

Command:

```
git clone --branch v6.2 git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git $KERNEL
```

```
# Coverage collection.
CONFIG_KCOV=y

# Debug info for symbolization.
CONFIG_DEBUG_INFO_DWARF4=y

# Memory bug detector
CONFIG_KASAN=y
CONFIG_KASAN_INLINE=y

# Required for Debian Stretch and later
CONFIG_CONFIGFS_FS=y
CONFIG_SECURITYFS=y
```

```
CONFIG_CMDLINE_BOOL=y
CONFIG_CMDLINE="net.ifnames=0"
```

Image

Preview Code Blame 286 lines (211 loc) · 7.02 KB

Raw Copy Download Edit

Kernel

Checkout Linux Kernel source

Command:

```
git clone --branch v6.2 git://git.k

# Coverage collection.
CONFIG_KCOV=y

# Debug info for symbolization.
CONFIG_DEBUG_INFO_DWARF4=y

# Memory bug detector
CONFIG_KASAN=y
CONFIG_KASAN_INLINE=y

# Required for Debian Stretch and la
CONFIG_CONFIGFS_FS=y
CONFIG_SECURITYFS=y

CONFIG_CMDLINE_BOOL=y
CONFIG_CMDLINE="net.ifnames=0"
```

Image

Development tools

- [Kernel Testing Guide](#)
- [Checkpatch](#)
- [clang-format](#)
- [Coccinelle](#)
- [Sparse](#)
- [KCOV: code coverage for fuzzing](#)**
 - [Prerequisites](#)
 - [Coverage collection](#)
 - [Comparison operands collection](#)
 - [Remote coverage collection](#)
- [Using gcov with the Linux kernel](#)
- [Kernel Address Sanitizer \(KASAN\)](#)
- [Kernel Memory Sanitizer \(KMSAN\)](#)
- [Undefined Behavior Sanitizer - UBSAN](#)
- [Kernel Memory Leak Detector](#)
- [Kernel Concurrency Sanitizer \(KCSAN\)](#)
- [Kernel Electric-Fence \(KFENCE\)](#)
- [Linux Kernel Selftests](#)
- [KUnit - Linux Kernel Unit Testing](#)
- [The Kernel Test Anything Protocol \(KTAP\), version 1](#)
- [UAPI Checker](#)
- [Linux Kernel GPIO based sloppy logic analyzer](#)
- [Using AutoFDO with the Linux kernel](#)
- [Using Propeller with the Linux kernel](#)
- [Testing guide](#)

KCOV: code coverage for fuzzing

English

KCOV collects and exposes kernel code coverage information in a form suitable for coverage-guided fuzzing. Coverage data of a running kernel is exported via the `kcov debugfs` file. Coverage collection is enabled on a task basis, and thus KCOV can capture precise coverage of a single system call.

Note that KCOV does not aim to collect as much coverage as possible. It aims to collect more or less stable coverage that is a function of syscall inputs. To achieve this goal, it does not collect coverage in soft/hard interrupts (unless remove coverage collection is enabled, see below) and from some inherently non-deterministic parts of the kernel (e.g. scheduler, locking).

Besides collecting code coverage, KCOV can also collect comparison operands. See the “Comparison operands collection” section for details.

Besides collecting coverage data from syscall handlers, KCOV can also collect coverage for annotated parts of the kernel executing in background kernel tasks or soft interrupts. See the “Remote coverage collection” section for details.

Prerequisites

KCOV relies on compiler instrumentation and requires GCC 6.1.0 or later or any Clang version supported by the kernel.

Collecting comparison operands is supported with GCC 8+ or with Clang.

To enable KCOV, configure the kernel with:

```
CONFIG_KCOV=y
```

To enable comparison operands collection, set:

```
CONFIG_KCOV_ENABLE_COMPARISONS=y
```

Coverage data only becomes accessible once debugfs has been mounted:



Image

Install debootstrap

Command:

```
sudo apt install debootstrap
```



Create Debian Bullseye Linux image

Create a Debian Bullseye Linux image with the minimal set of required packages.

Command:

```
mkdir $IMAGE
cd $IMAGE/
wget https://raw.githubusercontent.com/google/syzkaller/master/tools/create-image.sh -O create-image.sh
chmod +x create-image.sh
./create-image.sh
```



The result should be `$IMAGE/bullseye.img` disk image.

OR Create Debian Linux image with a different version

To create a Debian image with a different version (e.g. buster, stretch, sid), specify the `--distribution` option.

QEMU

Install QEMU

Command:

```
sudo apt install qemu-system-x86_64
```



OR Create Debian Linux image with a different version

To create a Debian image with a different version (e.g. buster, stretch, sid), specify the `--distribution` option.

QEMU

Install QEMU

Command:

```
sudo apt install qemu-system-x86
```



Verify

Make sure the kernel boots and `sshd` starts.

Command:

```
qemu-system-x86_64 \  
  -m 2G \  
  -smp 2 \  
  -kernel $KERNEL/arch/x86/boot/bzImage \  
  -append "console=ttyS0 root=/dev/sda earlyprintk=serial net.ifnames=0" \  
  -drive file=$IMAGE/bullseye.img,format=raw \  
  -net user,host=10.0.2.10,hostfwd=tcp:127.0.0.1:10021-:22 \  
  -net nic,model=e1000 \  
  -enable-kvm \  
  -nographic \  
  -pidfile vm.pid \  
  2>&1 | tee vm.log
```



syzkaller

Build syzkaller as described [here](#). Then create a manager config like the following, replacing the environment variables `$GOPATH`, `$KERNEL` and `$IMAGE` with their actual values.

```
-append "console=ttyS0 root=/dev/sda earlyprintk=serial net.ifnames=0" \  
-drive file=$IMAGE/bullseye.img,format=raw \  
-net user,host=10.0.2.10,hostfwd=tcp:127.0.0.1:10021-:22 \  
-net nic,model=e1000 \  
-enable-kvm \  
-nographic \  
-pidfile vm.pid \  
2>&1 | tee vm.log
```

syzkaller

Build syzkaller as described [here](#). Then create a manager config like the following, replacing the environment variables `$GOPATH`, `$KERNEL` and `$IMAGE` with their actual values.

```
{  
  "target": "linux/amd64",  
  "http": "127.0.0.1:56741",  
  "workdir": "$GOPATH/src/github.com/google/syzkaller/workdir",  
  "kernel_obj": "$KERNEL",  
  "image": "$IMAGE/bullseye.img",  
  "sshkey": "$IMAGE/bullseye.id_rsa",  
  "syzkaller": "$GOPATH/src/github.com/google/syzkaller",  
  "procs": 8,  
  "type": "qemu",  
  "vm": {  
    "count": 4,  
    "kernel": "$KERNEL/arch/x86/boot/bzImage",  
    "cpu": 2,  
    "mem": 2048  
  }  
}
```

