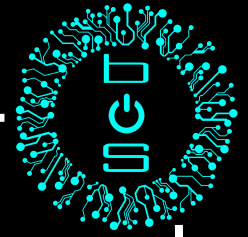


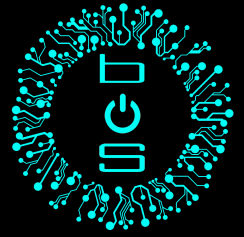
Evolution of kernel fuzzers in NetBSD

...

Siddharth Muralee

Team bi0s

>_ $ whoami



- Siddharth Muralee (R3x)
- Third year BTech CSE @ Amrita Vishwa Vidyapeetham
- CTF player - Team **bi0s**
- **Reverse Engineering** and **Exploitation**
- Core organising team @ **InCTF** and **InCTFj**
- Contributor to **NetBSD** (GSoC ‘18)
  - Kernel Code Quality improvement team
  - Security team

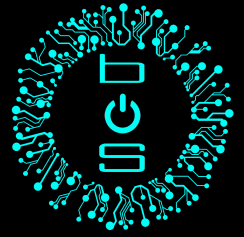
>_ The NetBSD Project

“Of course it runs NetBSD”

- Unix - like BSD Operating System
- Open Source
- Portability

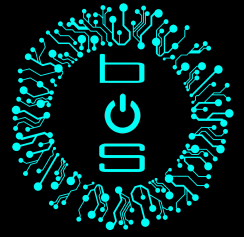
PowerPC, Alpha, SPARC, MIPS, SH3, ARM, amd64, i386, m68k, VAX, ...

>_ Agenda



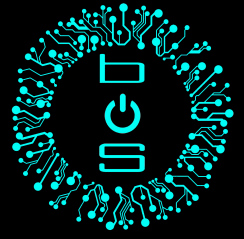
- Issues faced while fuzzing the kernel
- Sanitizers
- Kernel Code Coverage
- Syzkaller
- Future work

>_ Issues faced while fuzzing the kernel

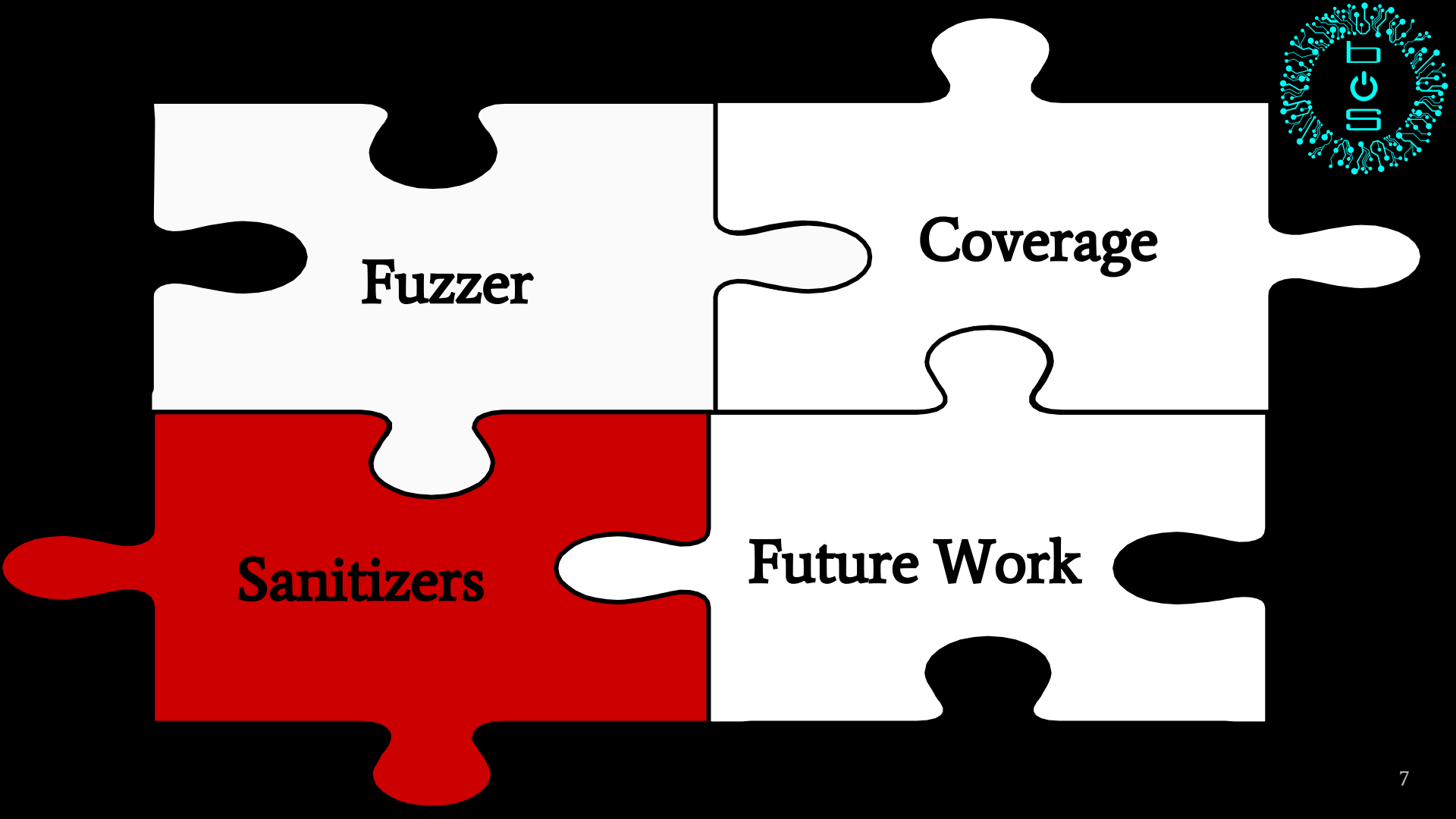


- Setup with fuzzer and VMs
  - Handle Crashes
  - Multiarch support
- Scraping Crashes/logs to generate reports
  - Console output
- Restricted kernel APIs
  - Sandboxing
  - Usermode privilege protection

>_ Issues faced while fuzzing the kernel (Contd.)



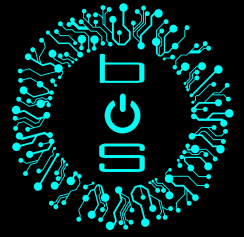
- Generating proper inputs for fuzzing
  - Need to identify proper contexts
- Getting proper reproducers
  - Kernel bug ~= Kernel Panic
  - Identify root cause
- Indetermination of Execution
  - Threads
  - Scheduling


Fuzzer
Coverage
Sanitizers
Future Work

>_ Sanitizers

- Dynamic testing tools
- Compiler Instrumented
- Available with GCC and Clang
- **Fuzzing Aid**

>_ Types of Sanitizers



>_ Address Sanitizer

detects invalid address usage bugs

>_ Undefined Behaviour Sanitizer

finds unspecified code semantic bugs

>_ Memory Sanitizer

finds uninitialized memory access bugs

>_ Thread Sanitizer

detects threading bugs

>_ Kernel Address Sanitizer (KASAN)

- Overflows
- Use after free (UAFs)

Supported in NetBSD :

- amd64
- aarch64

Compile NetBSD kernel with :

```
makeoptions  KASAN=1
options KASAN
```

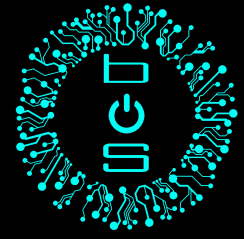
>_ KASAN - Overview

- Poisoning
- Shadow buffer
- Interceptors

```
*address = ...;  // or: ... = *address;
```

```
if (IsPoisoned(address)) {
  ReportError(address, kAccessSize, kIsWrite);
}
*address = ...;  // or: ... = *address;
```

>_ KASAN - sample report



ifconfig gif0 create
ifconfig gif0 up

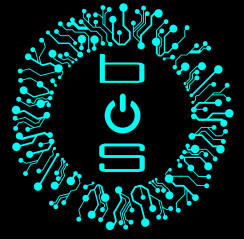
```
[   50.682919] kASan: Unauthorized Access In 0xffffffff80f22655: \
Addr 0xffffffff81b997a0 [8 bytes, read]
[   50.682919] #0 0xffffffff8021ce6a in kasan_memcpy <netbsd>
[   50.692999] #1 0xffffffff80f22655 in m_copyback_internal <netbsd>
   .......
[   50.703622] #13 0xffffffff80fde694 in gif_ioctl <netbsd>
[   50.703622] #14 0xffffffff80fcdb1f in doifioctl <netbsd>
```

>_ KASAN - sample report

ifconfig gif0 create
ifconfig gif0 up

```
[   50.682919] kASan: Unauthorized Access In 0xffffffff80f22655: \
Addr 0xffffffff81b997a0 [8 bytes, read]
[   50.682919] #0 0xffffffff8021ce6a in kasan_memcpy <netbsd>
[   50.692999] #1 0xffffffff80f22655 in m_copyback_internal <netbsd>
   .......
[   50.703622] #13 0xffffffff80fde694 in gif_ioctl <netbsd>
[   50.703622] #14 0xffffffff80fcdb1f in doifioctl <netbsd>
```

>_ KASAN - sample report



ifconfig gif0 create
ifconfig gif0 up

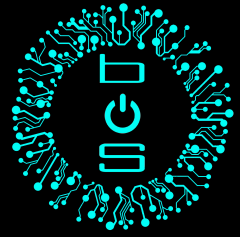
```
[   50.682919] kASan: Unauthorized Access In 0xffffffff80f22655: \
Addr 0xffffffff81b997a0 [8 bytes, read]
[   50.682919] #0 0xffffffff8021ce6a in kasan_memcpy <netbsd>
[   50.692999] #1 0xffffffff80f22655 in m_copyback_internal <netbsd>
   .......
[   50.703622] #13 0xffffffff80fde694 in gif_ioctl <netbsd>
[   50.703622] #14 0xffffffff80fcdb1f in doifioctl <netbsd>
```

>_ KASAN - sample report

ifconfig gif0 create
ifconfig gif0 up

```
[   50.682919] kASan: Unauthorized Access In 0xffffffff80f22655: \
Addr 0xffffffff81b997a0 [8 bytes, read]
[   50.682919] #0 0xffffffff8021ce6a in kasan_memcpy <netbsd>
[   50.692999] #1 0xffffffff80f22655 in m_copyback_internal <netbsd>
   .......
[   50.703622] #13 0xffffffff80fde694 in gif_ioctl <netbsd>
[   50.703622] #14 0xffffffff80fcdb1f in doifioctl <netbsd>
```

