Your NVMe had Been Syz'ed

Alon Zahavi – Insomni'hack 2024

whoami

Alon Zahavi

Security Researcher @CyberArk

Agenda

1. Syzkaller Overview
2. NVMe-oF/TCP Overview
3. Adding NVMe to syzkaller

0

Sneak Peek

1

Syzkaller Overview

Syzkaller Overview

Coverage guided kernel fuzzer

Inputs are description-based

Uses KCOV for kernel coverage (in Linux)

Syzkaller Overview

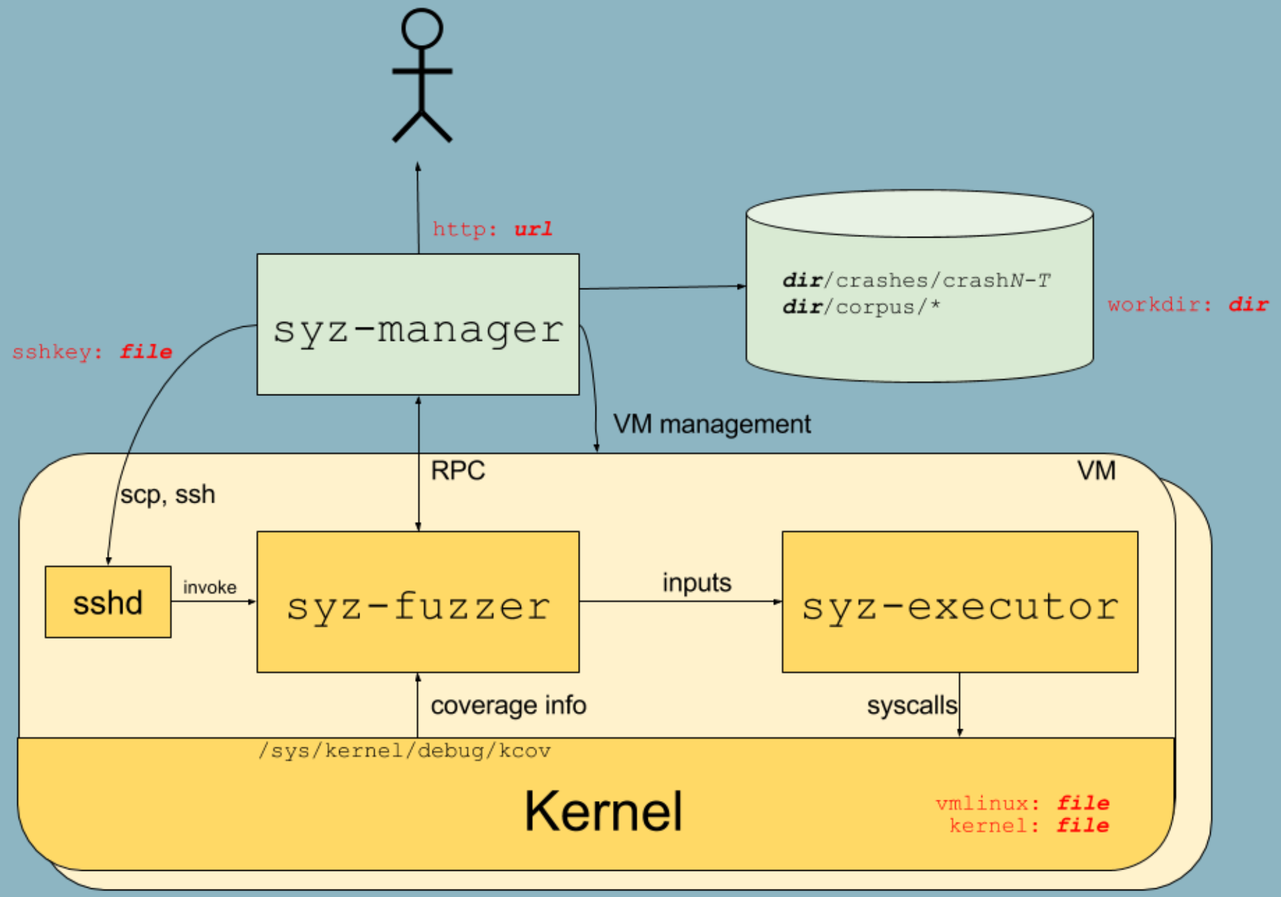
internals

Syzkaller Overview

internals

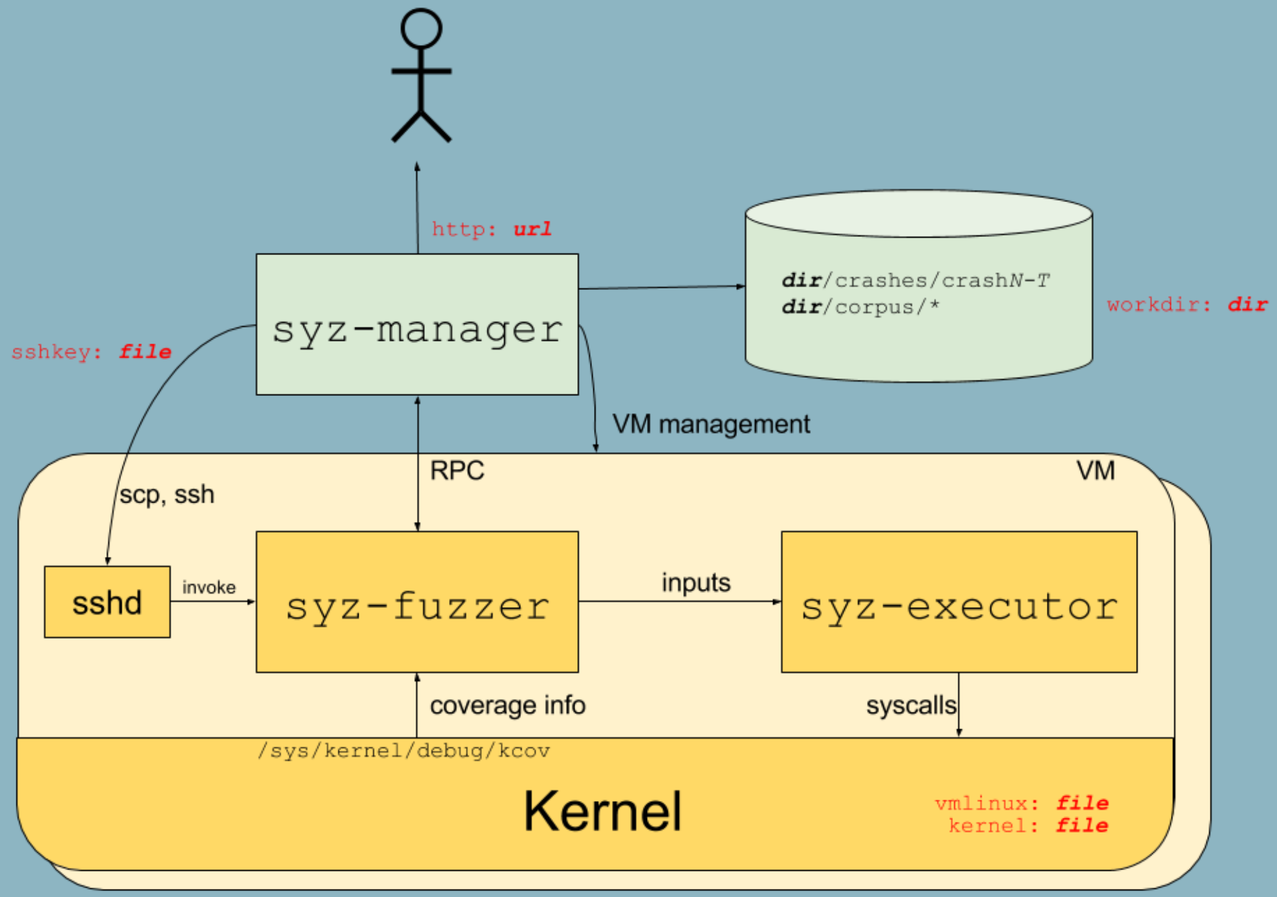
Syzkaller Overview

internals



Syzkaller Overview

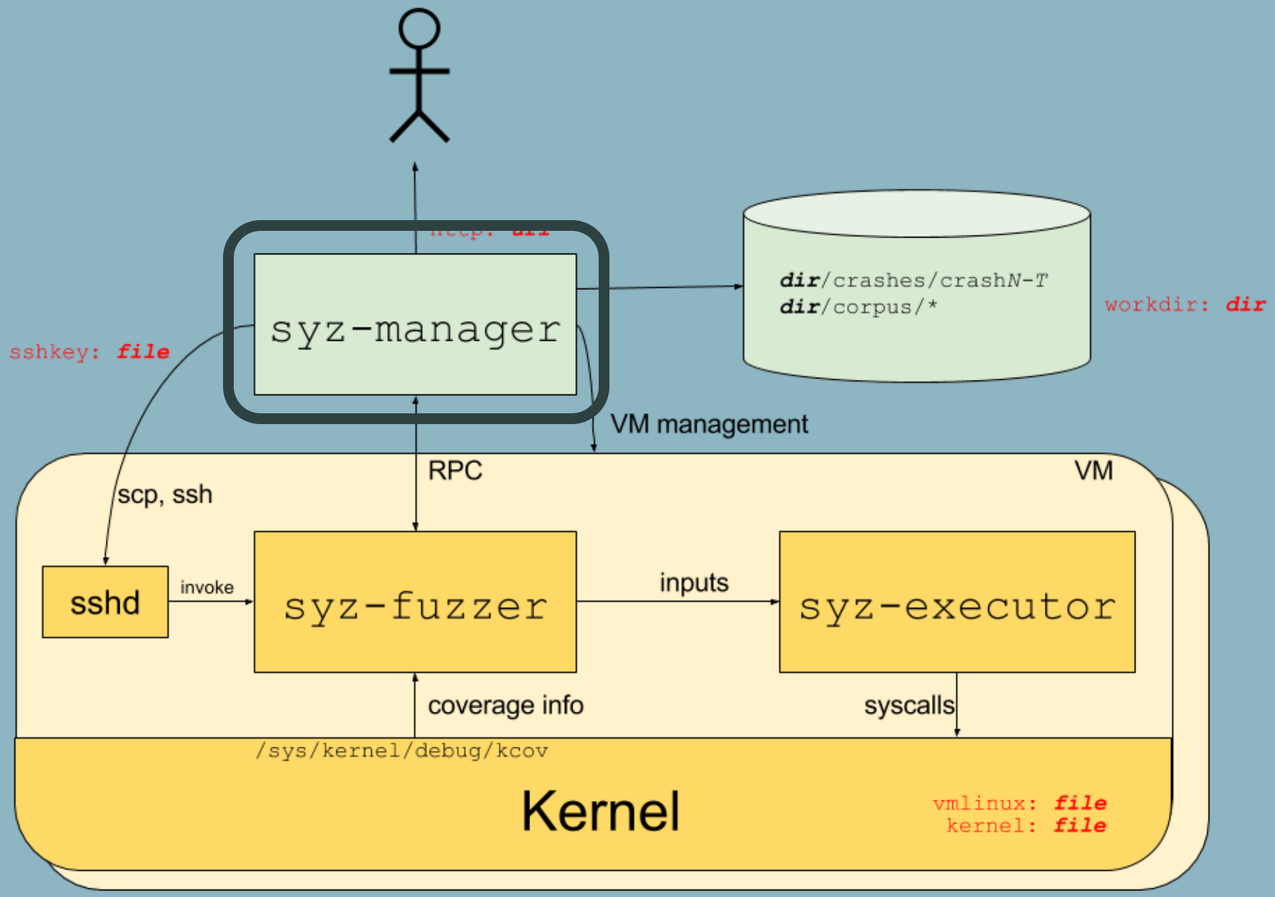
internals



Syzkaller Overview

internals

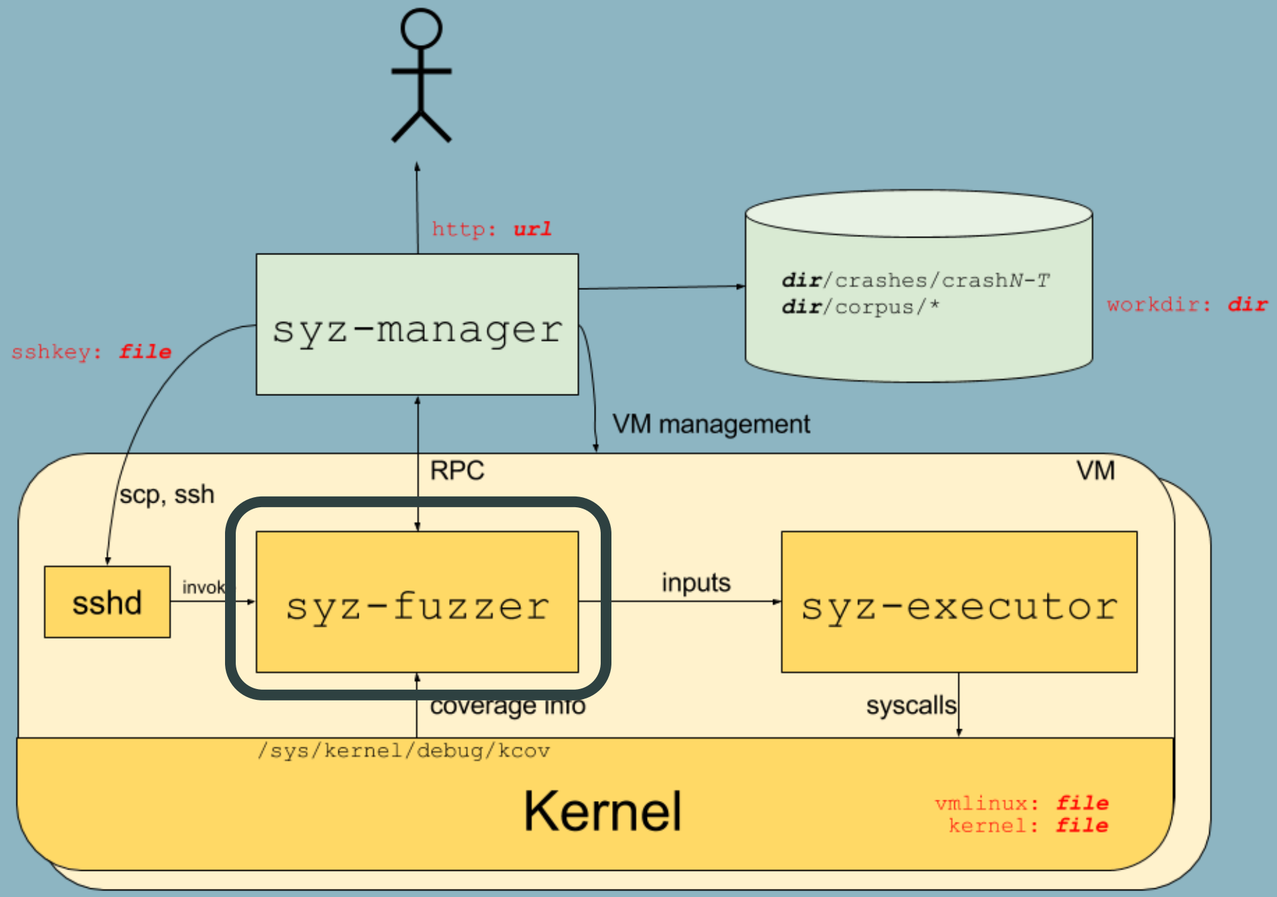
- **syz-manager** is the main part that manages the whole syzkaller fuzzing process