Run syzkaller manager:

```
mkdir workdir  
./bin/syz-manager -config=my.cfg
```

Now syzkaller should be running, you can check manager status with your web browser at `127.0.0.1:56741`.

Установка и запуск



Fedora 41
QEMU
Debian Bullseye
Linux Kernel v.6.2

```
Warning: Permanently added '[localhost]:3730' (ED25519) to the list of known hosts.
/syz-executor: /lib/x86_64-linux-gnu/libstdc++.so.6: version 'GLIBCXX_3.4.29' not found (required by /syz-executor)
/syz-executor: /lib/x86_64-linux-gnu/libc.so.6: version 'GLIBC_2.38' not found (required by /syz-executor)
/syz-executor: /lib/x86_64-linux-gnu/libc.so.6: version 'GLIBC_2.33' not found (required by /syz-executor)
/syz-executor: /lib/x86_64-linux-gnu/libc.so.6: version 'GLIBC_2.34' not found (required by /syz-executor)

VM DIAGNOSIS:
20:55:05 Registers:
info registers vcpu 0

CPU#0
RAX=fffffffacab58c0 RBX=fffffffad81c440 RCX=fffffffaca9a8f1 RDX=0000000000000001
RSI=fffffffacf8e760 RDI=000000000020b324 RBP=0000000000000000 RSP=fffffffad807e30
R8 =0000000000000001 R9 =ffff88806d233ec3 R10=ffffed100da467d8 R11=0000000000000006
R12=0000000000000000 R13=0000000000000000 R14=0000000000000000 R15=dffffc0000000000
RIP=fffffffacab58cf RFL=00000202 [-----] CPL=0 II=0 A20=1 SMM=0 HLT=1
ES =0000 0000000000000000 ffffffff 00c00000
CS =0010 0000000000000000 ffffffff 00a09b00 DPL=0 CS64 [-RA]
SS =0018 0000000000000000 ffffffff 00c09300 DPL=0 DS [-WA]
DS =0000 0000000000000000 ffffffff 00c00000
ES =0000 0000000000000000 ffffffff 00c00000
```

Установка и запуск



Fedora 41

QEMU

Debian Sid

Linux Kernel v.6.2



Результаты

```
Windows PowerShell x markoutte@Orion: ~/Develop x + v x
write$trusty_storage : fd_trusty_storage [openat$trusty_storage]
write$tun : fd_tun [openat$tun]
write$uinput_user_dev : fd_uinput [openat$uinput]
write$usbip_server : fd_usbip_server [syz_usbip_server_init]
write$vhost_msg : vhost_net [openat$vnet]
write$vhost_msg_v2 : vhost_net [openat$vnet]
write$yama_ptrace_scope : fd_yama_ptrace_scope [openat$yama_ptrace_scope]
]

BinFmtMisc : enabled
Comparisons : call 0 failed with errno 998
Coverage : enabled
DelayKcovMmap : enabled
DevlinkPCI : PCI device 0000:00:10.0 is not available
ExtraCoverage : enabled
Fault : CONFIG_FAULT_INJECTION is not enabled
KCSAN : write(/sys/kernel/debug/kcsan, on) failed
LRWPANEmulation : netlink_query_family_id failed
Leak : failed to write(kmemleak, "scan=off")
NetDevices : enabled
NetInjection : tun: can't open /dev/net/tun. (errno 2: No such file or directory
).
NicVF : PCI device 0000:00:11.0 is not available
SandboxAndroid : setfilecon: setxattr failed. (errno 22: Invalid argument). . proc
ess exited with status 67.
SandboxNamespace : sandbox fork failed. (errno 22: Invalid argument). . process exit
ed with status 67.
SandboxNone : enabled
SandboxSetuid : enabled
Swap : enabled
USBEmulation : failed to chmod /dev/raw-gadget
VhciInjection : socket(AF_BLUETOOTH, SOCK_RAW, BTPROTO_HCI) failed. (errno 97: Ad
dress family not supported by protocol).
WifiEmulation : netlink_query_family_id failed. (errno 2: No such file or directo
ry).
syscalls : 2245/7791

2025/03/06 21:33:37 corpus : 9501 (215 seeds), 35 to be reminimized, 25 to
be resmashed
2025/03/06 21:33:43 candidates=9501 corpus=0 coverage=0 exec total=656 (21/sec) pending=0 re
producing=0
2025/03/06 21:33:53 candidates=9501 corpus=0 coverage=0 exec total=656 (16/sec) pending=0 re
producing=0
2025/03/06 21:34:03 candidates=9079 corpus=192 coverage=15168 exec total=2094 (41/sec) pendi
ng=0 reproducing=0
2025/03/06 21:34:16 candidates=8294 corpus=1121 coverage=31420 exec total=5646 (92/sec) pend
ing=0 reproducing=0
2025/03/06 21:34:26 candidates=7714 corpus=1473 coverage=35040 exec total=7547 (106/sec) pen
ding=0 reproducing=0
2025/03/06 21:34:36 candidates=6922 corpus=2295 coverage=49111 exec total=10993 (135/sec) pe
nding=0 reproducing=0
```

```

write$trusty_storage : fd_trusty_storage [openat$trusty_storage]
write$tun : fd_tun [openat$tun]
write$uinput : fd_uinput [openat$uinput]
write$usbip_server : fd_usbip_server [syz_usbip_server_init]
write$vhost_msg : vhost_net [openat$vnet]
write$vhost_msg_v2 : vhost_net [openat$vnet]
write$yama_ptrace_scope : fd_yama_ptrace_scope [openat$yama_ptrace_scope]
]

BinFmtMisc : enabled
Comparisons : call 0 failed with errno 998
Coverage : enabled
DelayKcovMmap : enabled
DevlinkPCI : PCI device 0000:00:10.0 is not available
ExtraCoverage : enabled
Fault : CONFIG_FAULT_INJECTION is not enabled
KCSAN : write(/sys/kernel/debug/kcsan, on) failed
LRWPANEmulation : netlink_query_family_id failed
Leak : failed to write(kmemleak, "scan=off")
NetDevices : enabled
NetInjection : tun: can't open /dev/net/tun. (errno 2: No such file or directory)
).
NicVF : PCI device 0000:00:11.0 is not available
SandboxAndroid : setfilecon: setattr failed. (errno 22: Invalid argument). . proc
ess exited with status 67.
SandboxNamespace : sandbox fork failed. (errno 22: Invalid argument). . process exit
ed with status 67.
SandboxNone : enabled
SandboxSetuid : enabled
Swap : enabled
USBEmulation : failed to chmod /dev/raw-gadget
VhciInjection : socket(AF_BLUETOOTH, SOCK_RAW, BTPROTO_HCI) failed. (errno 97: Ad
dress family not supported by protocol).
WifiEmulation : netlink_query_family_id failed. (errno 2: No such file or directo
ry).
syscalls : 2245/7791

2025/03/06 21:33:37 corpus : 9501 (215 seeds), 35 to be reminimized, 25 to
be resmashed
2025/03/06 21:33:43 candidates=9501 corpus=0 coverage=0 exec total=656 (21/sec) pending=0 re
producing=0
2025/03/06 21:33:53 candidates=9501 corpus=0 coverage=0 exec total=656 (16/sec) pending=0 re
producing=0
2025/03/06 21:34:03 candidates=9079 corpus=192 coverage=15168 exec total=2094 (41/sec) pendi
ng=0 reproducing=0
2025/03/06 21:34:16 candidates=8294 corpus=1121 coverage=31420 exec total=5646 (92/sec) pend
ing=0 reproducing=0
2025/03/06 21:34:26 candidates=7714 corpus=1473 coverage=35040 exec total=7547 (106/sec) pen
ding=0 reproducing=0
2025/03/06 21:34:36 candidates=6922 corpus=2295 coverage=49111 exec total=10993 (135/sec) pe
nding=0 reproducing=0