>_ Kernel Undefined Behaviour Sanitizer

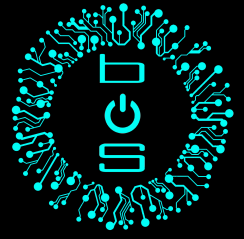


- Signed overflows
- unportable bit shift
- Unaligned memory access

Compile NetBSD kernel with :

```
options KUBSAN
```

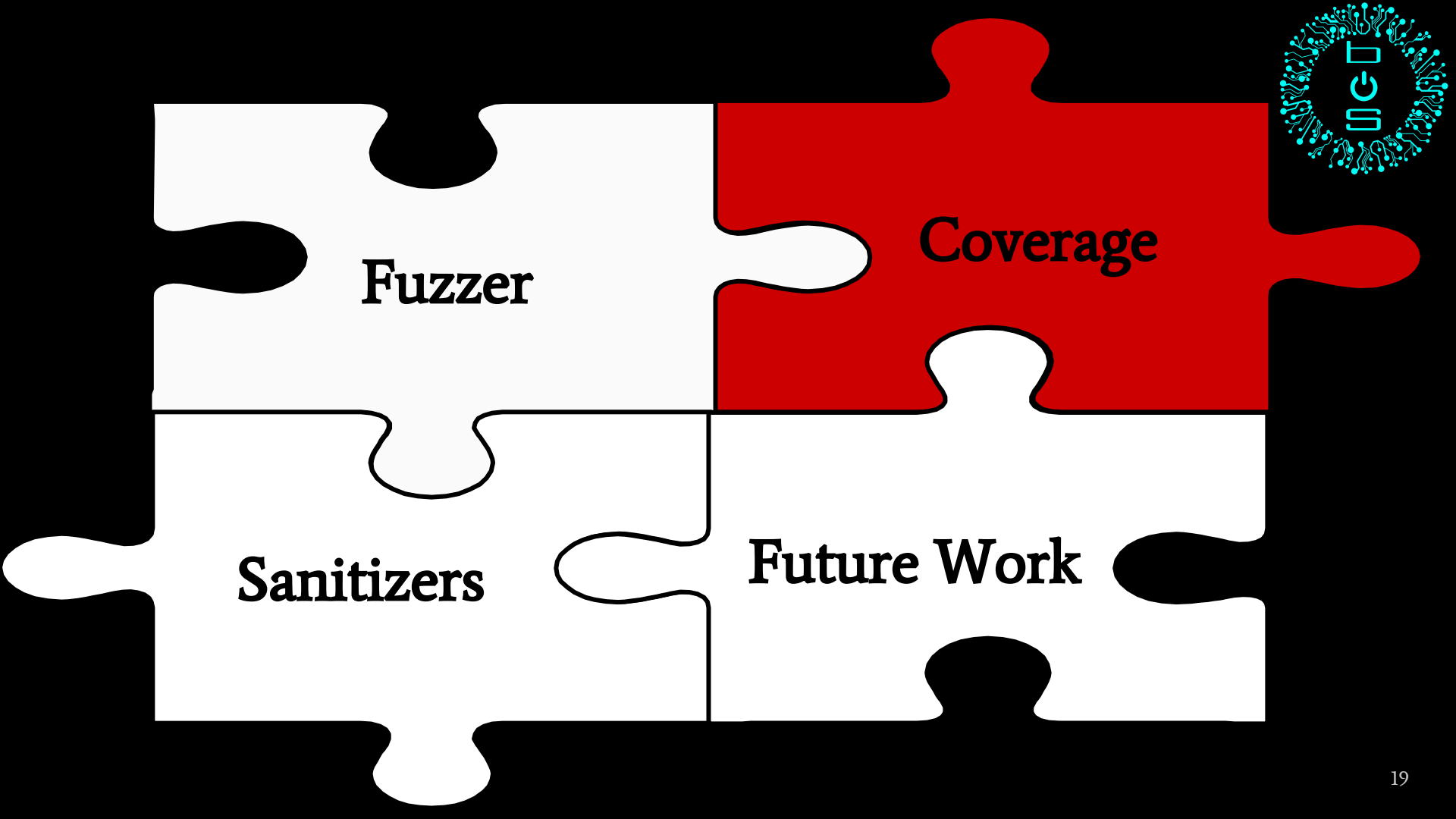
>_ KLEAK



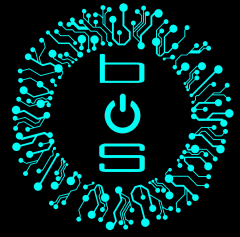
- Developed by
  - Thomas Barabosch
  - Maxime Villard
- Detect kernel **information leaks**
- Uses **taint tracking**
- Scanned buffers passed from kernel to userspace

>_ TODO

- Kernel Memory Sanitizer
- Kernel Thread Sanitizer
- Improve existing Sanitizers


Fuzzer
Coverage
Sanitizers
Future Work

>_ Coverage Guided Fuzzing



- Instrument the target (Compiler/Binary)
- Get path coverage of each input
- Mutate the input based on the coverage.
- Proved to be more efficient than ordinary fuzzing
  - AFL
  - Honggfuzz
  - Syzkaller

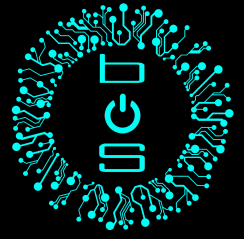
>_ Kernel Code Coverage (KCov)

- Compiler instrumentation
- Supported in GCC and Clang
- Recently ported to NetBSD
- Implemented as a Kernel Module
  - Ioctl - Enable/Disable

Compile NetBSD kernel with :

```
makeoptions  KCOV=1
options KCOV
```

>_ Kernel Code Coverage (KCov)

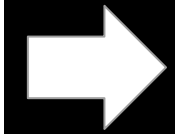


- Two major modes :
  - Trace PC mode
  - Trace Compare mode
- Just add the compiler flag :
  - `-fsanitize-coverage=trace-pc` - PC mode
  - `-fsanitize-coverage=trace-cmp` - Compare mode

>_ Trace PC mode

```
int func1() {
        ....
        ....
        func2();
        return 0;
}

int func2() {
        ...
        ...
        return 1;
}
```

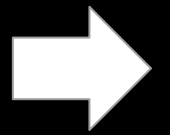


```
int func1() {
        __sanitizer_cov_trace_pc();
        func2();
        __sanitizer_cov_trace_pc();
        return 0;
}

int func2() {
        __sanitizer_cov_trace_pc();
        return 1;
}
```

>_ Trace PC mode (Contd.)

```
if (x == y) {
        ....
        ....
}
```



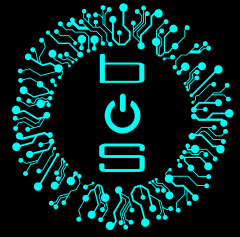
```
__sanitizer_cov_trace_pc();
if (x == y) {
        __sanitizer_cov_trace_pc();
}
__sanitizer_cov_trace_pc();
```

>_ Trace PC mode (Contd.)

- We do not trace during
  - Boot
  - Interrupts
- Returns back addresses traced
- addr2line - used to find the function names.

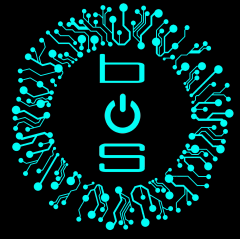
```
void __sanitizer_cov_trace_pc(void) {
        if (in_boot || in_interrupt)
                return;
        count++
        buf[count] = IP
}
```

>_ KCov Example



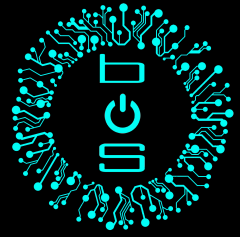

```
fd = open("/dev/kcov", O_RDWR);
ioctl(fd, KCOV_IOC_SETBUFSIZE, &size);
cover = mmap(NULL, PAGE_SIZE * sizeof(unsigned long),
    PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
ioctl(fd, KCOV_IOC_ENABLE);
read(-1, NULL, 0); // syscall target
ioctl(fd, KCOV_IOC_DISABLE)
for (i = 0; i < cover[0]; i++)
        printf("%p\n", (void *)cover[i + 1])
```

>_ KCov Example



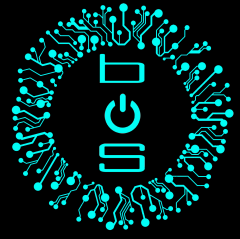
```
fd = open("/dev/kcov", O_RDWR);
ioctl(fd, KCOV_IOC_SETBUFSIZE, &size);
cover = mmap(NULL, PAGE_SIZE * sizeof(unsigned long),
    PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
ioctl(fd, KCOV_IOC_ENABLE);
read(-1, NULL, 0); // syscall target
ioctl(fd, KCOV_IOC_DISABLE)
for (i = 0; i < cover[0]; i++)
        printf("%p\n", (void *)cover[i + 1])
```

>_ KCov Example



```
fd = open("/dev/kcov", O_RDWR);
ioctl(fd, KCOV_IOC_SETBUFSIZE, &size);
cover = mmap(NULL, PAGE_SIZE * sizeof(unsigned long),
    PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
ioctl(fd, KCOV_IOC_ENABLE);
read(-1, NULL, 0); // syscall target
ioctl(fd, KCOV_IOC_DISABLE)
for (i = 0; i < cover[0]; i++)
        printf("%p\n", (void *)cover[i + 1])
```

>_ KCov Example

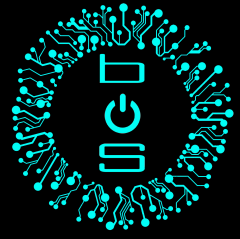


```
fd = open("/dev/kcov", O_RDWR);
ioctl(fd, KCOV_IOC_SETBUFSIZE, &size);
cover = mmap(NULL, PAGE_SIZE * sizeof(unsigned long),
    PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
ioctl(fd, KCOV_IOC_ENABLE);
read(-1, NULL, 0); // syscall target
ioctl(fd, KCOV_IOC_DISABLE)
for (i = 0; i < cover[0]; i++)
        printf("%p\n", (void *)cover[i + 1])
```

>_ KCov Trace

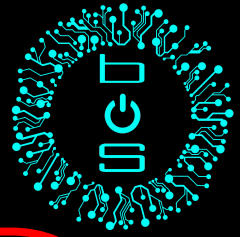
```
localhost# ./a.out
0xffffffff80259ac9
0xffffffff80259aeb
0xffffffff80259b22
0xffffffff80259c21
0xffffffff80259c2e
0xffffffff80bcaff7
0xffffffff80b48a94
0xffffffff80b48b05
0xffffffff80bcb02d
0xffffffff80259c85
0xffffffff80259b8c
0xffffffff80259cd0
0xffffffff80259ce9
```

>_ KCov Trace | addr2line

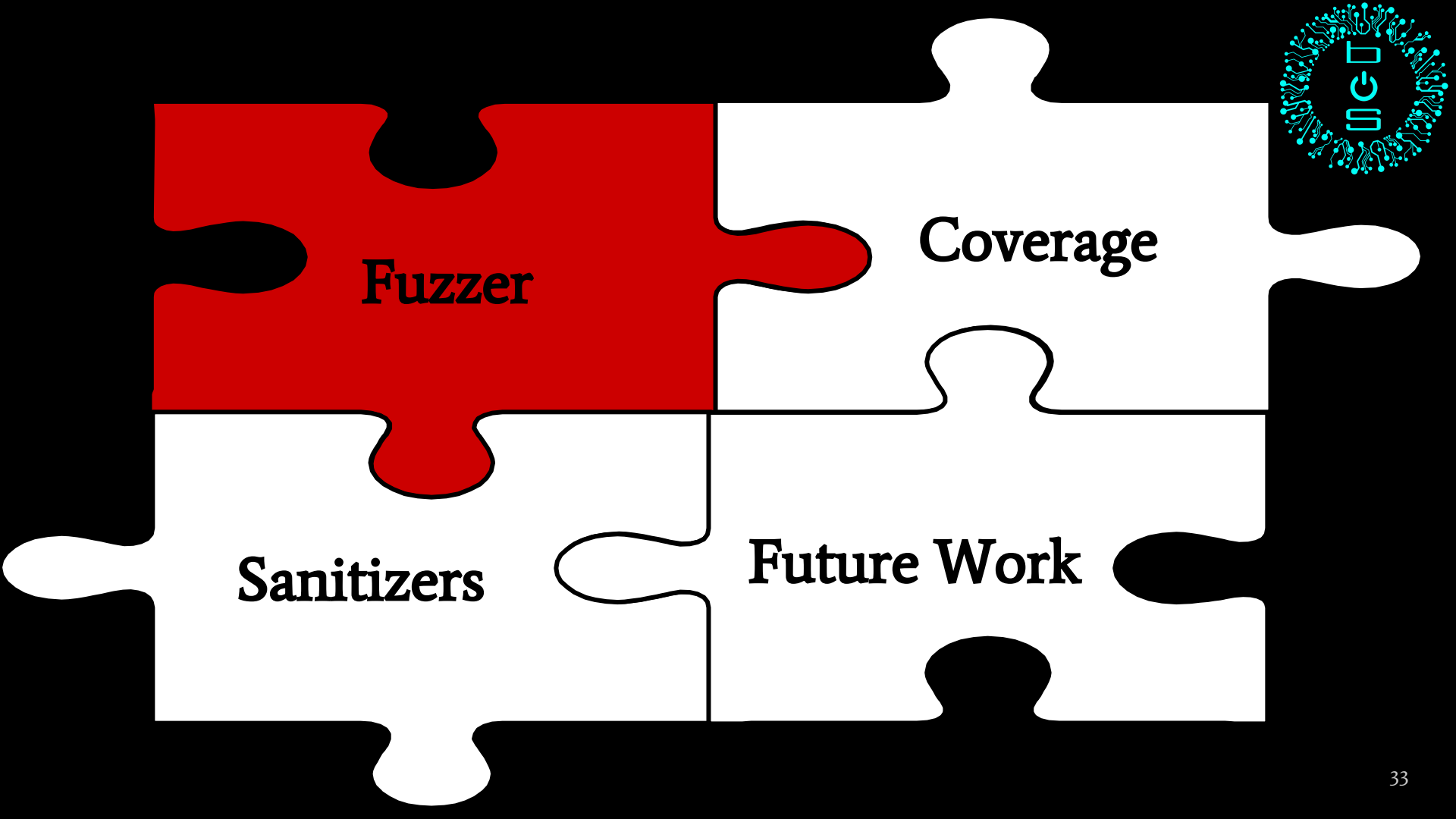


sy_invoke	/src/sys/sys/syscallvar.h:84
sy_invoke	/src/sys/sys/syscallvar.h:86
sys_read	/src/sys/kern/sys_generic.c:110
x86_curlwp	/repos/obj1/sys/arch/amd64/compile/GENERIC/./machine/cpu.h:67
fd_getfile	/src/sys/kern/kern_descrip.c:416
sys_read	/src/sys/kern/sys_generic.c:123
sy_invoke	/src/sys/sys/syscallvar.h:97
syscall	/src/sys/arch/x86/x86/syscall.c:142