Syzkaller Overview

internals

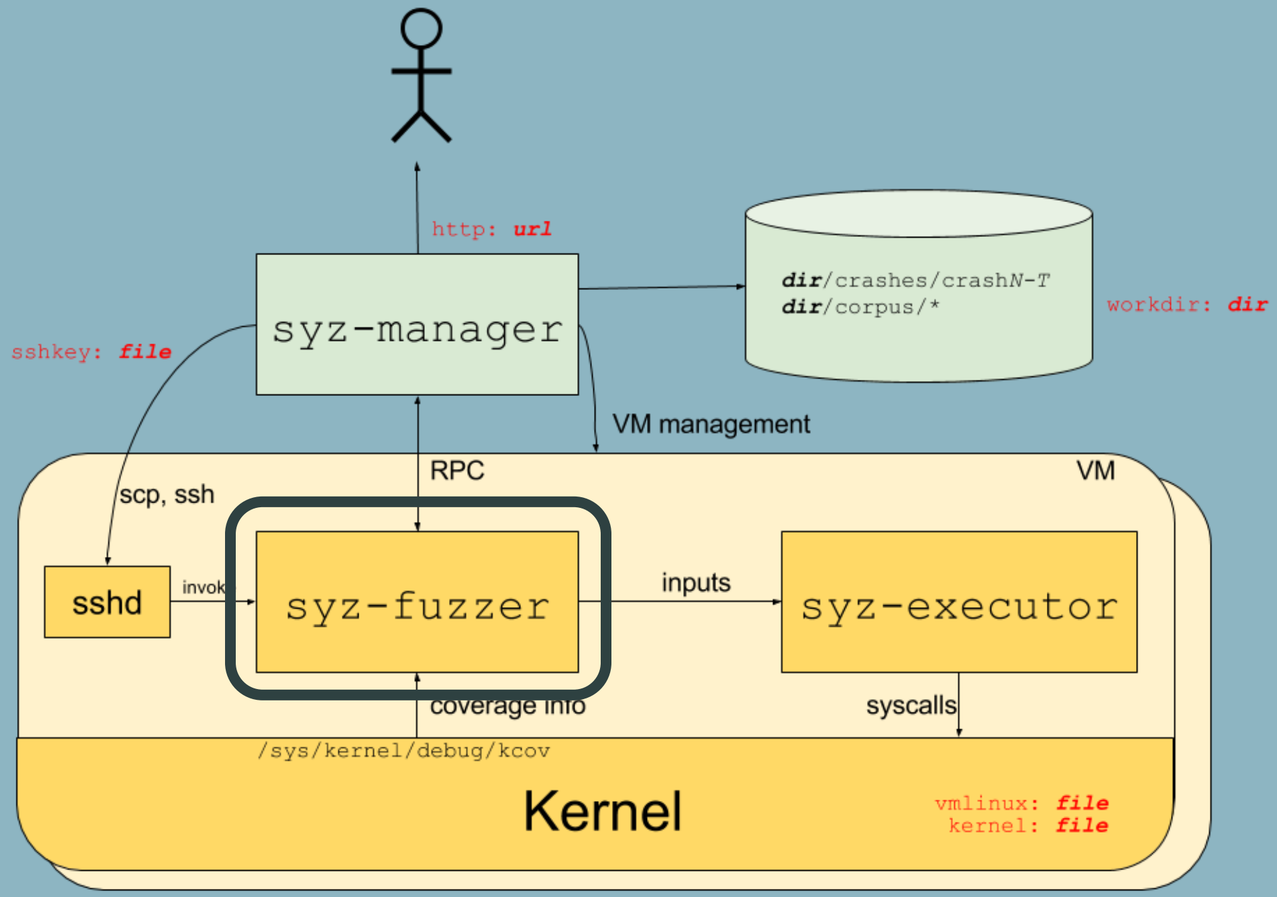
- **syz-manager** is the main part that manages the whole syzkaller fuzzing process
- **syz-manager** starts VM instances and starts **syz-fuzzer** processes in those QEMU VMs.



Syzkaller Overview

internals

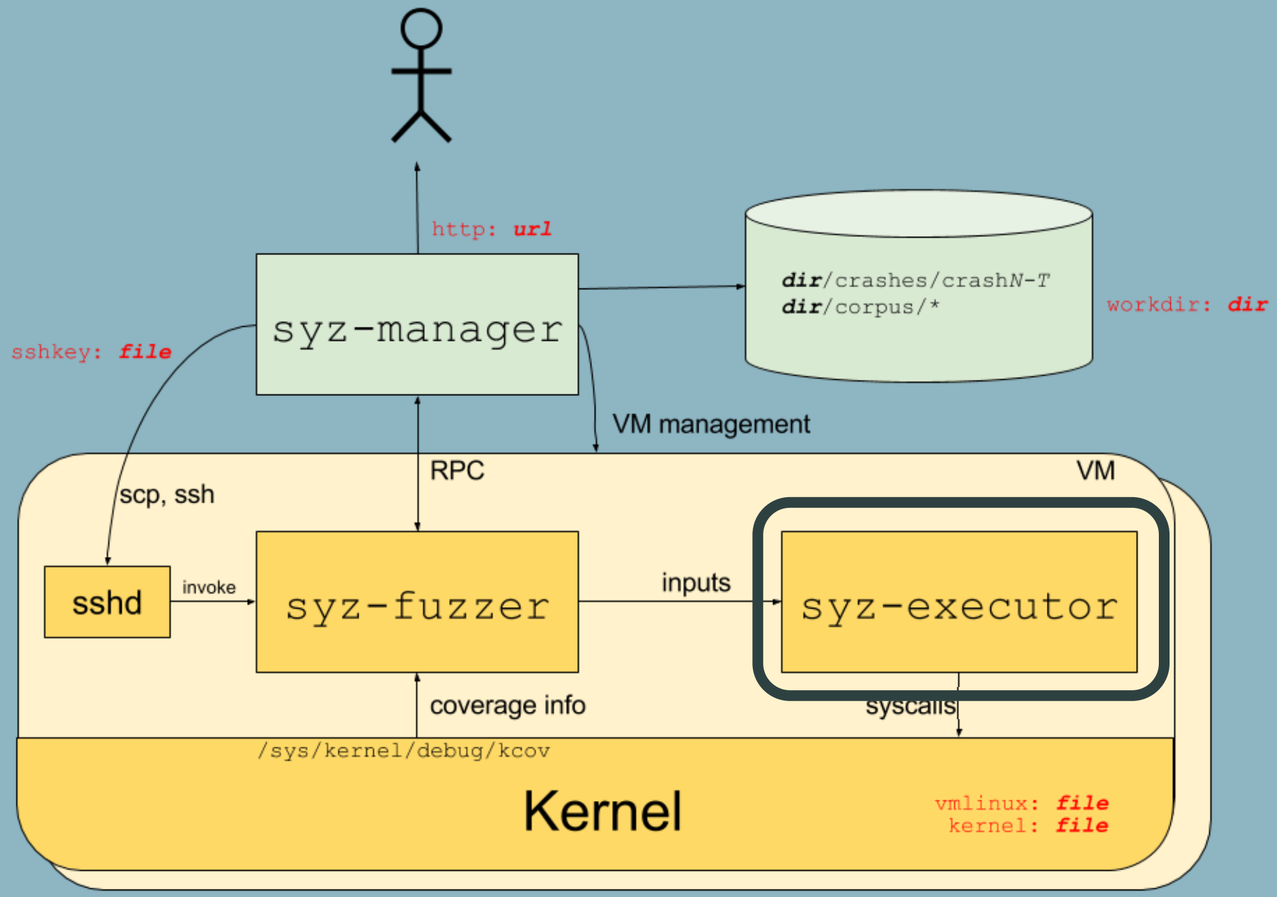
- **syz-manager** is the main part that manages the whole syzkaller fuzzing process
- **syz-manager** starts VM instances and starts **syz-fuzzer** processes in those QEMU VMs.
- **syz-manager** monitors the **syz-fuzzer** processes