```

127.0.0.1:56741

syzkaller

stats
coverage
syscalls
corpus
VMs
config

candidates	1
corpus	8075
coverage	81727
exec total	42617 (129/sec)
pending	4
reproducing	0
crash types	1
crashes	4
suppressed	0
syscalls	2245
uptime	332 sec

Crashes:

Description	Count	First Time	Last Time	Report
can't ssh into the instance	2	2025/02/16 21:19	2025/02/16 21:19	
failed to read from qemu: EOF	16	2025/02/16 16:50	2025/02/16 16:50	
KASAN: stack-out-of-bounds Read in profile_pc	3	2025/02/16 21:15	2025/02/16 21:15	has C repro
no output from test machine	2	2025/02/16 22:17	2025/02/16 22:17	
SYZFAIL: mmap of data segment failed	95	2025/03/06 21:35	2025/03/06 21:35	
WARNING in blk_rq_map_user_iov	4	2025/02/16 19:45	2025/02/16 19:45	has C repro
WARNING: zero-size vmalloc in corrupted	2	2025/02/16 17:53	2025/02/16 17:53	
WARNING: zero-size vmalloc in sel_write_load	7	2025/02/16 19:46	2025/02/16 19:46	has C repro

Log:

```

2025/03/06 21:35:54 candidates=5942 corpus=3158 coverage=55121 exec total=14917 (98/sec) pending=4 reproducing=0
2025/03/06 21:36:04 candidates=5942 corpus=3158 coverage=55121 exec total=14917 (92/sec) pending=4 reproducing=0
2025/03/06 21:36:14 candidates=5546 corpus=3398 coverage=56569 exec total=16240 (94/sec) pending=4 reproducing=0
2025/03/06 21:36:27 candidates=4675 corpus=4533 coverage=62973 exec total=20439 (112/sec) pending=4 reproducing=0
2025/03/06 21:36:37 candidates=3832 corpus=5364 coverage=65724 exec total=24007 (125/sec) pending=4 reproducing=0
2025/03/06 21:36:47 candidates=3088 corpus=6113 coverage=71600 exec total=27205 (135/sec) pending=4 reproducing=0
2025/03/06 21:36:57 candidates=2414 corpus=6766 coverage=74851 exec total=30112 (142/sec) pending=4 reproducing=0
2025/03/06 21:37:10 candidates=1785 corpus=7391 coverage=77797 exec total=32909 (148/sec) pending=4 reproducing=0
2025/03/06 21:37:20 candidates=1385 corpus=7792 coverage=80210 exec total=34911 (151/sec) pending=4 reproducing=0
2025/03/06 21:37:30 candidates=1269 corpus=7895 coverage=81027 exec total=35787 (148/sec) pending=4 reproducing=0
2025/03/06 21:37:43 candidates=1225 corpus=7937 coverage=81176 exec total=36685 (146/sec) pending=4 reproducing=0
2025/03/06 21:37:53 candidates=1194 corpus=7973 coverage=81332 exec total=37338 (143/sec) pending=4 reproducing=0
2025/03/06 21:38:03 candidates=1164 corpus=7995 coverage=81388 exec total=37865 (139/sec) pending=4 reproducing=0
2025/03/06 21:38:16 candidates=1138 corpus=8011 coverage=81426 exec total=38827 (138/sec) pending=4 reproducing=0
2025/03/06 21:38:26 candidates=1117 corpus=8031 coverage=81541 exec total=39525 (135/sec) pending=4 reproducing=0
2025/03/06 21:38:36 candidates=580 corpus=8043 coverage=81572 exec total=40405 (134/sec) pending=4 reproducing=0
2025/03/06 21:38:49 candidates=3 corpus=8062 coverage=81668 exec total=41370 (133/sec) pending=4 reproducing=0
2025/03/06 21:38:59 candidates=1 corpus=8069 coverage=81723 exec total=41909 (130/sec) pending=4 reproducing=0