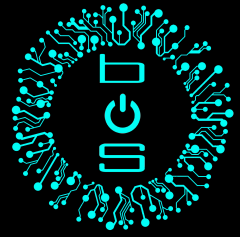
>_ KCov Trace | addr2line



sy_invoke	/src/sys/sys/syscallvar.h:84
sy_invoke	/src/sys/sys/syscallvar.h:86
sys_read	/src/sys/kern/sys_generic.c:110
x86_curlwp	/repos/obj1/sys/arch/amd64/compile/GENERIC/./machine/cpu.h:67
fd_getfile	/src/sys/kern/kern_descrip.c:416
sys_read	/src/sys/kern/sys_generic.c:123
sy_invoke	/src/sys/sys/syscallvar.h:97
syscall	/src/sys/arch/x86/x86/syscall.c:142


Fuzzer
Coverage
Sanitizers
Future Work

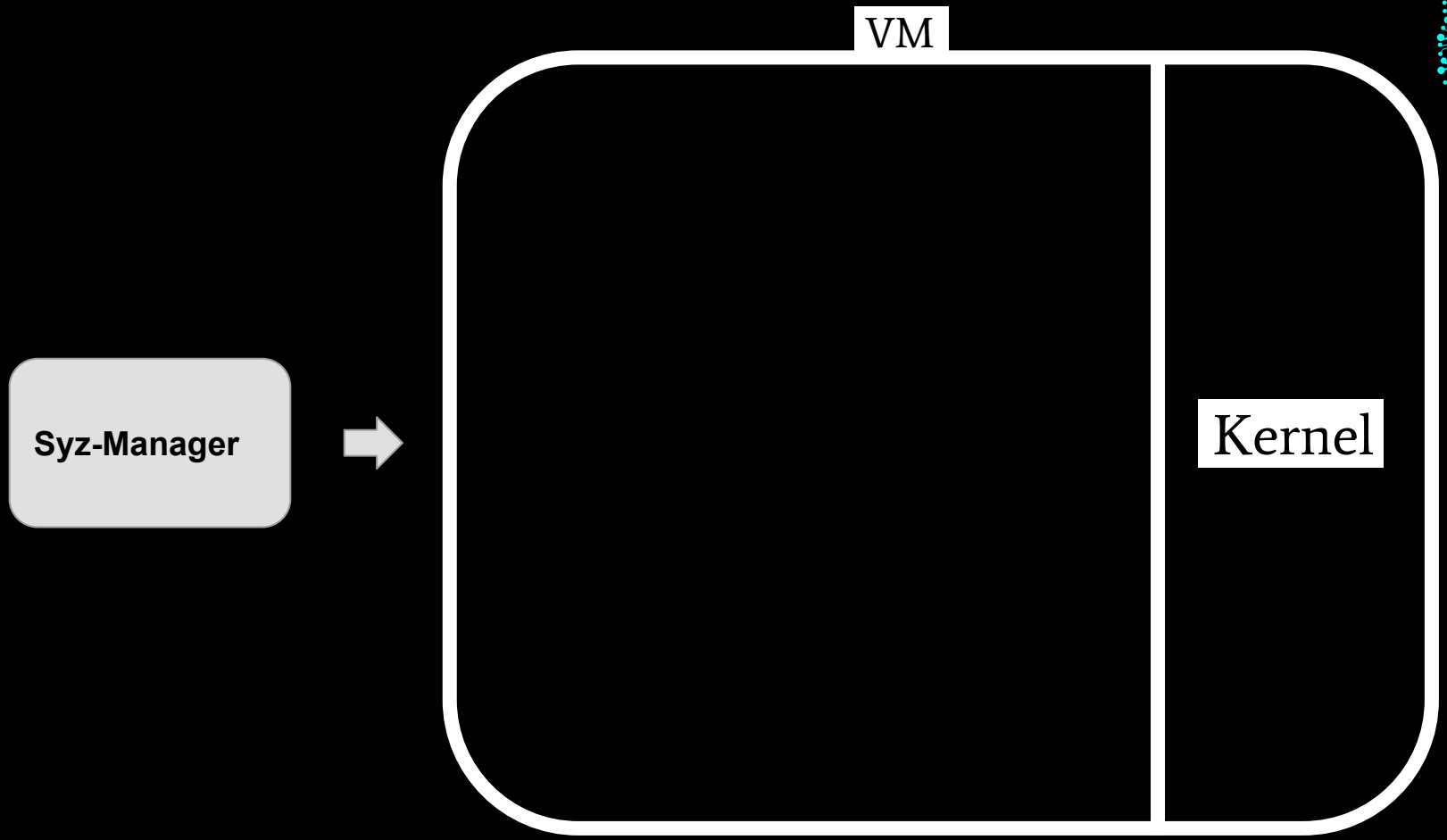
>_ Syzkaller



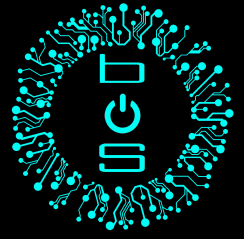
- Coverage Guided **kernel** fuzzer
- Google
- Written in Go and C
- Supported OS's
  - Linux
  - OpenBSD
  - NetBSD
  - Fuchsia
  - Akaros
  - FreeBSD

>_ Syzkaller (Contd.)

- Uses KCov feature
- Primary target : System Calls
- Secondary target :
  - Network Stack
  - Filesystem Stack (Under progress)
- Close to around **2000** bugs


VM
Syz-Manager
Kernel

>_ Sample config (Syz-Manager)

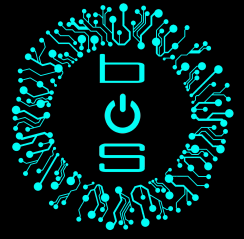


```
{
        "target": "netbsd/amd64",      //Target OS and Architecture
        "workdir":  "work",            //Directory to store crash details
        "syzkaller": "./",             // Syzkaller Directory
        "image": "netbsd.img",
        "sshkey":"netbsdkey",
        "type": "qemu",
        "vm": {
                "qemu": "qemu-system-x86_64",
                "count": 2,
                "cpu": 2,
                "mem": 4048
        }
}
```

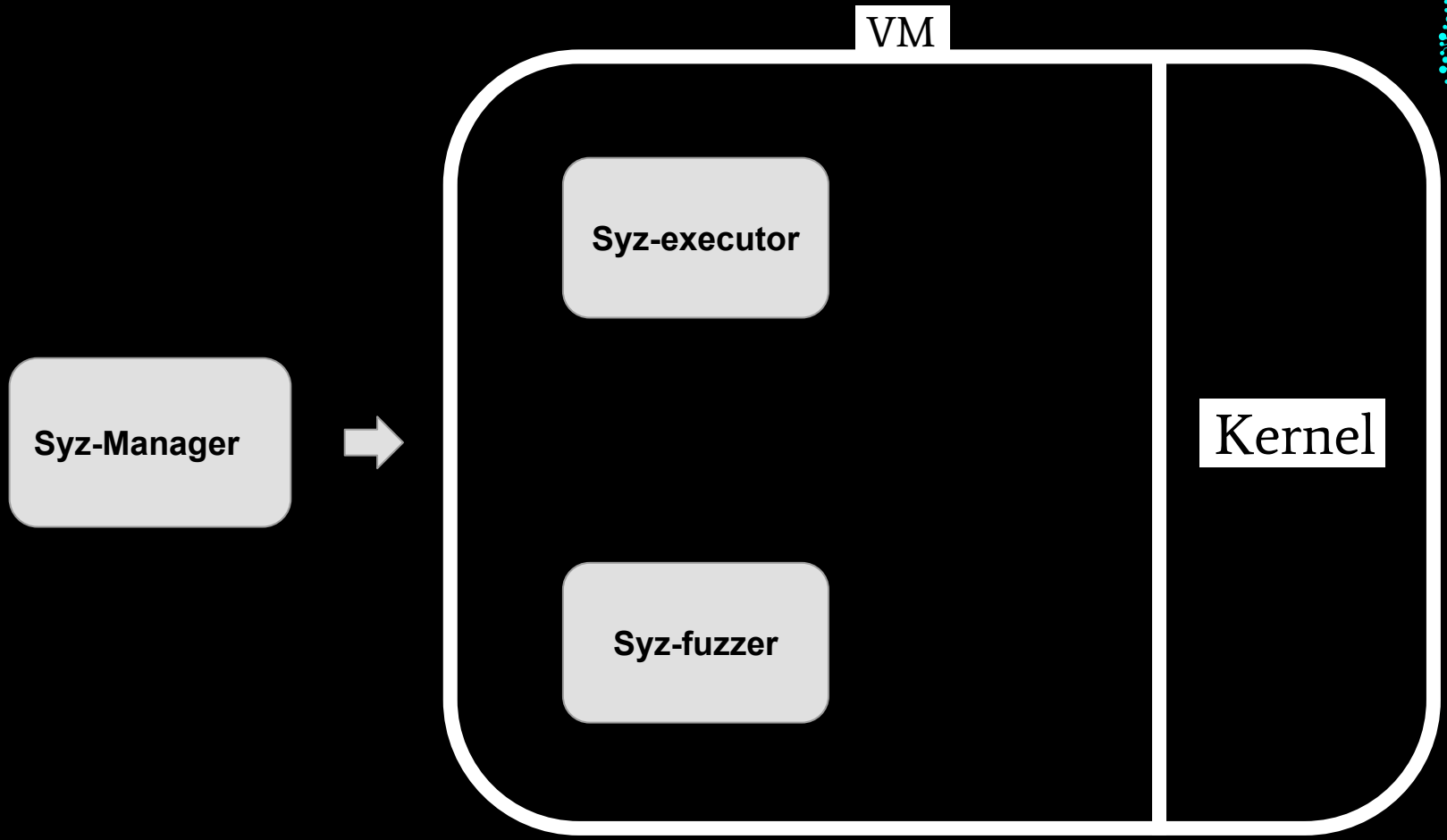
>_ Sample config (Contd.)

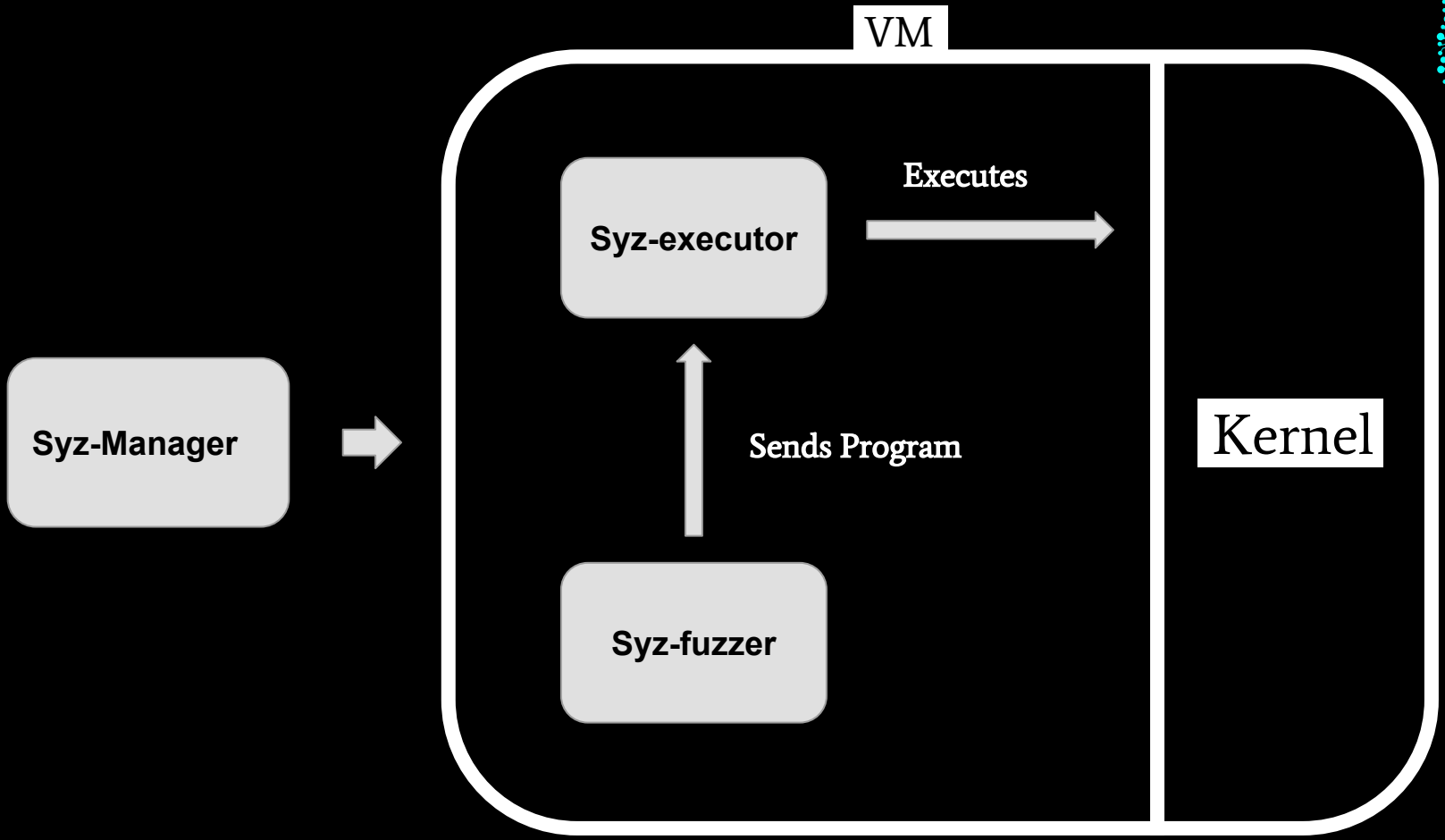
```
{
        "target": "netbsd/amd64",
        "workdir": "work",
        "syzkaller": "./",
        "image": "netbsd.img",          // Disk with NetBSD OS installed
        "sshkey":"netbsdkey",           // Ssh Key for the Installed OS
        "type": "qemu",                 // VM used
        "vm": {
                "qemu": "qemu-system-x86_64",
                "count": 2,
                "cpu": 2,
                "mem": 4048
        }
}
```