Syzkaller Overview

internals

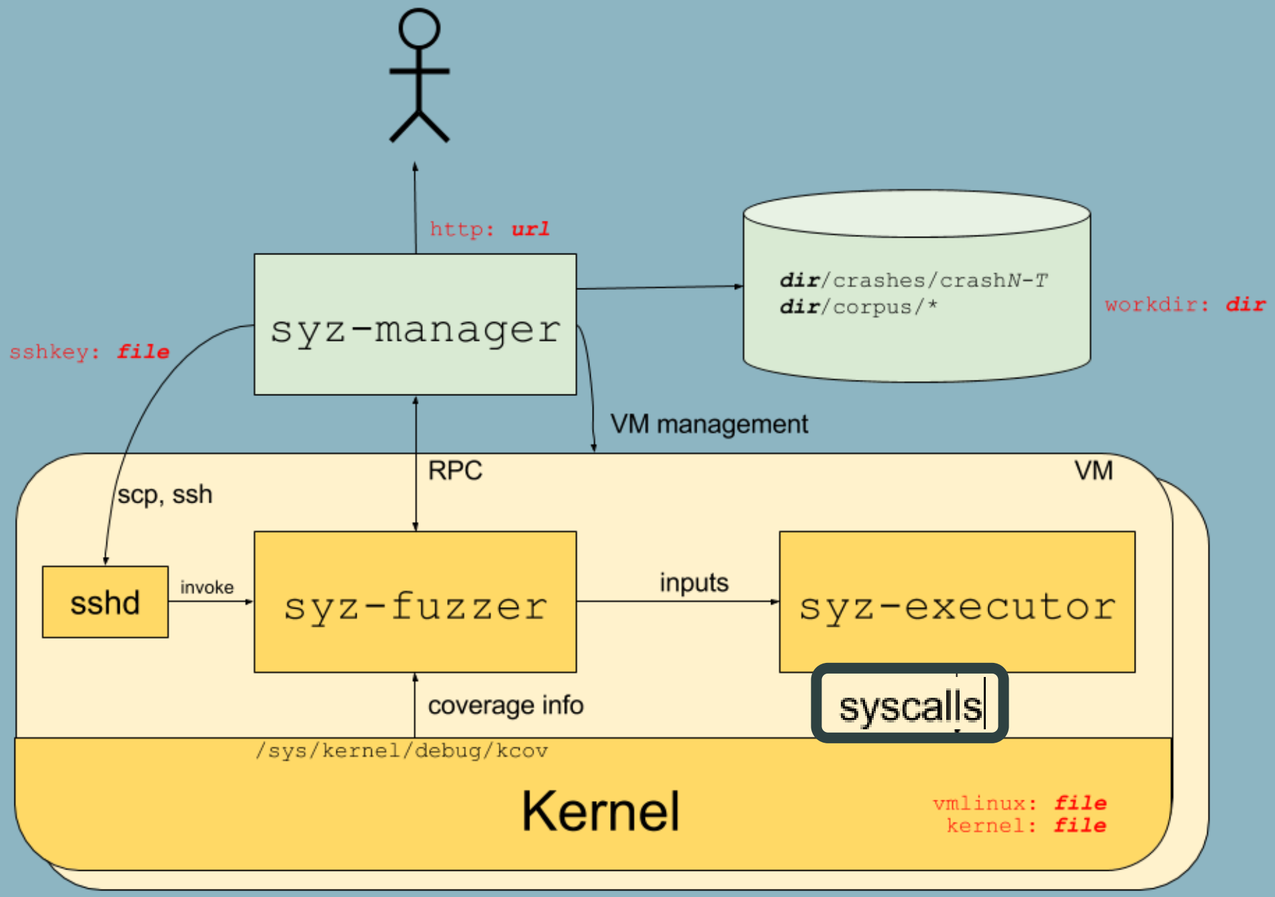
- **syz-manager** is the main part that manages the whole syzkaller fuzzing process
- **syz-manager** starts VM instances and starts **syz-fuzzer** processes in those QEMU VMs.
- **syz-manager** monitors the **syz-fuzzer** processes
- **syz-fuzzer** starts **syz-executor** processes inside the VM



Syzkaller Overview

internals

- **syz-manager** is the main part that manages the whole syzkaller fuzzing process
- **syz-manager** starts VM instances and starts **syz-fuzzer** processes in those QEMU VMs.
- **syz-manager** monitors the **syz-fuzzer** processes
- **syz-fuzzer** starts **syz-executor** processes inside the VM
- **syz-executor** executes a single input program and sends the results back to **syz-fuzzer**