```

candidates	1
corpus	8075
coverage	81727
exec total	42617 (129/sec)
pending	4
reproducing	0
crash types	1
crashes	4
suppressed	0
syscalls	2245
uptime	332 sec

Crashes:

<u>Description</u>	<u>Count</u>	<u>First Time</u>	<u>Last Time</u>	<u>Report</u>
can't ssh into the instance	2	2025/02/16 21:19	2025/02/16 21:19	
failed to read from qemu: EOF	16	2025/02/16 16:50	2025/02/16 16:50	
KASAN: stack-out-of-bounds Read in profile_pc	3	2025/02/16 21:15	2025/02/16 21:15	has C repro
no output from test machine	2	2025/02/16 22:17	2025/02/16 22:17	
SYZFAIL: mmap of data segment failed	95	2025/03/06 21:35	2025/03/06 21:35	
WARNING in blk_rq_map_user_iov	4	2025/02/16 19:45	2025/02/16 19:45	has C repro
WARNING: zero-size vmalloc in corrupted	2	2025/02/16 17:53	2025/02/16 17:53	
WARNING: zero-size vmalloc in sel_write_load	7	2025/02/16 19:46	2025/02/16 19:46	has C repro

Log:

```

2025/03/06 21:35:54 candidates=5942 corpus=3158 coverage=55121 exec total=14917 (98/sec) pending=4 reproducing=0
2025/03/06 21:36:04 candidates=5942 corpus=3158 coverage=55121 exec total=14917 (92/sec) pending=4 reproducing=0
2025/03/06 21:36:14 candidates=5546 corpus=3398 coverage=56569 exec total=16240 (94/sec) pending=4 reproducing=0
2025/03/06 21:36:27 candidates=4675 corpus=4533 coverage=62973 exec total=20439 (112/sec) pending=4 reproducing=0
2025/03/06 21:36:37 candidates=3832 corpus=5364 coverage=65724 exec total=24007 (125/sec) pending=4 reproducing=0
2025/03/06 21:36:47 candidates=3088 corpus=6113 coverage=71600 exec total=27205 (135/sec) pending=4 reproducing=0
2025/03/06 21:36:57 candidates=2414 corpus=6766 coverage=74851 exec total=30112 (142/sec) pending=4 reproducing=0
2025/03/06 21:37:10 candidates=1785 corpus=7391 coverage=77797 exec total=32909 (148/sec) pending=4 reproducing=0
2025/03/06 21:37:20 candidates=1385 corpus=7792 coverage=80210 exec total=34911 (151/sec) pending=4 reproducing=0
2025/03/06 21:37:30 candidates=1269 corpus=7895 coverage=81027 exec total=35787 (148/sec) pending=4 reproducing=0
2025/03/06 21:37:43 candidates=1225 corpus=7937 coverage=81176 exec total=36685 (146/sec) pending=4 reproducing=0
2025/03/06 21:37:53 candidates=1194 corpus=7973 coverage=81332 exec total=37338 (143/sec) pending=4 reproducing=0
2025/03/06 21:38:03 candidates=1164 corpus=7995 coverage=81388 exec total=37865 (139/sec) pending=4 reproducing=0
2025/03/06 21:38:16 candidates=1138 corpus=8011 coverage=81426 exec total=38827 (138/sec) pending=4 reproducing=0
2025/03/06 21:38:26 candidates=1117 corpus=8031 coverage=81541 exec total=39525 (135/sec) pending=4 reproducing=0
2025/03/06 21:38:36 candidates=580 corpus=8043 coverage=81572 exec total=40405 (134/sec) pending=4 reproducing=0
2025/03/06 21:38:49 candidates=3 corpus=8062 coverage=81668 exec total=41370 (133/sec) pending=4 reproducing=0
2025/03/06 21:38:59 candidates=1 corpus=8069 coverage=81723 exec total=41909 (130/sec) pending=4 reproducing=0
    
```



syzkaller



stats



coverage



syscalls



corpus



VMs



config

candidates	1
corpus	8075
coverage	81727
exec total	42617 (129/sec)
pending	4
reproducing	0
crash types	1
crashes	4
suppressed	0
syscalls	2245
uptime	332 sec

crash types	1
crashes	4
suppressed	0
syscalls	2245
uptime	332 sec

Crashes:

<u>Description</u>	<u>Count</u>	<u>First Time</u>	<u>Last Time</u>
can't ssh into the instance	2	2025/02/16 21:19	2025/02/16 21:19
failed to read from qemu: EOF	16	2025/02/16 16:50	2025/02/16 16:50
KASAN: stack-out-of-bounds Read in profile_pc	3	2025/02/16 21:15	2025/02/16 21:15
no output from test machine	2	2025/02/16 22:17	2025/02/16 22:17
SYZFAIL: mmap of data segment failed	95	2025/03/06 21:35	2025/03/06 21:35
WARNING in blk_rq_map_user_iov	4	2025/02/16 19:45	2025/02/16 19:45
WARNING: zero-size vmalloc in corrupted	2	2025/02/16 17:53	2025/02/16 17:53
WARNING: zero-size vmalloc in sel_write_load	7	2025/02/16 19:46	2025/02/16 19:46

Log:

```

2025/03/06 21:35:54 candidates=5942 corpus=3158 coverage=55121 exec total=14917 (98/sec) pending=4 reproducing=
2025/03/06 21:36:04 candidates=5942 corpus=3158 coverage=55121 exec total=14917 (92/sec) pending=4 reproducing=
2025/03/06 21:36:14 candidates=5546 corpus=3398 coverage=56569 exec total=16240 (94/sec) pending=4 reproducing=
2025/03/06 21:36:27 candidates=4675 corpus=4533 coverage=62973 exec total=20439 (112/sec) pending=4 reproducing=
2025/03/06 21:36:37 candidates=3832 corpus=5364 coverage=65724 exec total=24007 (125/sec) pending=4 reproducing=