>_ Sample config (Contd.)

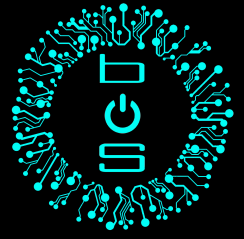


```
{
        "target": "netbsd/amd64",
        "workdir": "work",
        "syzkaller": "./",
        "image": "netbsd.img",
        "sshkey":"netbsdkey",
        "procs": 2,
        "type": "qemu",
        "vm": {                     // Config for VM
                "qemu": "qemu-system-x86_64",   // Qemu binary to be used
                "count": 2,                     // Number of parallel VMs
                "cpu": 2,                       // Number of CPU's in each VM
                "mem": 1024                     // RAM memory for each VM
        }
}
```


VM
Syz-executor
Syz-Manager
Kernel
Syz-fuzzer


VM
Syz-executor
Executes
Syz-Manager
Sends Program
Kernel
Syz-fuzzer

>_ Creating programs

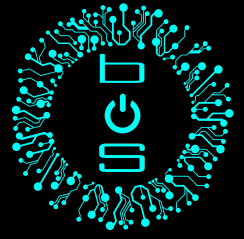


- Scrape kernel codebase
- Generate function prototypes

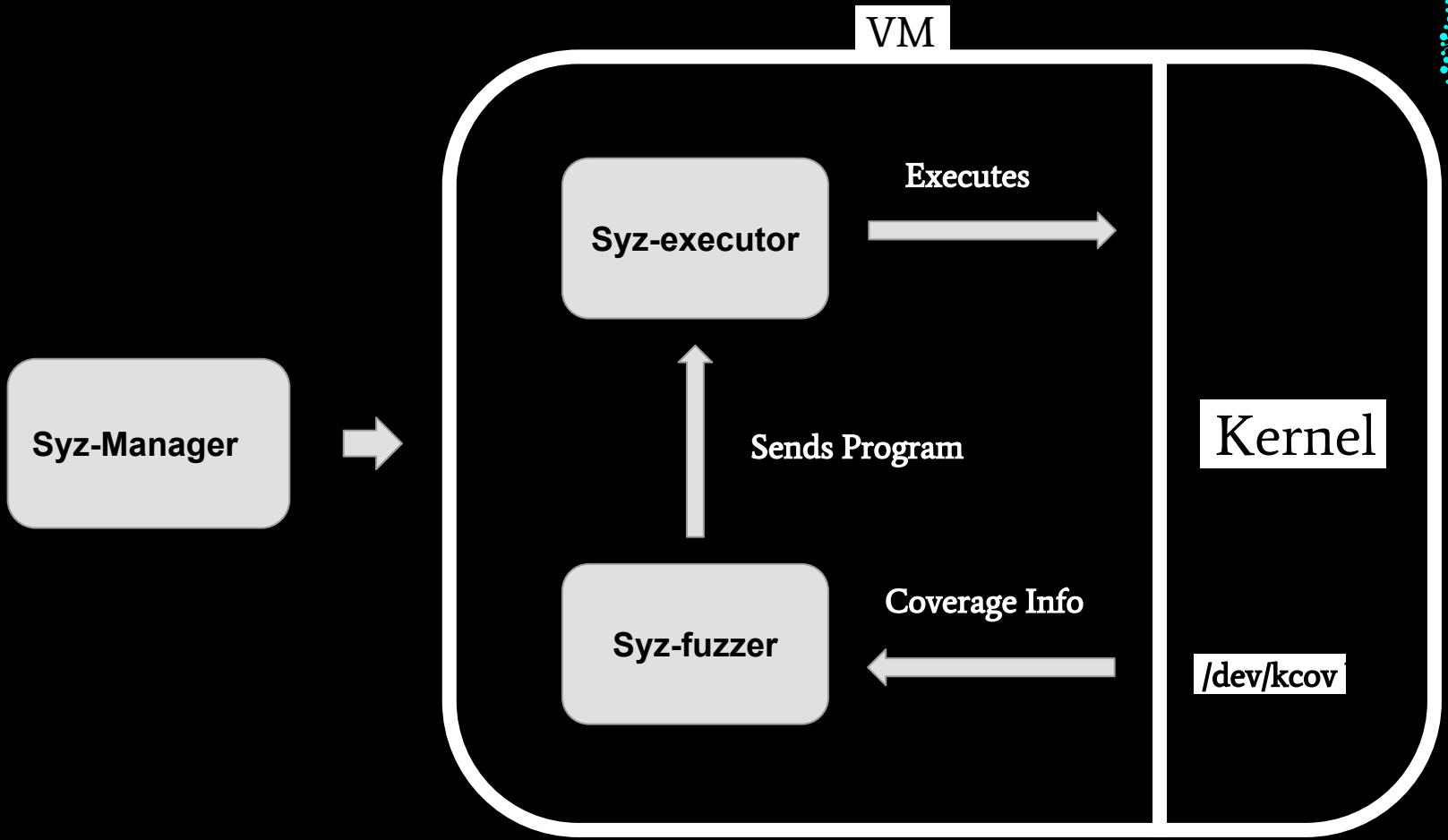
```
open(file filename, flags flags[open_flags], mode flags[open_mode]) fd
open_mode = S_IRUSR, S_IWUSR, S_IXUSR, S_IRGRP, S_IWGRP, S_IXGRP, S_IROTH, S_IWOTH, S_IXOTH
```

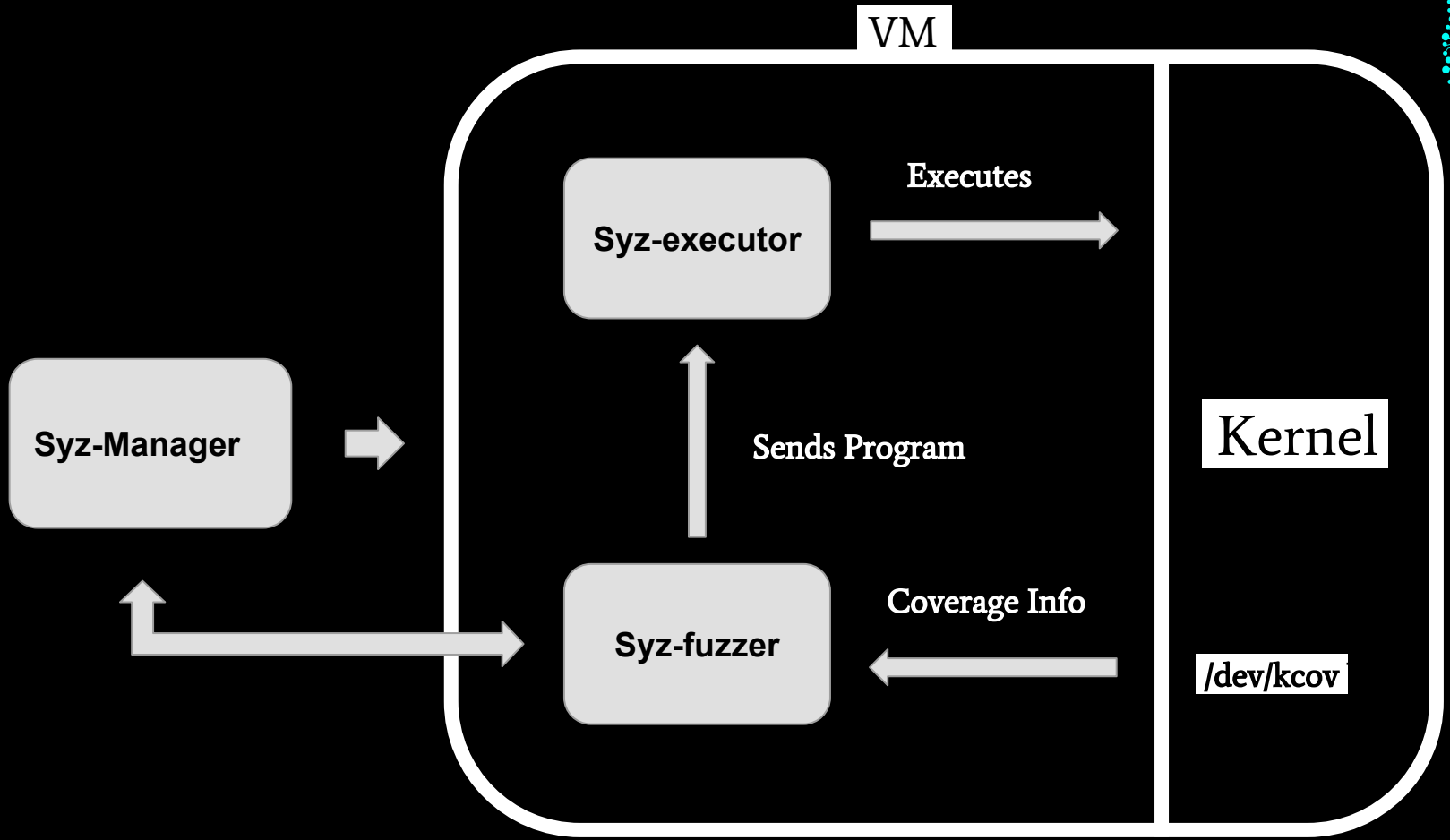
- Uses a pseudo formal grammar for representation