Syzkaller Overview

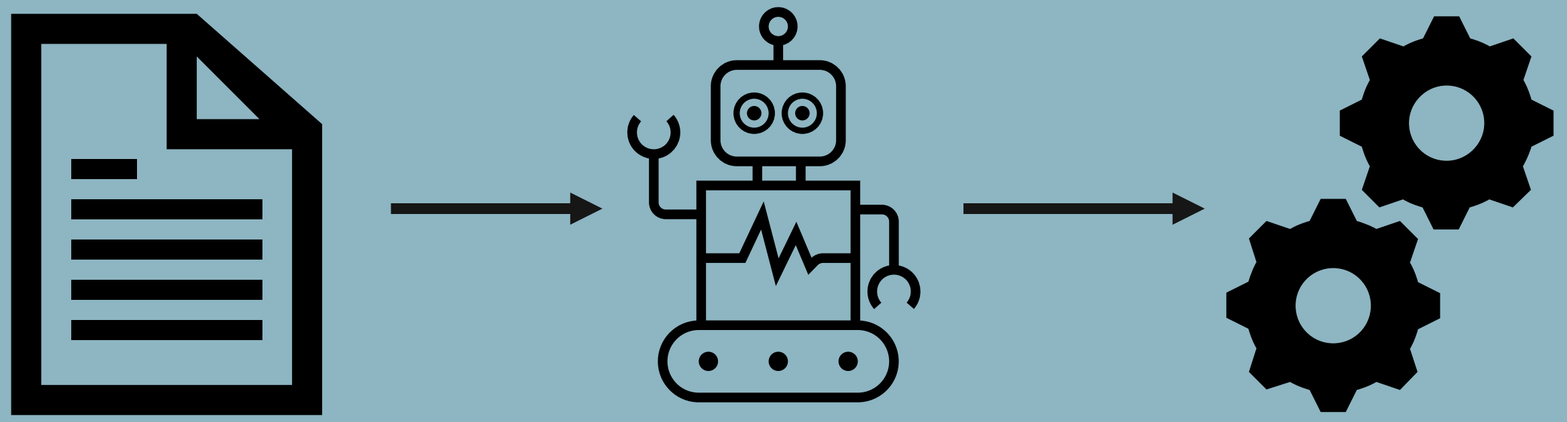
syzlang

Syzkaller Overview

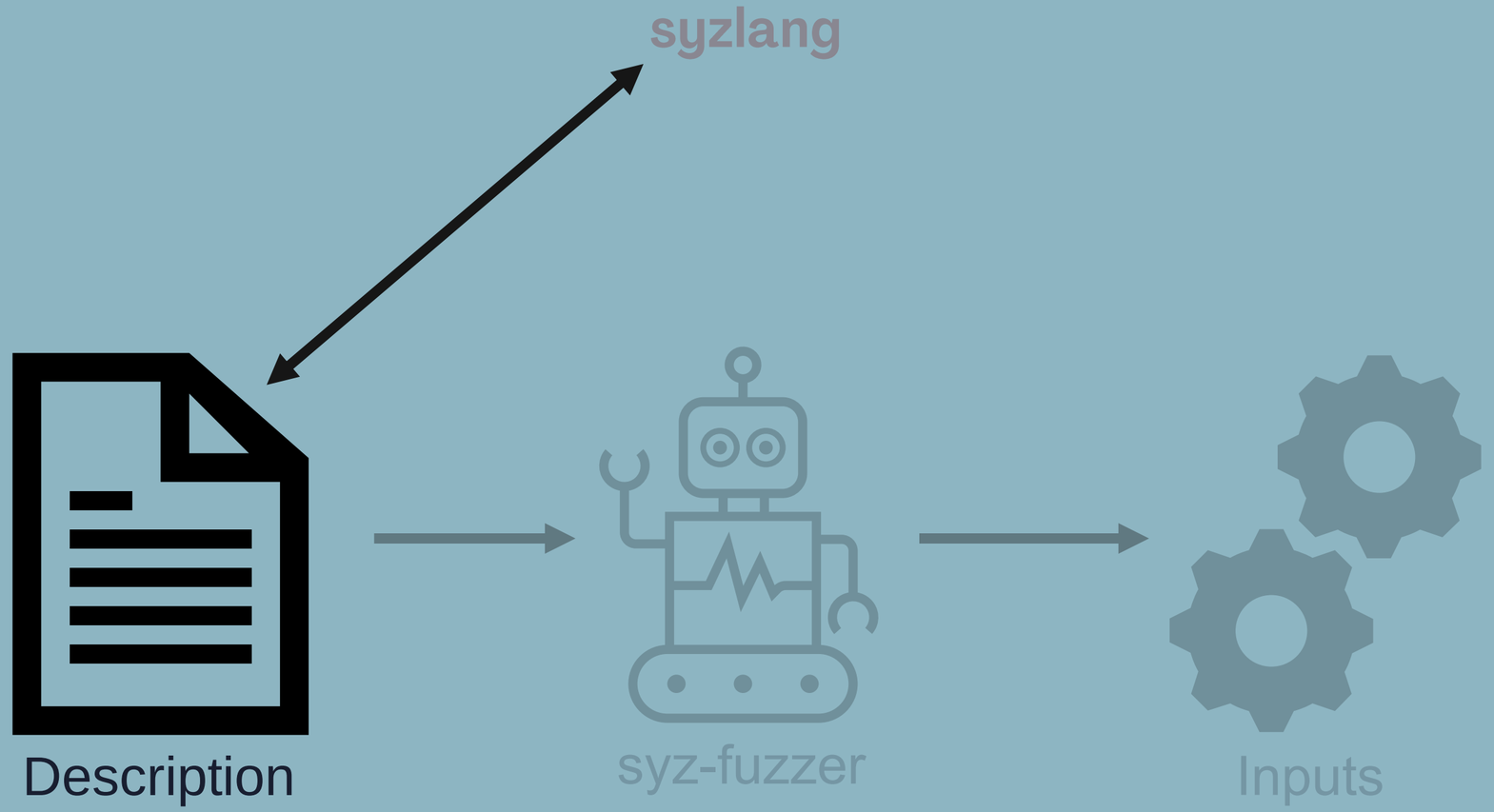
syzlang

Syzkaller Overview

syzlang



Syzkaller Overview



Syzkaller Overview

syzlang - syscalls

Syzkaller Overview

syzlang - syscalls

```
SYSCALL_DEFINE3(read, unsigned int, fd, char __user *, buf, size_t, count)
```

Syzkaller Overview

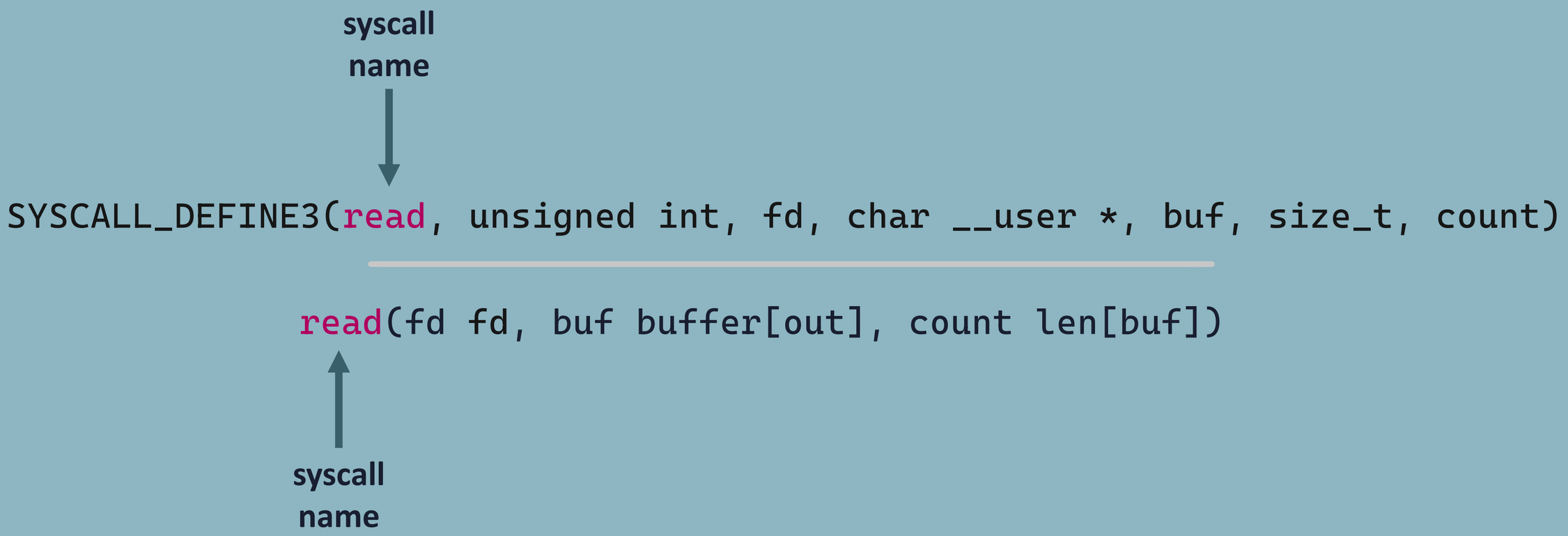
syzlang - syscalls

```
SYSCALL_DEFINE3(read, unsigned int, fd, char __user *, buf, size_t, count)
```

```
read(fd fd, buf buffer[out], count len[buf])
```