```

syzkaller

📊 stats
📄 coverage
🖨️ syscalls
🗄️ corpus
🖥️ VMs
🔧 config

candidates	1
corpus	8075
coverage	81727
exec total	42617 (129/sec)
pending	4
reproducing	0
crash types	1
crashes	4
suppressed	0
syscalls	2245
uptime	332 sec

Crashes:

Description	Count	First Time	Last Time	Report
can't ssh into the instance	2	2025/02/16 21:19	2025/02/16 21:19	
failed to read from qemu: EOF	16	2025/02/16 16:50	2025/02/16 16:50	
KASAN: stack-out-of-bounds Read in profile_pc	3	2025/02/16 21:15	2025/02/16 21:15	has C repro
no output from test machine	2	2025/02/16 22:17	2025/02/16 22:17	
SYZFAIL: mmap of data segment failed	95	2025/03/06 21:35	2025/03/06 21:35	
WARNING in blk_rq_map_user_iov	4	2025/02/16 19:45	2025/02/16 19:45	has C repro
WARNING: zero-size vmalloc in corrupted	2	2025/02/16 17:53	2025/02/16 17:53	
WARNING: zero-size vmalloc in sel_write_load	7	2025/02/16 19:46	2025/02/16 19:46	has C repro

Log:

```

2025/03/06 21:35:54 candidates=5942 corpus=3158 coverage=55121 exec total=14917 (98/sec) pending=4 reproducing=0
2025/03/06 21:36:04 candidates=5942 corpus=3158 coverage=55121 exec total=14917 (92/sec) pending=4 reproducing=0
2025/03/06 21:36:14 candidates=5546 corpus=3398 coverage=56569 exec total=16240 (94/sec) pending=4 reproducing=0
2025/03/06 21:36:27 candidates=4675 corpus=4533 coverage=62973 exec total=20439 (112/sec) pending=4 reproducing=0
2025/03/06 21:36:37 candidates=3832 corpus=5364 coverage=65724 exec total=24007 (125/sec) pending=4 reproducing=0
2025/03/06 21:36:47 candidates=3088 corpus=6113 coverage=71600 exec total=27205 (135/sec) pending=4 reproducing=0
2025/03/06 21:36:57 candidates=2414 corpus=6766 coverage=74851 exec total=30112 (142/sec) pending=4 reproducing=0
2025/03/06 21:37:10 candidates=1785 corpus=7391 coverage=77797 exec total=32909 (148/sec) pending=4 reproducing=0
2025/03/06 21:37:20 candidates=1385 corpus=7792 coverage=80210 exec total=34911 (151/sec) pending=4 reproducing=0
2025/03/06 21:37:30 candidates=1269 corpus=7895 coverage=81027 exec total=35787 (148/sec) pending=4 reproducing=0
2025/03/06 21:37:43 candidates=1225 corpus=7937 coverage=81176 exec total=36685 (146/sec) pending=4 reproducing=0
2025/03/06 21:37:53 candidates=1194 corpus=7973 coverage=81332 exec total=37338 (143/sec) pending=4 reproducing=0
2025/03/06 21:38:03 candidates=1164 corpus=7995 coverage=81388 exec total=37865 (139/sec) pending=4 reproducing=0
2025/03/06 21:38:16 candidates=1138 corpus=8011 coverage=81426 exec total=38827 (138/sec) pending=4 reproducing=0
2025/03/06 21:38:26 candidates=1117 corpus=8031 coverage=81541 exec total=39525 (135/sec) pending=4 reproducing=0
2025/03/06 21:38:36 candidates=580 corpus=8043 coverage=81572 exec total=40405 (134/sec) pending=4 reproducing=0
2025/03/06 21:38:49 candidates=3 corpus=8062 coverage=81668 exec total=41370 (133/sec) pending=4 reproducing=0
2025/03/06 21:38:59 candidates=1 corpus=8069 coverage=81723 exec total=41909 (130/sec) pending=4 reproducing=0
    
```

arch/x86	9%	of 15930
block	28%	of 9462
certs	---	of 19
crypto	4%	of 3635
drivers	3%	of 219088
fs	31%	of 75863
include	100%	of 432
init	6%	of 274
io_uring	16%	of 5681
ipc	56%	of 2471
kernel	20%	of 47884
▶ bpf	4%	of 782
▶ cgroup	25%	of 4231
▶ dma	3%	of 954
▶ events	25%	of 5045
▶ futex	42%	of 925
▶ irq	1%	of 2896
▶ locking	100%	of 1
▶ module	7%	of 1141
▶ power	18%	of 2059
▶ printk	40%	of 975
▼ time	32%	of 4275
▶ alarmtimer.c	31%	of 222
▶ clockevents.c	7%	of 186
▶ clocksource.c	---	of 356
▶ hrtimer.c	41%	of 392
▶ timer.c	49%	of 143
▶ jiffies.c	---	of 2
▶ namespace.c	31%	of 147
▶ ntp.c	51%	of 142
▶ posix-clock.c	43%	of 68
▶ posix-cpu-timers.c	62%	of 342
▶ posix-timers.c	45%	of 487
▶ tick-broadcast-hrtimer.c	---	of 4
▶ tick-broadcast.c	2%	of 218
▶ tick-common.c	2%	of 97
▶ tick-oneshot.c	19%	of 22
▶ tick-sched.c	7%	of 203
▶ time.c	33%	of 204
▶ timeconv.c	67%	of 15
▶ timecounter.c	---	of 6
▶ timekeeping.c	45%	of 378
▶ timekeeping_debug.c	---	of 10
▶ timer.c	22%	of 559
▶ timer_list.c	85%	of 59
▶ vsyscall.c	70%	of 13
▶ trace	3%	of 9026
▶ acct.c	11%	of 169
▶ async.c	19%	of 59
▶ audit.c	19%	of 584