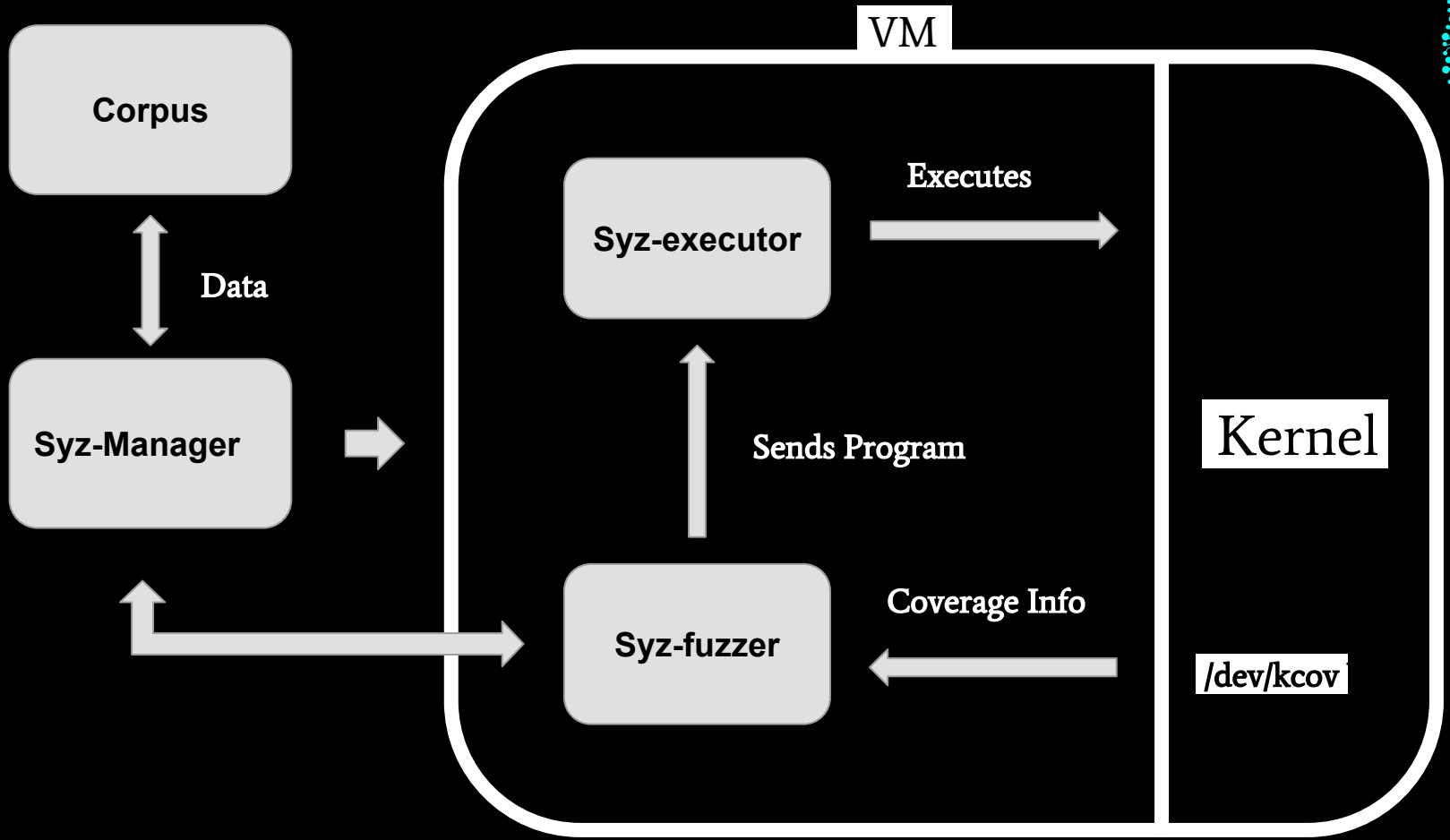
>_ Creating programs (Contd.)

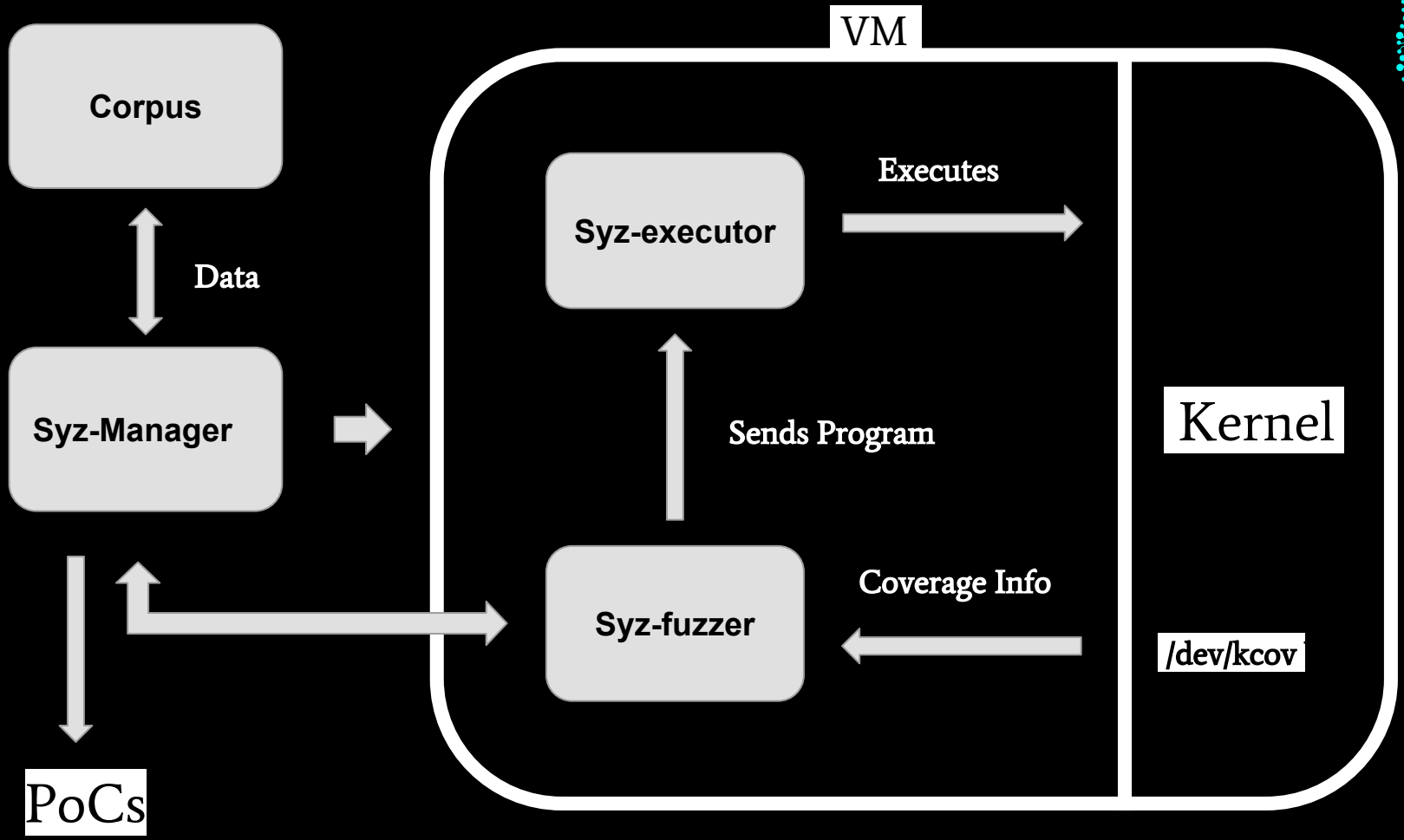


- System calls are executed using Syz-executor
- Mutating inputs
  - Random inputs
  - **bit flips**
- Syz-prog2c - converts Syzkaller representation in C programs

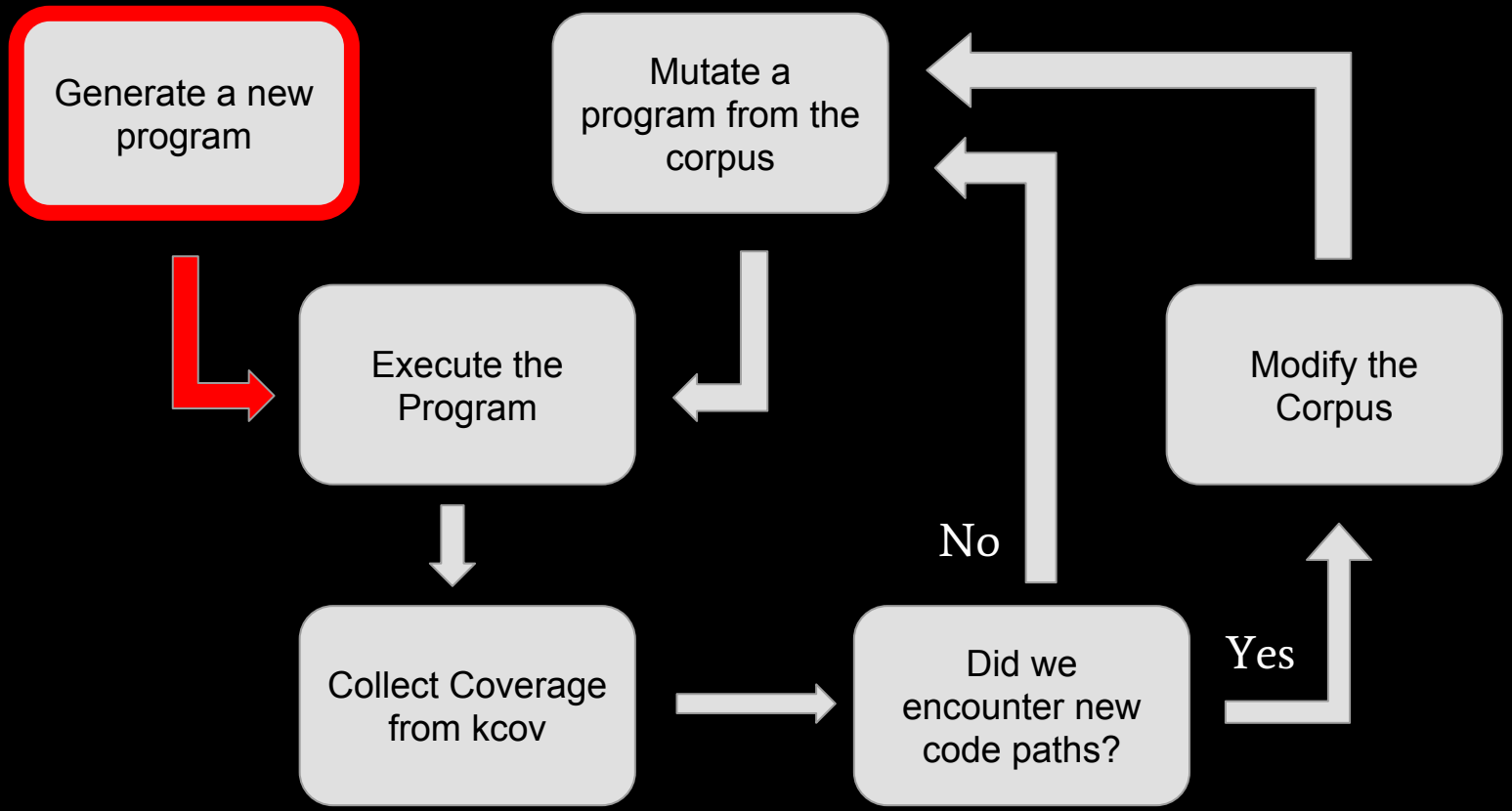

VM
Syz-executor
Executes
Syz-Manager
Sends Program
Kernel
Syz-fuzzer
Coverage Info
/dev/kcov


VM
Syz-executor
Executes
Syz-Manager
Sends Program
Kernel
Syz-fuzzer
Coverage Info
/dev/kcov