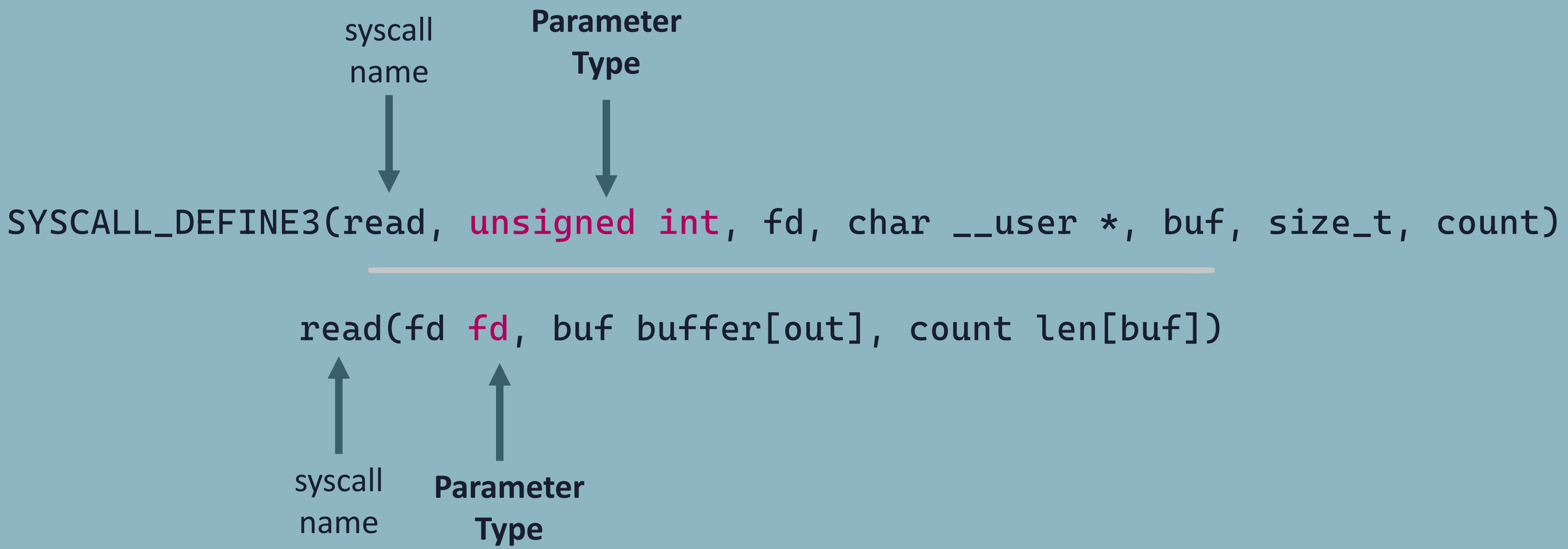
Syzkaller Overview

syzlang - syscalls



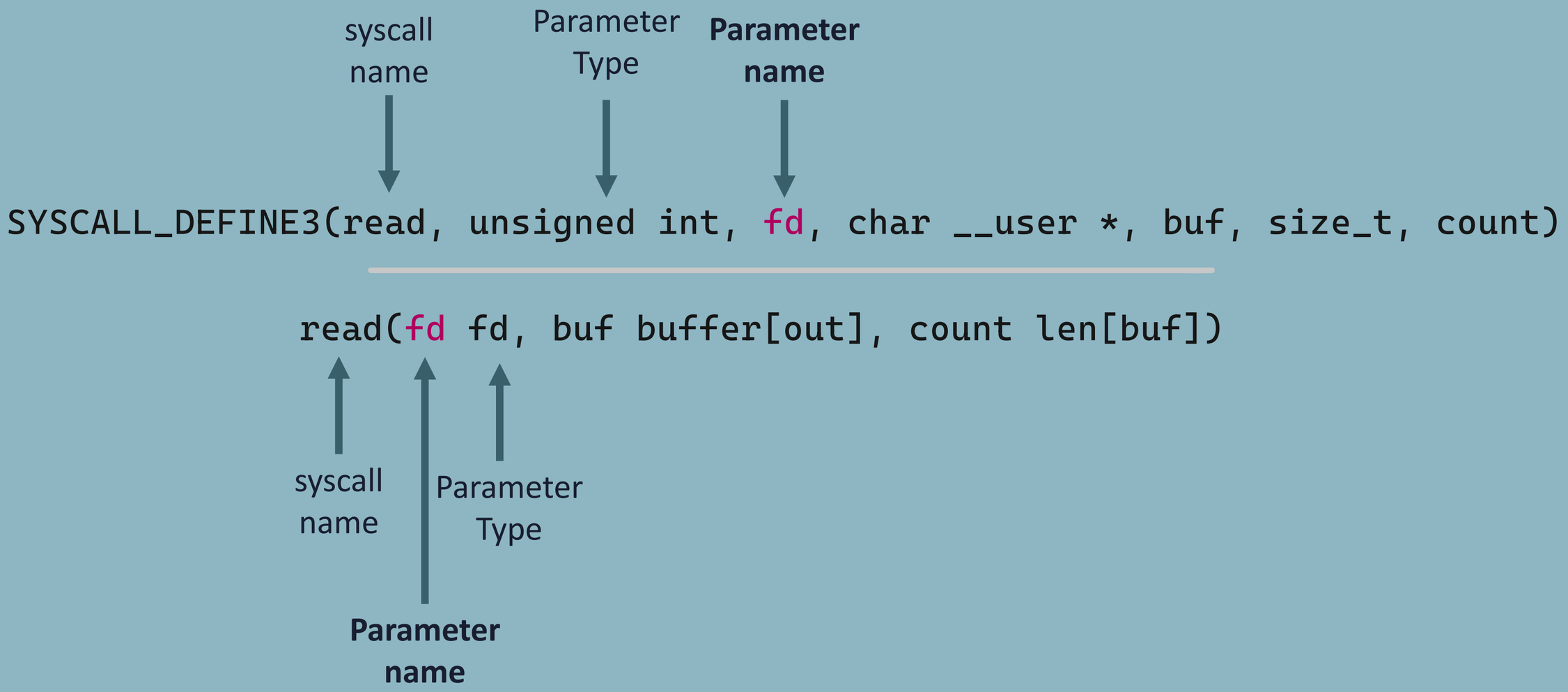
Syzkaller Overview

syzlang - syscalls



Syzkaller Overview

syzlang - syscalls



Syzkaller Overview

syzlang - syscalls

```
read$myfile(fd fd_myfile, data buffer[out], len len[data])
```

Syzkaller Overview

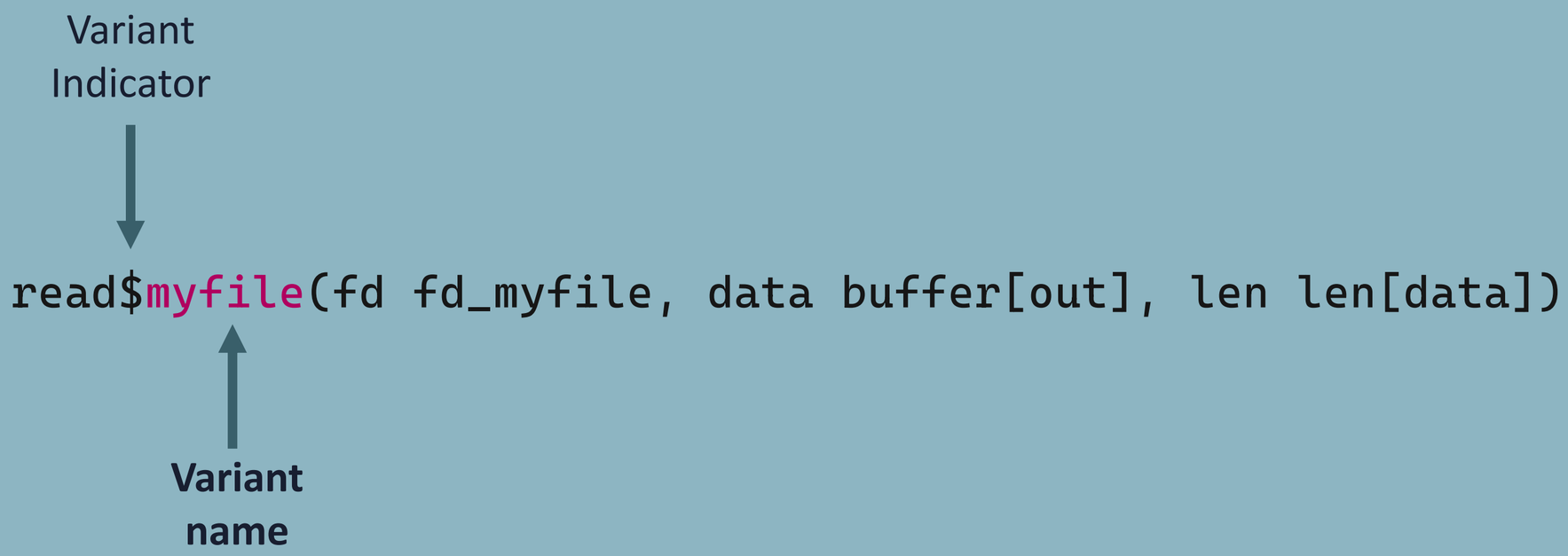
syzlang - syscalls

Variant Indicator

```
read$myfile(fd fd_myfile, data buffer[out], len len[data])
```