arch/x86	9%	of 15930
block	28%	of 9462
certs	---	of 19
crypto	4%	of 3635
drivers	3%	of 219088
fs	31%	of 75863
include	100%	of 432
init	6%	of 274
io_uring	16%	of 5681
ipc	56%	of 2471
kernel	20%	of 47884
bpf	4%	of 782
cgroup	25%	of 4231
dma	3%	of 954
events	25%	of 5045
futex	42%	of 925
irq	1%	of 2896
locking	100%	of 1
module	7%	of 1141
power	18%	of 2059
printk	40%	of 975
time	32%	of 4275
alarmtimer.c	31%	of 222
clockevents.c	7%	of 186
clocksource.c	---	of 356
hrtimer.c	41%	of 392
itimer.c	49%	of 143
jiffies.c	---	of 2
namespace.c	31%	of 147
ntp.c	51%	of 142
posix-clock.c	43%	of 68
posix-cpu-timers.c	62%	of 342
posix-timers.c	45%	of 487
tick-broadcast-hrtimer.c	---	of 4
tick-broadcast.c	2%	of 218
tick-common.c	2%	of 97
tick-oneshot.c	19%	of 22
tick-sched.c	7%	of 203
time.c	33%	of 204
timeconv.c	67%	of 15
timecounter.c	---	of 6
timekeeping.c	45%	of 378
timekeeping_debug.c	---	of 10
timer.c	22%	of 559
timer_list.c	85%	of 59
vsyscall.c	70%	of 13
trace	3%	of 9026
acct.c	11%	of 169
async.c	19%	of 59
audit.c	19%	of 584

```

7 #include <linux/device.h>
8 #include <linux/export.h>
9 #include <linux/file.h>
10 #include <linux/posix-clock.h>
11 #include <linux/slab.h>
12 #include <linux/syscalls.h>
13 #include <linux/uaccess.h>
14
15 #include "posix-timers.h"
16
17 /*
18  * Returns NULL if the posix_clock instance attached to 'fp' is old and stale.
19  */
20 static struct posix_clock *get_posix_clock(struct file *fp)
21 {
22     struct posix_clock *clk = fp->private_data;
23
24     down_read(&clk->rwsem);
25
26     if (!clk->zombie)
27         return clk;
28
29     up_read(&clk->rwsem);
30
31     return NULL;
32 }
33
34 static void put_posix_clock(struct posix_clock *clk)
35 {
36     up_read(&clk->rwsem);
37 }
38
39 static ssize_t posix_clock_read(struct file *fp, char __user *buf,
40                               size_t count, loff_t *ppos)
41 {
42     struct posix_clock *clk = get_posix_clock(fp);
43     int err = -EINVAL;
44
45     if (!clk)
46         return -ENODEV;
47
48     if (clk->ops.read)
49         err = clk->ops.read(clk, fp->f_flags, buf, count);
50
51     put_posix_clock(clk);
52
53     return err;
54 }
55
56 static __poll_t posix_clock_poll(struct file *fp, poll_table *wait)
57 {
58     struct posix_clock *clk = get_posix_clock(fp);
59     __poll_t result = 0;
60
61     if (!clk)
62         return EPOLLERR;
63
64     if (clk->ops.poll)
65         result = clk->ops.poll(clk, fp, wait);
66
67     put_posix_clock(clk);
68
69     return result;
70 }
71

```

Covered: black (#000000)

All PC values associated to that line are covered. There is number on the left side indicating how many programs have triggered executing the PC values associated to this line. You can click on that number and it will open last executed program. Example below shows how single line which is fully covered is shown.

Both: orange (#c86400)

There are several PC values associated to the line and not all of these are executed. Again there is number left to the source code line that can be clicked to open last program triggering associated PC values. Example below shows single line which has both executed and non-executed PC values associated to it.

Weak-uncovered: crimson red (#c80000)

Function (symbol) this line is in doesn't have any coverage. I.e. the function is not executed at all. Please note that if compiler have optimized certain symbol out and made the code inline instead symbol associated with this line is the one where the code is compiled into. This makes it sometimes real hard to figure out meaning of coloring. Example below shows how single line which is uncovered and PC values associated to it are in function(s) that are not executed either is shown.

Uncovered: red (#ff0000)

Line is uncovered. Function (symbol) this line is in is executed and one of the PC values associated to this line. Example below shows how single line which is not covered is shown.

Not instrumented: grey (#505050)

PC values associated to the line are not instrumented or source line doesn't generate code at all. Example below shows how all not instrumented code is shown.

Per-syscall coverage:

Syscall	Inputs	Total	Coverage	Cover overflows	Comps overflows	Prio
sys_mount_image\$vfat [7232]	12	62	7476	1384	0	prio
sys_read_part_table [7287]	26	49	8634	798	0	prio
mlockall [3998]	17	34	3019	765	0	prio
write [7378]	132	182	16878	530	0	prio
sys_clone3 [7090]	39	68	9826	491	0	prio
ioctl\$SRNDADDENTROPY [2011]	5	10	324	406	0	prio
sys_mount_image\$ext4 [7203]	149	453	21525	357	0	prio
openat\$selinux_policy [4579]	4	8	2270	349	0	prio
sys_clone [7089]	59	274	9295	298	0	prio
syslog [7067]	17	19	1252	155	0	prio
sys_mount_image\$msdos [7215]	38	88	11976	136	0	prio
perf_event_open\$group [4663]	75	108	3674	85	0	prio
sys_mount_image\$iso9660 [7211]	51	131	11569	70	0	prio
writew [7789]	41	50	6299	65	0	prio
read [5003]	132	165	8081	60	0	prio
write\$selinux_user [7751]	11	11	739	27	0	prio
move_pages [4072]	23	24	1099	17	0	prio
keyctl\$KEYCTL_CAPABILITIES [3802]	6	6	566	15	0	prio
openat2\$dir [4659]	22	30	4258	11	0	prio
perf_event_open [4661]	20	31	2806	6	0	prio
readv [5324]	125	144	6965	4	0	prio
ioctl\$VT_DISALLOCATE [2638]	4	5	1331	3	0	prio
mount\$tmpfs [4067]	61	114	9581	3	0	prio
openat\$binfmt [4477]	14	87	8052	3	0	prio
unshare [7354]	24	58	7350	3	0	prio
close [209]	109	245	12265	2	0	prio
ioctl\$BLKTRACESTR [935]	4	6	1674	2	0	prio
ioctl\$SSG_BLKTRACESTR [2055]	3	3	2203	2	0	prio
seccomp\$SECCOMP_SET_MODE_FILTER_LISTENER [5425]	29	39	3451	2	0	prio
setsockopt\$inet_udp_encap [6861]	4	7	554	2	0	prio
ioctl\$SSG_BLKTRACESTR [2058]	3	4	1611	1	0	prio
mlock [3994]	21	63	2719	1	0	prio
mmap [4000]	50	498	6646	1	0	prio
preadv [4848]	59	64	3635	1	0	prio
setsockopt\$inet6_udp_encap [6742]	2	3	303	1	0	prio
setsockopt\$inet_tcp_MD5SIG [6850]	2	4	402	1	0	prio