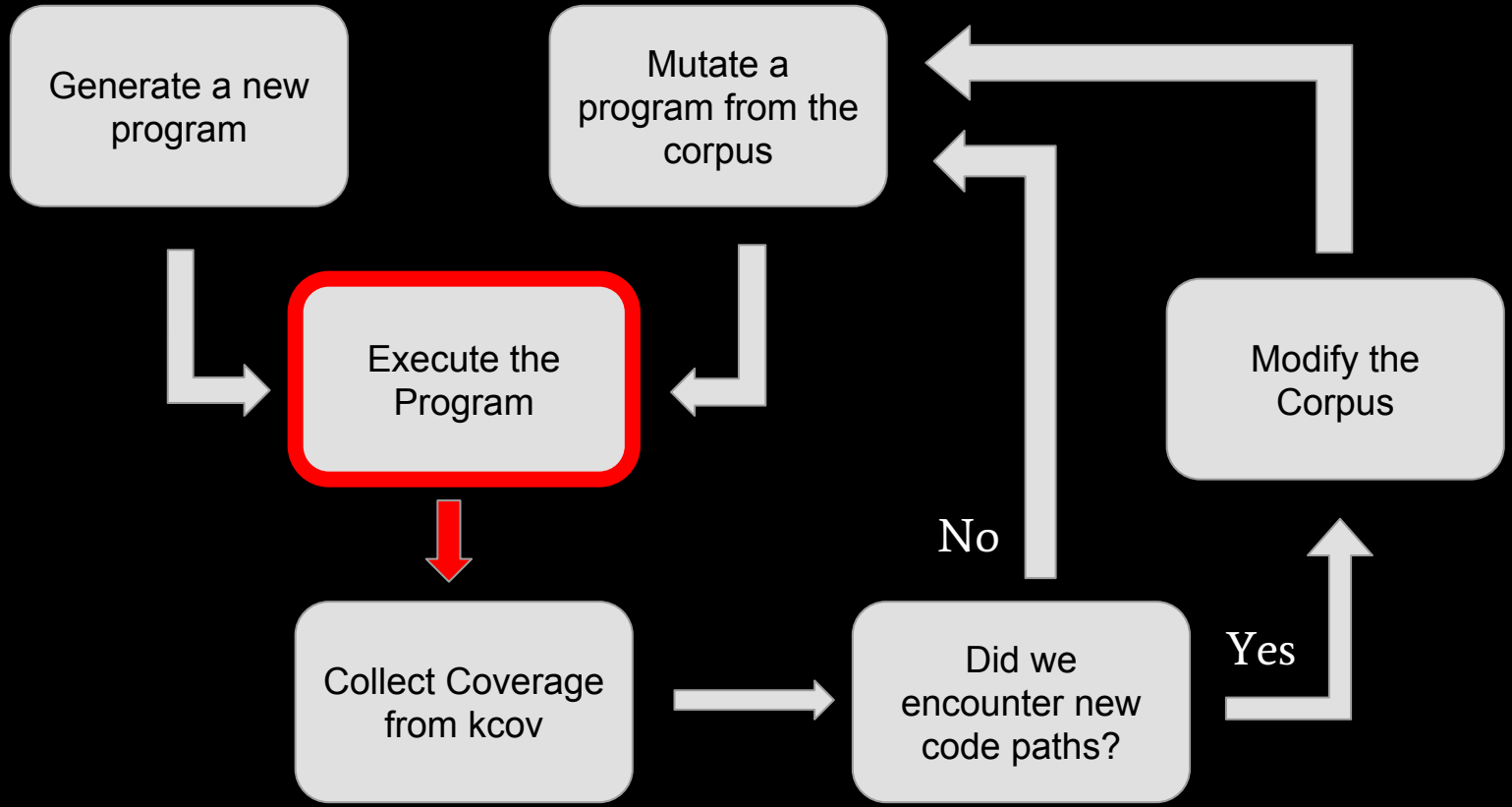

VM
Corpus
Data
Syz-Manager
Syz-executor
Executes
Sends Program
Syz-fuzzer
Coverage Info
Kernel
/dev/kcov


VM
Corpus
Data
Syz-Manager
PoCs
Syz-executor
Executes
Sends Program
Syz-fuzzer
Coverage Info
Kernel
/dev/kcov

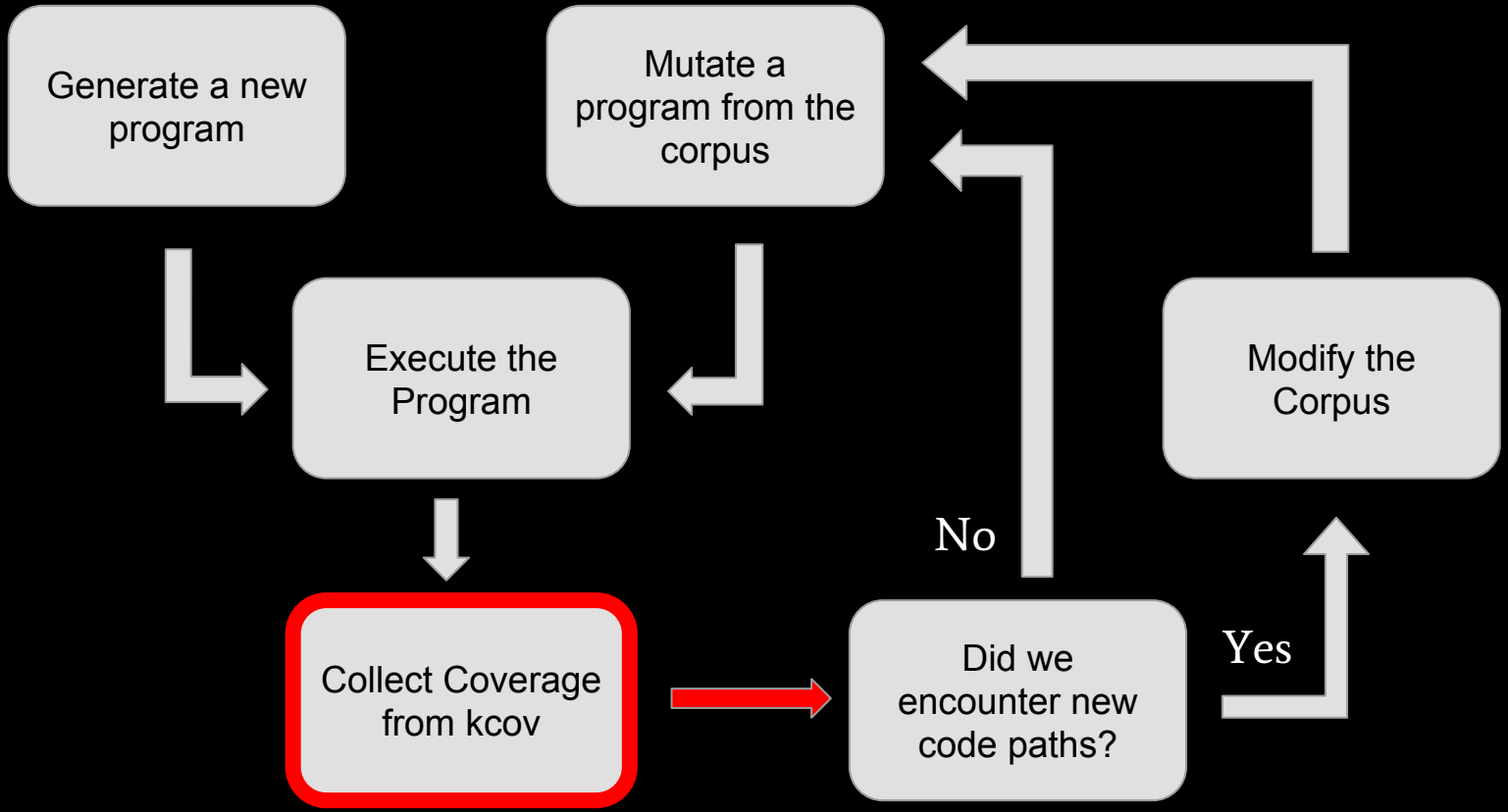
Role of Coverage



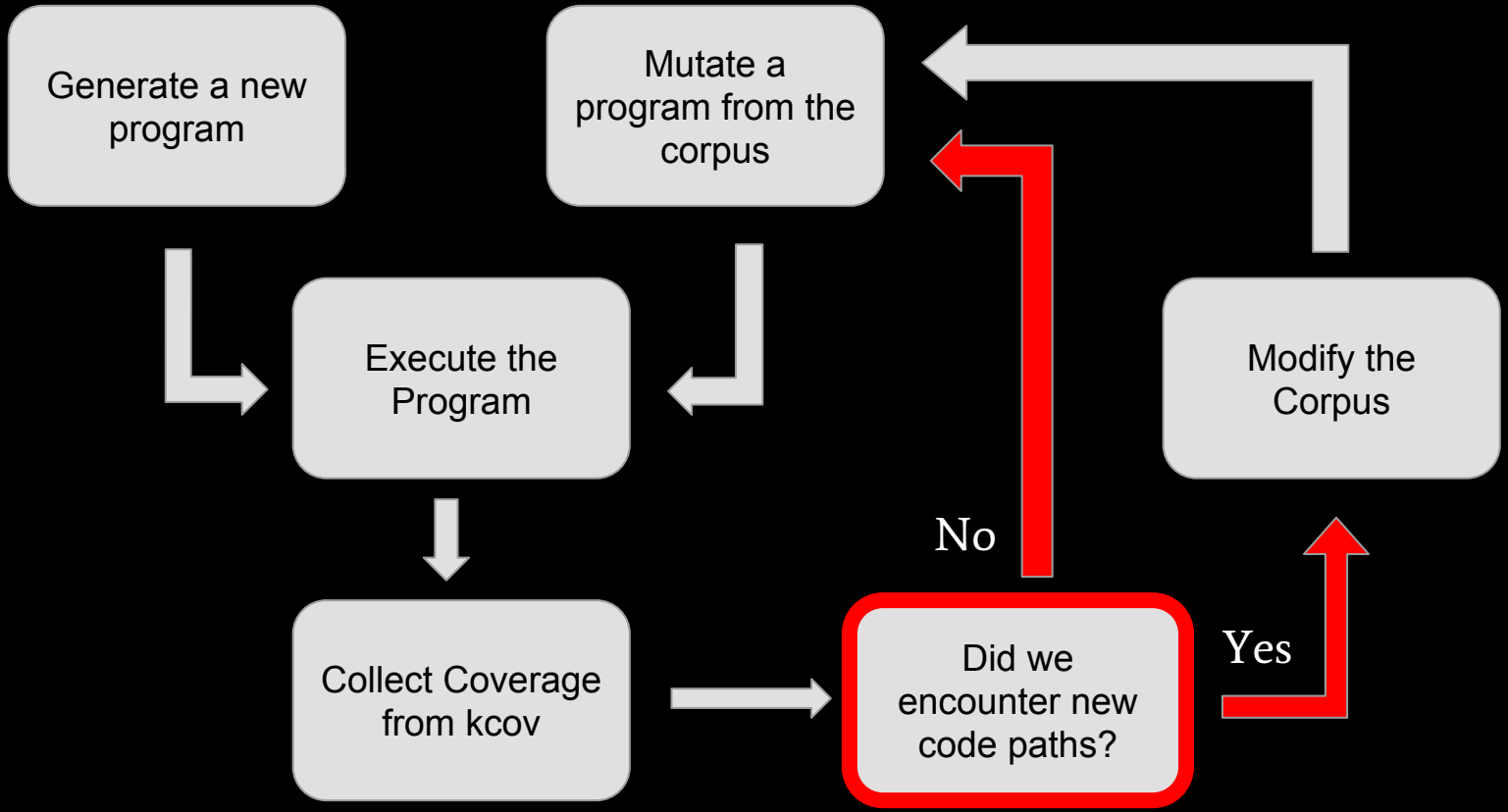
Role of Coverage (Contd)



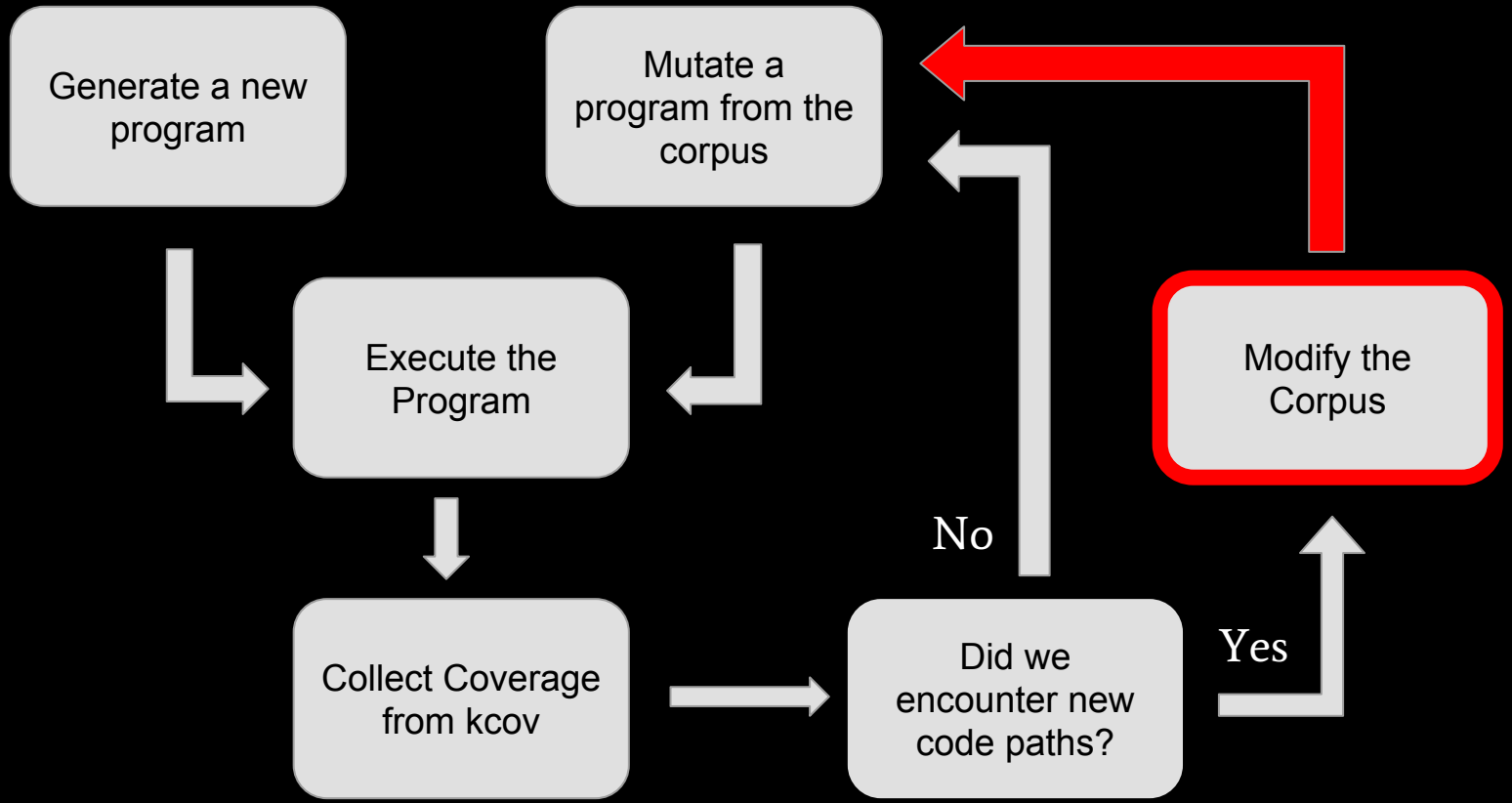
Role of Coverage (Contd)