Syzkaller Overview

syzlang - syscalls

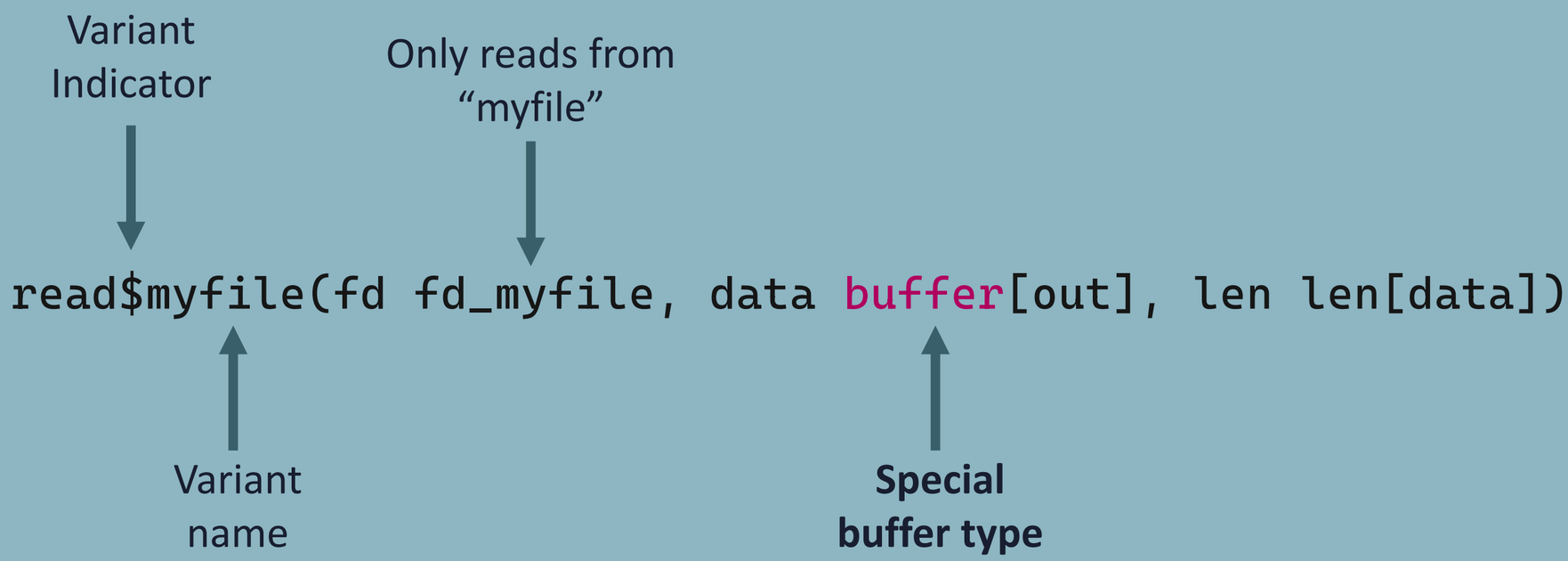


Syzkaller Overview

syzlang - syscalls

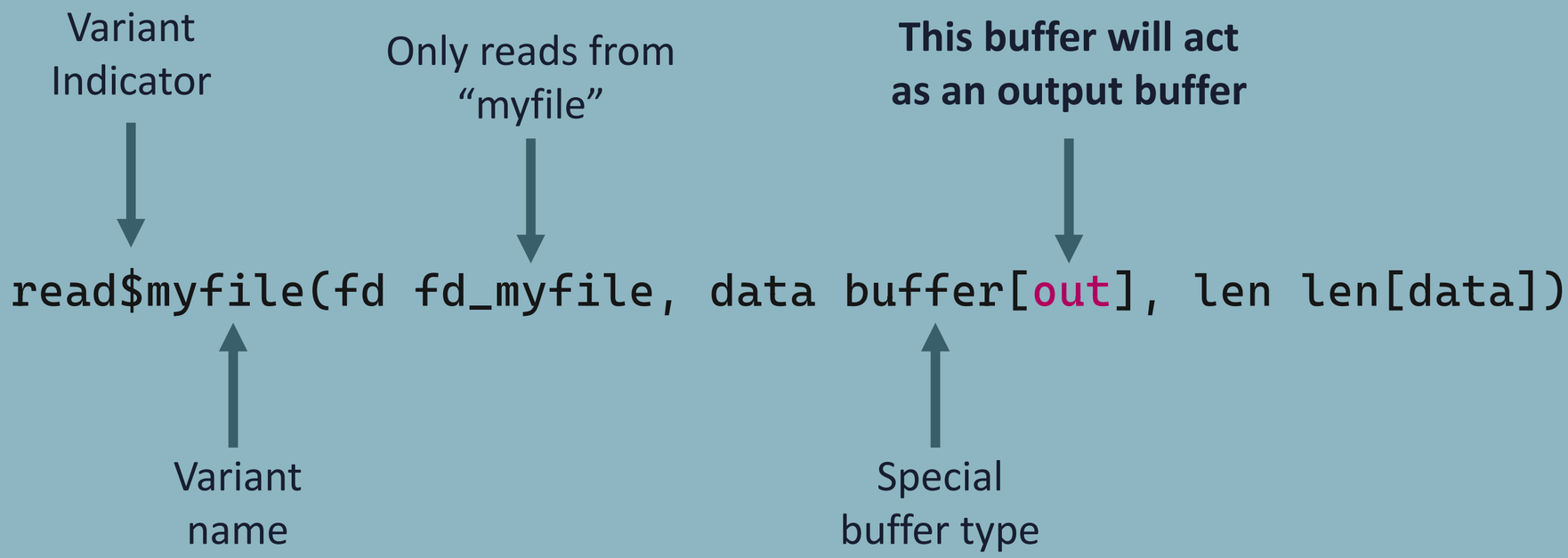
Syzkaller Overview

syzlang - syscalls



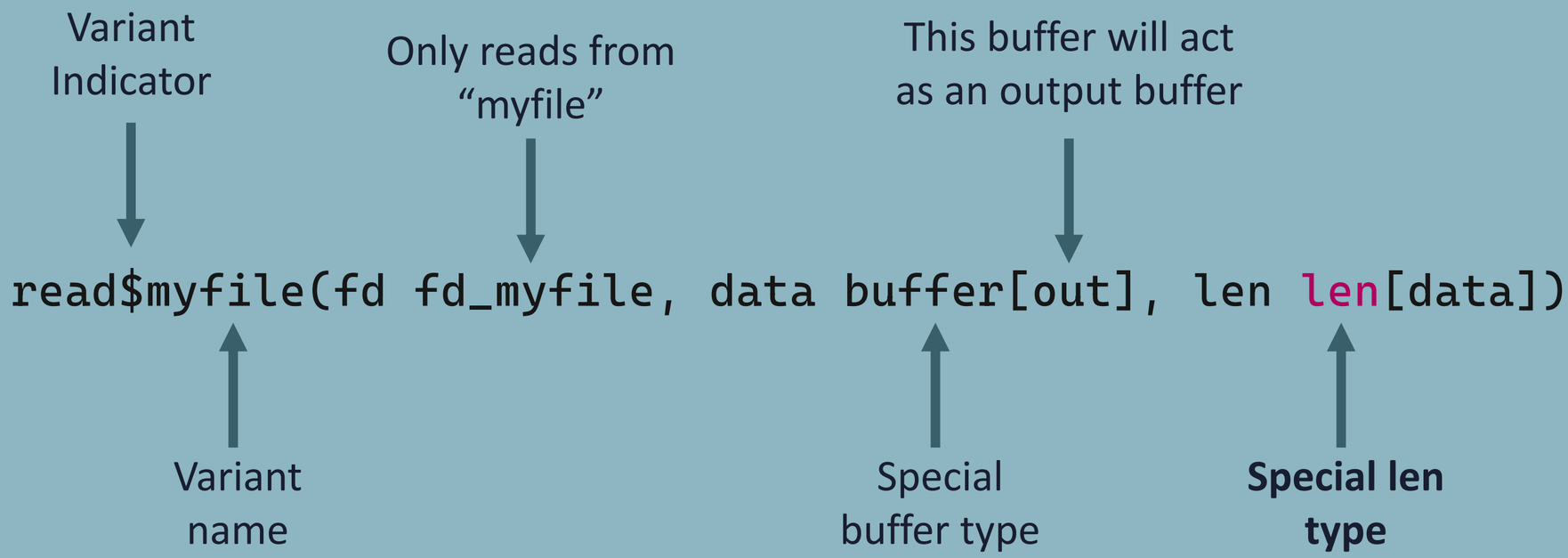
Syzkaller Overview

syzlang - syscalls



Syzkaller Overview

syzlang - syscalls



Syzkaller Overview

syzlang - syscalls

Only reads from
“myfile”

```
read$myfile(fd fd_myfile, data buffer[out], len len[data])
```

Syzkaller Overview

syzlang - resources

“Resources represent values that need to be passed from output of one syscall to input of another syscall”

Syzkaller Overview

syzlang - resources

"Resources represent values that need to be passed from output of one syscall to input of another syscall"

```
resource fd_myfile[fd]
```

Syzkaller Overview

syzlang - resources

“Resources represent values that need to be passed from output of one syscall to input of another syscall”

```
resource fd_myfile[fd]
```

Resource declaration

Syzkaller Overview

syzlang - resources

"Resources represent values that need to be passed from output of one syscall to input of another syscall"

```
resource fd_myfile[fd]
```

Resource declaration

Resource name

Syzkaller Overview

syzlang - resources

"Resources represent values that need to be passed from output of one syscall to input of another syscall"

```
resource fd_myfile[fd]
```

Resource declaration

Resource name

Resource underlying type

Syzkaller Overview

syzlang - resources

"Resources represent values that need to be passed from output of one syscall to input of another syscall"

```
resource fd_myfile[fd]
```

Syzkaller Overview

syzlang - resources

"Resources represent values that need to be passed from output of one syscall to input of another syscall"

```
resource fd_myfile[fd]
```

Syzkaller Overview

syzlang - resources

"Resources represent values that need to be passed from output of one syscall to input of another syscall"

```
resource fd_myfile[fd]

open$myfile(file string[in, "/myfile"], f flags[open_flags], mode flags[open_mode])
```

Syzkaller Overview

syzlang - resources

"Resources represent values that need to be passed from output of one syscall to input of another syscall"

```
resource fd_myfile[fd]

open$myfile(file string[in, "/myfile"], f flags[open_flags], mode flags[open_mode]) fd_myfile
```

Syzkaller Overview

syzlang - resources

"Resources represent values that need to be passed from output of one syscall to input of another syscall"

```
resource fd_myfile[fd]

open$myfile(file string[in, "/myfile"], f flags[open_flags], mode flags[open_mode]) fd_myfile

read$myfile(fd fd_myfile, data buffer[out], len len[data])
```

Syzkaller Overview

syzlang - resources

"Resources represent values that need to be passed from output of one syscall to input of another syscall"

```
resource fd_myfile[fd]

open$myfile(file string[in, "/myfile"], f flags[open_flags], mode flags[open_mode]) fd_myfile

read$myfile(fd fd_myfile, data buffer[out], len len[data])
```

Syzkaller Overview

syzlang - structs

Syzkaller Overview

syzlang - structs

```
struct foo {
        __u32   field1;
        __u64   field2;
        __u8    arr[32];
        __u16   *ptr;
}
```

Syzkaller Overview

syzlang - structs

Linux Sources - C

```
struct foo {
        __u32   field1;