Inputs: число входных значений в корпусе, добавленных из-за этого вызова.

Total: общее количество входных значений в корпусе, содержащих этот вызов

Coverage: покрытие от этого вызова

В заключение

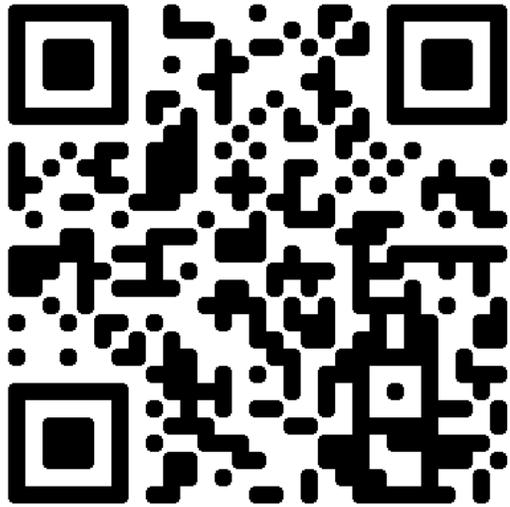
SoK: Unraveling the Veil of OS Kernel Fuzzing

Jiacheng Xu, He Sun, Shihao Jiang, Qinying Wang, Mingming Zhang, Xiang Li, Kaiwen Shen, Peng Cheng, Jiming Chen, Charles Zhang, and Shouling Ji

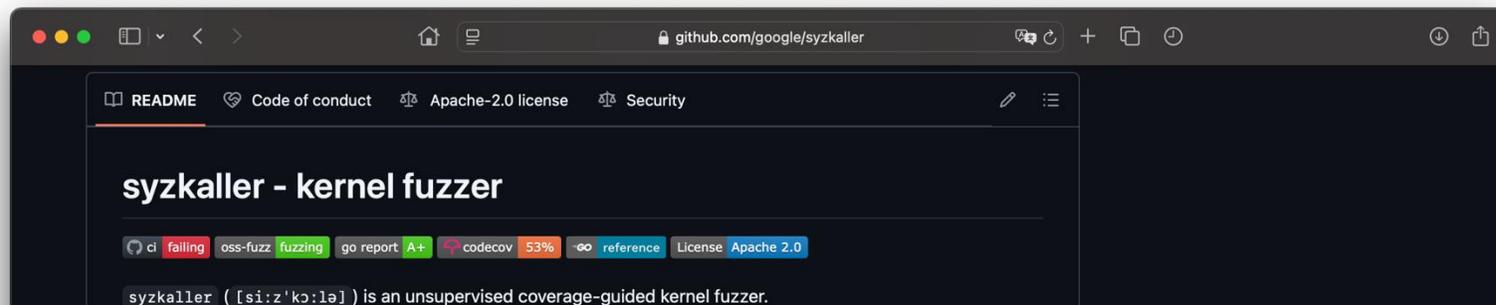
Table 1: Summary of OS kernel fuzzers and their corresponding functionalities, types, critical techniques and fuzzing interfaces.

Stage	Method	Functionality	Type	Technique	Input Interface			
					peripheral	syscall	filesystem	network
Execution environment	PeriScope [142]	1.1	○	On-device fuzzing	●			
	TEEz [24]	1.1 2.1 3.3	○			●		
	USBfuzz [121]	1.1	○		●			
	ECMO [72]	1.1	○	Emulation-based fuzzing		●		
	FirmSolo [14]	1.1	●				●	
	Greenhouse [151]	1.1	●				●	
	KextFuzz [166]	1.2 2.1	○	Invasive instrumentation		●		
	R2D2 [135]	1.2 3.3	○					
	kAFL [132]	1.1 1.2	○	Non-invasive tracing			●	
	μAFL [93]	1.2	○					
	OASIS [60]	1.2	○				●	
	BoKASAN [33]	1.3	●				●	
	BVF [147]	1.3	○	Oracle for memory corruption		●		
	Digtools [118]	1.1 1.3	●			●		
	Hydra [82]	1.3	○	Oracle for non-crash bugs			●	
	Monarch [103]	1.3	○				●	
DroneSecurity [129]	1.1 1.3	●					●	
Input specification	Janus [164]	2.1	○	Multi-dimensional input		●	●	
	PrIntFuzz [104]	1.1 2.1	○		●	●		
	DevFuzz [159]	2.1 3.2	○		●	●		
	SATURN [167]	2.1	○		●	●		
	Difuze [36]	2.2	●		●			
	SyzGen [31]	1.2 2.2 2.3	○		●			
	NtFuzz [34]	2.2	●	Format recovery		●		
	Dr. Fuzz [179]	1.2 2.2	○		●			
	FANS [97]	2.2 2.3	●		●			
	SyzDescribe [54]	2.2	○		●			
	IMF [53]	2.3	●	Explicit dependency		●		
	Dogfood [28]	2.3	○			●		
	MoonShine [117]	2.3	○	Implicit dependency		●		
	HEALER [145]	2.3	○			●		
	MOCK [162]	2.3 3.2	○			●		
	Fuzzing	FIRM-AFL [180]	3.1	○	Virtualization enhancement		●	
Thunderkaller [88]		3.1	○			●		
Agamoto [143]		3.1	○	System snapshot		●		
ReUSB [66]		2.1 3.1 3.2	○		●	●		
CAB-Fuzz [83]		3.2	○		●			
HFL [78]		2.3 3.2	○	Constraint solving		●		
SFuzz [30]		3.2	○			●		
SyzVegas [155]		3.2	○	Decision intelligence		●		
Puzzer [60]		3.2	○			●		