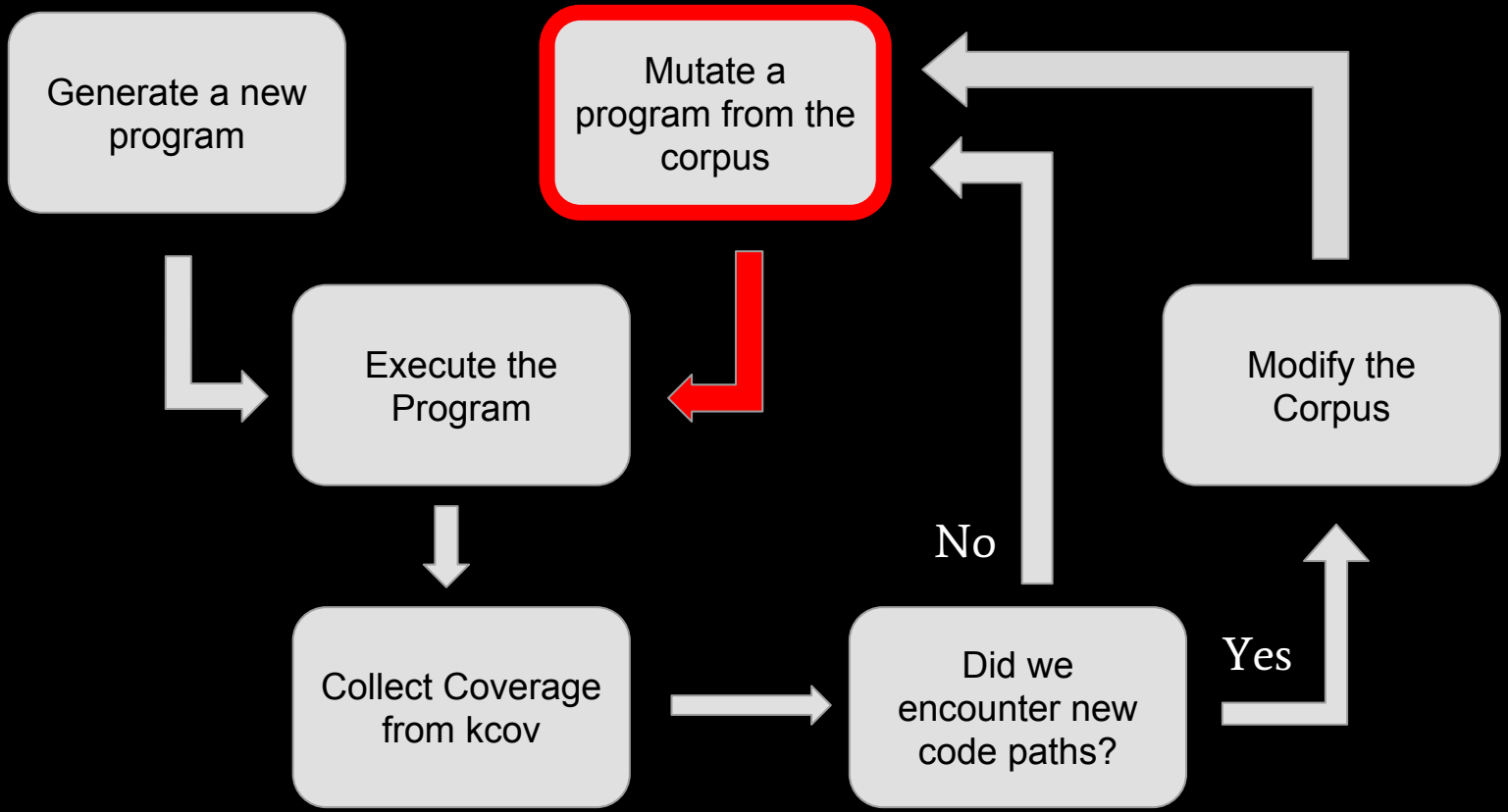
Role of Coverage (Contd)



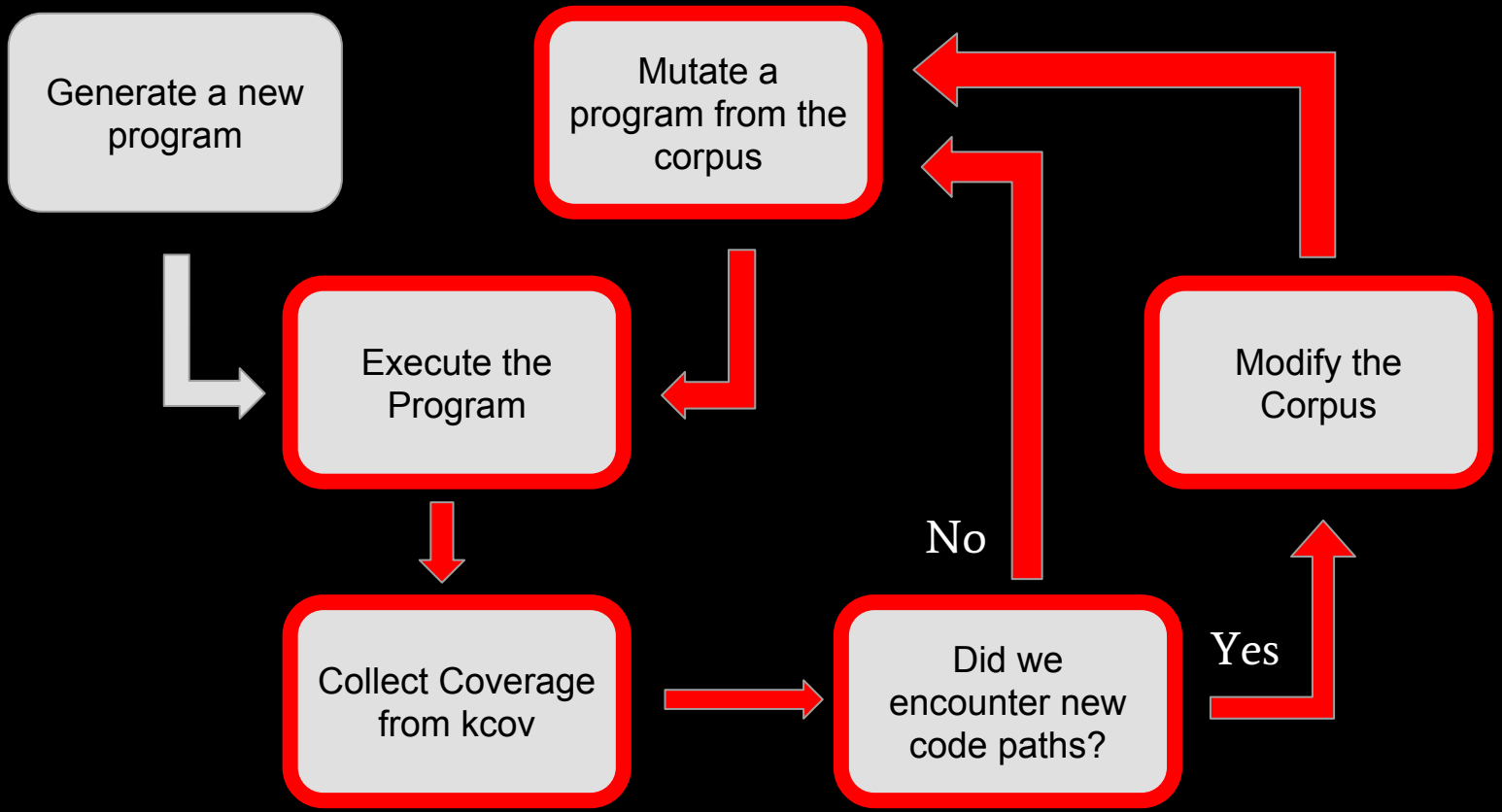
Role of Coverage (Contd)

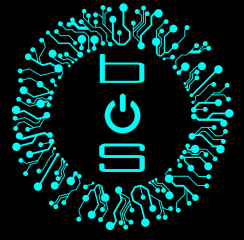


Role of Coverage (Contd)



Role of Coverage (Contd)



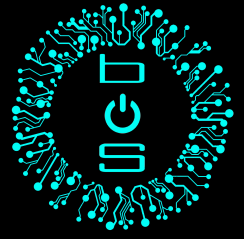


1 weekend = 18 bugs

3 fixed!

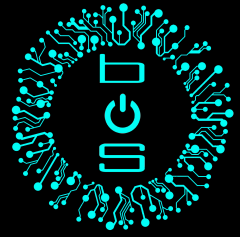

BUGS FOUND
START BUG FIXING MODE

>_ Syzbot



- 24/7 automatic fuzzing
- Google Cloud Engine
- Automatically sends bug reports to a mailing list
- Maintains a dashboard
- Also provides possible reproducers

https://syzkaller.appspot.com/#netbsd



NetBSD

fixed bugs (0)

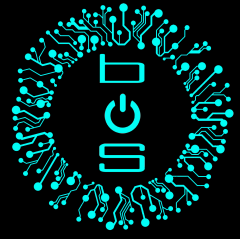
Managers:

Name	Active	Uptime	Build	Kernel	Syzkaller	Corpus	Coverage	Crashes	Execs	Failed Build
netbsd/ci2-netbsd	now	0m	now	dc893675	7a06e792	2715	3682	283	29872	

upstream (8):

Title	Repro	Count	Last	Reported
assert failed: requested_size > 0		2	47m	5h44m
corrupted report		1	6h31m	6h30m
no output from test machine		2	9h50m	11h12m
panic: receive 3	C	3	10h58m	11h51m
panic: event_init: unable to initialize		4	now	11h53m
assert failed: c->c_cpu->cc_lwp == curlwp || c->c_cpu->cc_active != c		2	11h43m	11h58m
ASan bug	C	84	now	12h01m
assert failed: c->c_magic == CALLOUT_MAGIC		437	now	12h01m

https://syzkaller.appspot.com/#netbsd



NetBSD

fixed bugs (0)

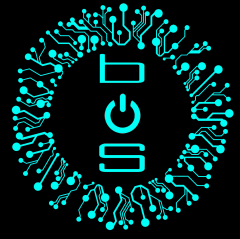
Managers:

Name	Active	Uptime	Build	Kernel	Syzkaller	Corpus	Coverage	Crashes	Execs	Failed Build
netbsd/ci2-netbsd	now	0m	now	dc893675	7a06e792	2715	3682	283	29872	

upstream (8):

Title	Repro	Count	Last	Reported
assert failed: requested_size > 0		2	47m	5h44m
corrupted report		1	6h31m	6h30m
no output from test machine		2	9h50m	11h12m
panic: receive 3	C	3	10h58m	11h51m
panic: event_init: unable to initialize		4	now	11h53m
assert failed: c->c_cpu->cc_lwp == curlwp || c->c_cpu->cc_active != c		2	11h43m	11h58m
ASan bug	C	84	now	12h01m
assert failed: c->c_magic == CALLOUT_MAGIC		437	now	12h01m

https://syzkaller.appspot.com/#netbsd



NetBSD

fixed bugs (0)

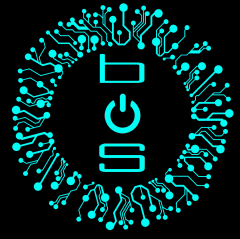
Managers:

Name	Active	Uptime	Build	Kernel	Syzkaller	Corpus	Coverage	Crashes	Execs	Failed Build
netbsd/ci2-netbsd	now	0m	now	dc893675	7a06e792	2715	3682	283	29872	

upstream (8):