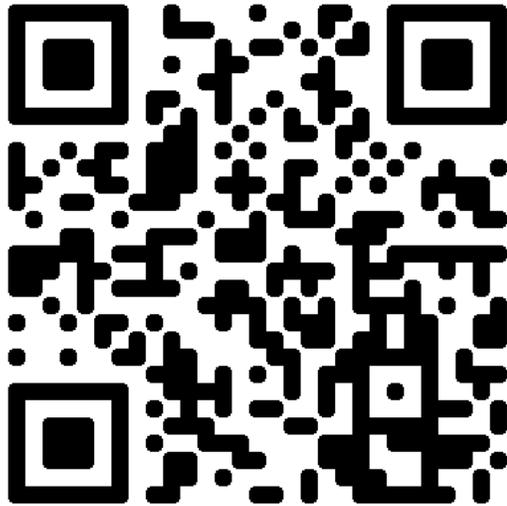
Полезные ссылки



<https://github.com/google/syzkaller>



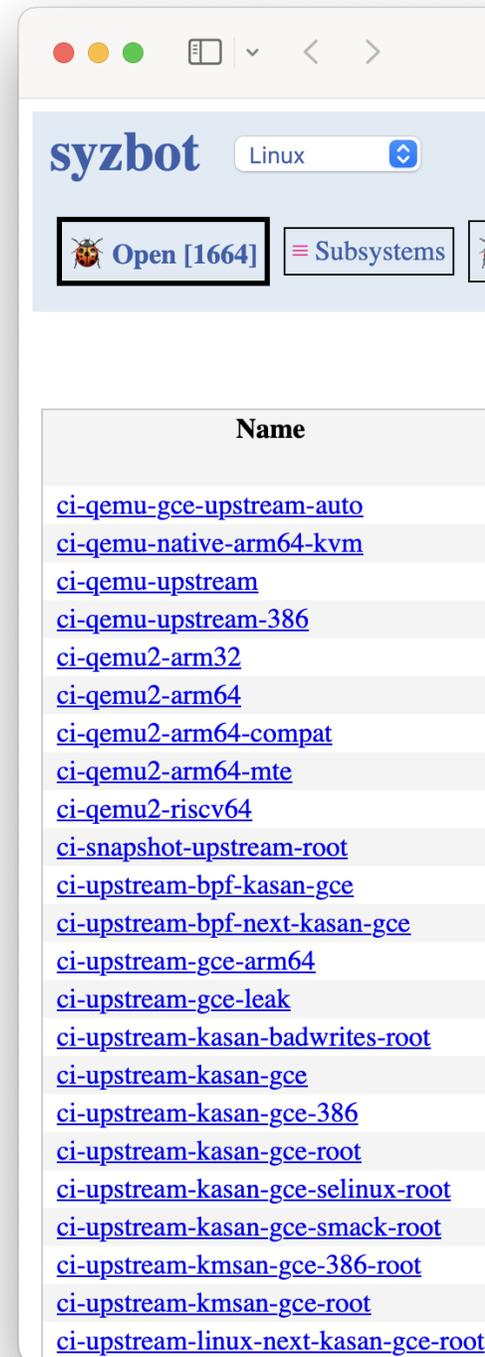
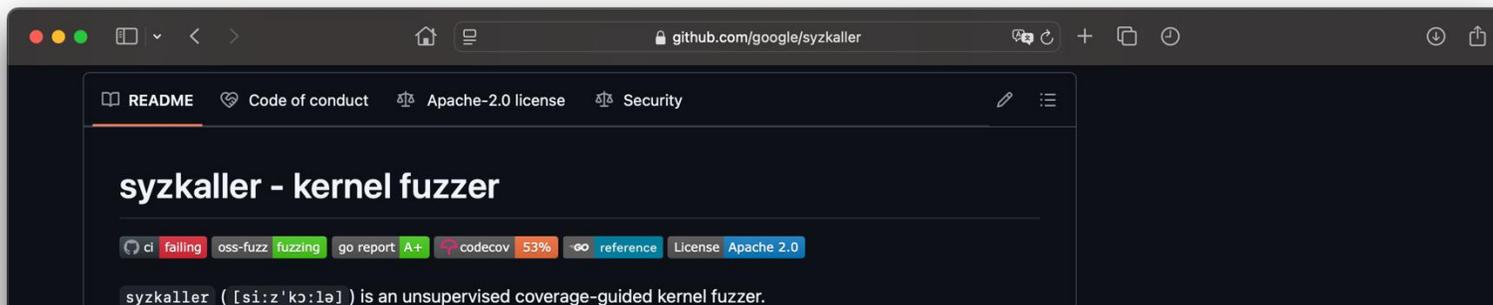
Полезные ссылки



<https://github.com/google/syzkaller>



<https://syzkaller.appspot.com/upstream>



Спасибо за внимание

```
1  #include <unistd.h>
2  #include <stdlib.h>
3  #include <stdio.h>
4
5  int main() {
6      for (int i = 0; i < 1000; i++) {
7          int pid = fork();
8          if (pid == 0) {
9              printf("Fuzzing starts: %d\n", i);
10             usleep(100000);
11             if (i % 13 == 0) {
12                 exit(44);
13             } else {
14                 exit(0);
15             }
16         }
17         int status = 0;
18         wait(&status);
19         printf("Finish: %d with status %d\n", i, WEXITSTATUS(status));
20     }
21     printf("Fuzzing campaign is finished\n");
22 }
23
```