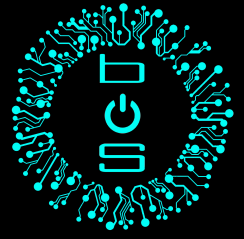
Title	Repro	Count	Last	Reported
assert failed: requested_size > 0		2	47m	5h44m
corrupted report		1	6h31m	6h30m
no output from test machine		2	9h50m	11h12m
panic: receive 3	C	3	10h58m	11h51m
panic: event_init: unable to initialize		4	now	11h53m
assert failed: c->c_cpu->cc_lwp == curlwp || c->c_cpu->cc_active != c		2	11h43m	11h58m
ASan bug	C	84	now	12h01m
assert failed: c->c_magic == CALLOUT_MAGIC		437	now	12h01m

>_ Bugs found - #1 (NetBSD)



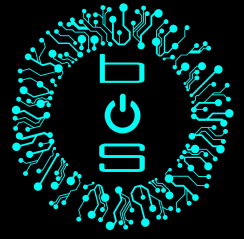
setitimer(2) : *assert failed* **`c->c_magic == CALLOUT_MAGIC`**

- **Kernel Assertion** failed : Causes kernel panic
- Cause was improper initialization
- Fixed by @mlelstv



```
[  36.0260844] panic: kernel diagnostic assertion "c->c_magic ==
CALLOUT_MAGIC" failed: file
"/syzkaller/managers/netbsd/kernel/sys/kern/kern_timeout.c", line 474
[  36.0260844] cpu0: Begin traceback...
[  36.0260844] vpanic() at netbsd:vpanic+0x214
[  36.0260844] _GLOBAL__sub_D_65535_0_cpu_configure() at
netbsd:_GLOBAL__sub_D_65535_0_cpu_configure
[  36.0260844] callout_halt() at netbsd:callout_halt+0x327
[  36.0260844] timer_settime() at netbsd:timer_settime+0x41c
[  36.0260844] dosetitimer() at netbsd:dosetitimer+0x3eb
[  36.0260844] sys___setitimer50() at netbsd:sys___setitimer50+0x127
[  36.0260844] sys_syscall() at netbsd:sys_syscall+0xe2
[  36.0260844] syscall() at netbsd:syscall+0x30e
[  36.0260844] --- syscall (number 0) ---
[  36.0260844] 752e1c63f4ca:
[  36.0260844] cpu0: End traceback...
[  36.0260844] rebooting...
```

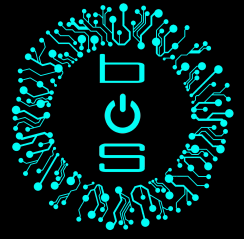
>_ Bugs found - #2 (NetBSD)



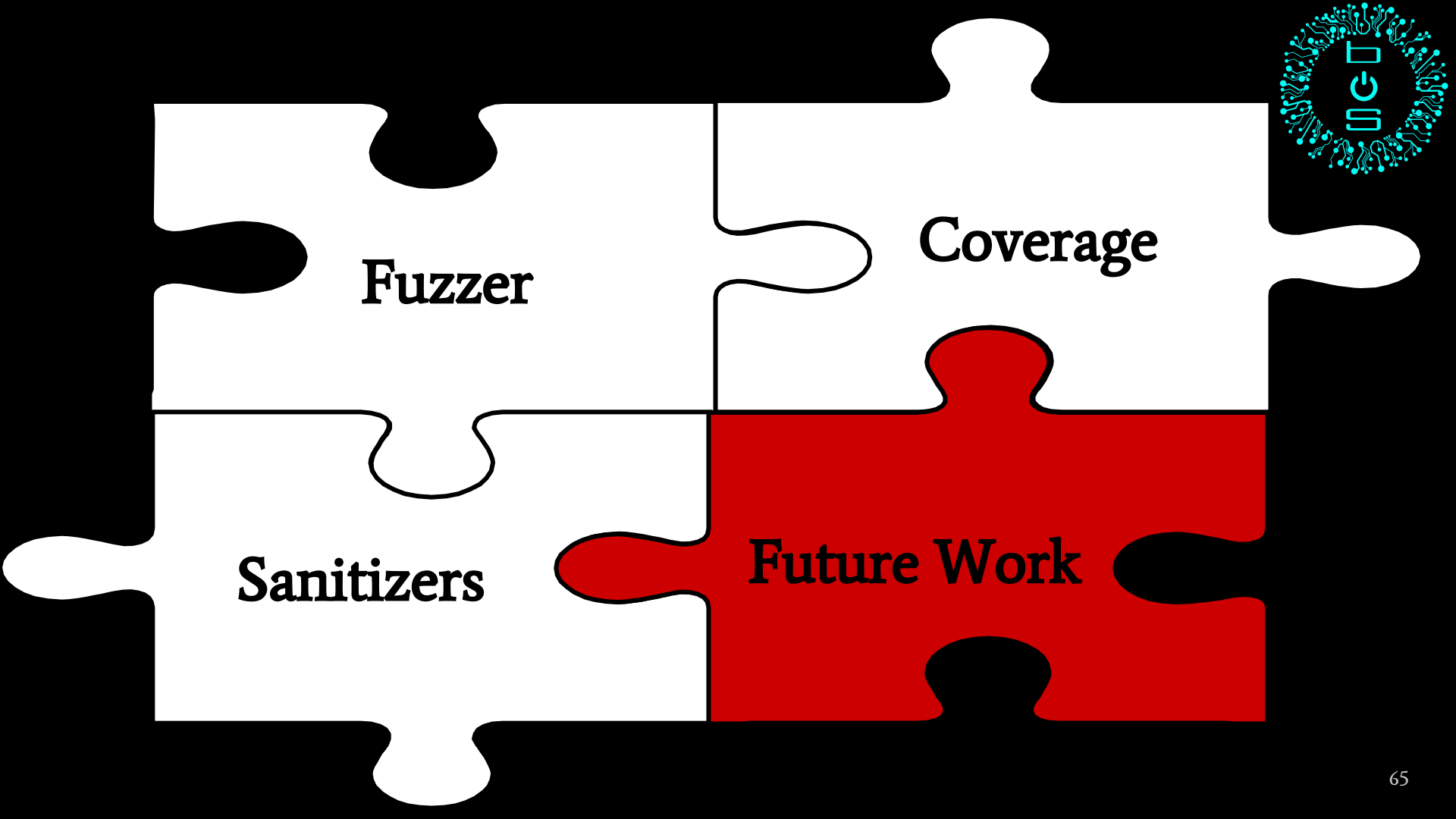
Connect(2) and Send(2) : *Out of Bounds read*

- Protocols such as RIP, CAN, SCTP - had a missing length check in their Connect and Send function
- Detected with Kernel Address Sanitizer
- Severity : Medium
- Fixed upstream by @maxv

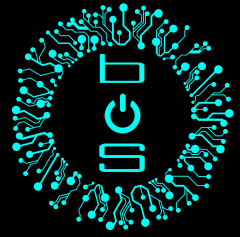
>_ Network Interface



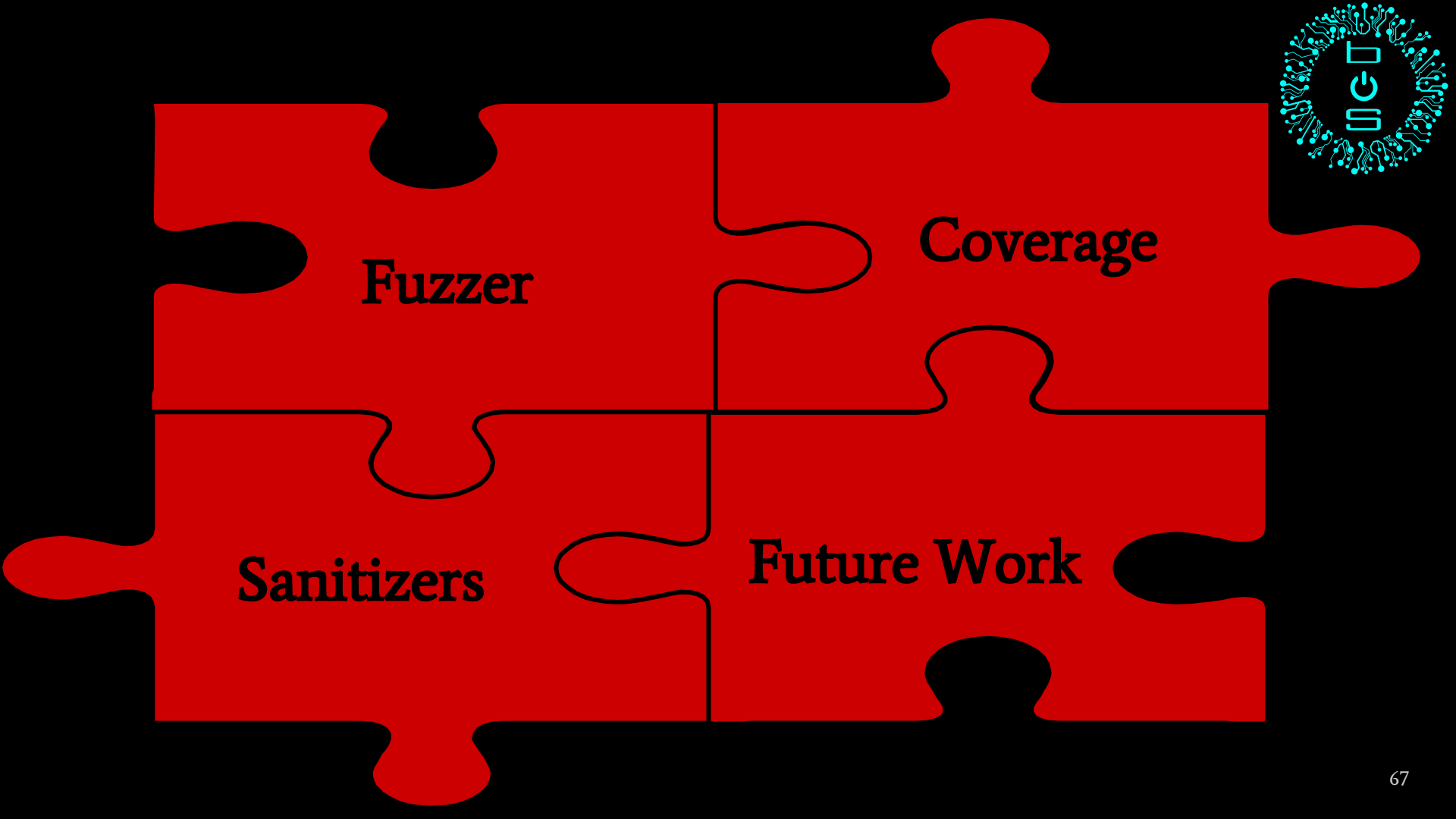
- Target : **Network Stack**
- Implemented in Linux
- Interface : **TAP/TUN** interface
- Receive and Send **crafted packets**
- Fake syscalls implemented
  - Pass packet through the virtual interface
  - externally receive a packet back


Fuzzer
Coverage
Sanitizers
Future Work

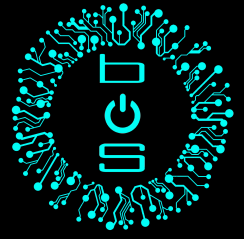
>_ Future Work



- Add better support for Syzkaller
- Add support for other subsystems
  - Filesystem
- Better reproducers
- Add support for multiple architectures
- Port Kernel Memory Sanitizer
- Port other fuzzers for NetBSD
  - AFL - Triforce

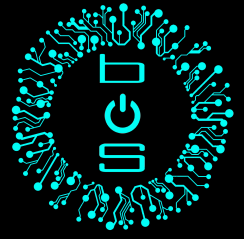

Fuzzer
Coverage
Sanitizers
Future Work

>_ Quick Rewind

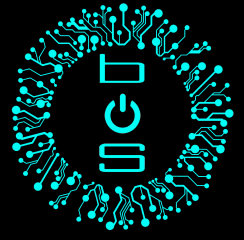


- Setup
  - Syz-Manager
  - Syz-Fuzzer
- Reports
  - Console logs
  - Report Symbolization
- Restricted Kernel APIs
  - System call layer
  - Additional System Calls
- Proper inputs
  - Scraping codebase
  - Pseudo grammar
  - Mutation
- Reproducers
  - Syz-prog2c
- Indetermination
  - Coverage

>_ Credits



- NetBSD Foundation
  - Kamil Rytarowski (@ryoshu)
  - Maxime Villard (@maxv)
- Google
  - Dmitry Vyukov (@dyukov)



Thank You

Any Questions?