        __u64   field2;
        __u8    arr[32];
        __u16   *ptr;
}
```

Syzkaller Description - syzlang

```
foo {
        field1 int32
        field2 int64
        arr array[int8, 32]
        ptr ptr[in, int16]
}
```

Syzkaller Overview

syzlang - structs

Linux Sources - C

```
struct foo {
        __u32   field1;
        __u64   field2;
        __u8    arr[32];
        __u16   *ptr;
}
```

Syzkaller Description - syzlang

```
foo {
        field1 int32
        field2 int64
        arr array[int8, 32]
        ptr ptr[in, int16]
}
```

Syzkaller Overview

syzlang - structs

Linux Sources - C

```
struct foo {
        __u32   field1;
        __u64   field2;
        __u8    arr[32];
        __u16   *ptr;
}
```

Syzkaller Description - syzlang

```
foo {
        field1 int32
        field2 int64
        arr array[int8, 32]
        ptr ptr[in, int16]
}
```

Syzkaller Overview

syzlang - structs

Linux Sources - C

```
struct foo {
        __u32   field1;
        __u64   field2;
        __u8    arr[32];
        __u16   *ptr;
}
```

Syzkaller Description - syzlang

```
foo {
        field1 int32
        field2 int64
        arr array[int8, 32]
        ptr ptr[in, int16]
}
```

Syzkaller Overview

syzlang - structs

Linux Sources - C

```
struct foo {
        __u32   field1;
        __u64   field2;
        __u8    arr[32];
        __u16   *ptr;
}
```

Syzkaller Description - syzlang

```
foo {
        field1 int32
        field2 int64
        arr array[int8, 32]
        ptr ptr[in, int16]
}
```

Syzkaller Overview

syzlang - structs

Linux Sources - C

```
struct foo {
        __u32   field1;
        __u64   field2;
        __u8    arr[32];
        __u16   *ptr;
}
```

Syzkaller Description - syzlang

```
foo {
        field1 int32
        field2 int64
        arr array[int8, 32]
        ptr ptr[in, int16]
}
```

Syzkaller Overview

syzlang - structs

Linux Sources - C

```
struct foo {
        __u32   field1;
        __u64   field2;
        __u8    arr[32];
        __u16   *ptr;
}
```

Syzkaller Description - syzlang

```
foo {
        field1 int32
        field2 int64
        arr array[int8, 32]
        ptr ptr[in, int16]
}
```

Syzkaller Overview

syzlang - structs

Linux Sources - C

```
struct foo {
        __u32   field1;
        __u64   field2;
        __u8    arr[32];
        __u16   *ptr;
}
```

Syzkaller Description - syzlang

```
foo {
        field1 int32
        field2 int64
        arr array[int8, 32]
        ptr ptr[out, int16]
}
```

Syzkaller Overview

coverage

Syzkaller Overview

coverage

KCOV

Syzkaller Overview

coverage - KCOV

- Kernel code coverage subsystem
- Designed to collect coverage information for running tasks
- Used by syzkaller to understand where it reached in the kernel code for better mutations

Syzkaller Overview

coverage - KCOV

Syzkaller Overview

coverage - KCOV

- During the kernel build, the compiler adds a call to `__sanitizer_cov_trace_pc` to every basic block

```
int my_kern_func {
    if (...) {
        ...
    }
    return 0;
}
```

Syzkaller Overview

coverage - KCOV

- During the kernel build, the compiler adds a call to `__sanitizer_cov_trace_pc` to every basic block

```
int my_kern_func {
    __sanitizer_cov_trace_pc();
    if (...) {
        __sanitizer_cov_trace_pc();
        ...
    }
     __sanitizer_cov_trace_pc();
    return 0;
}
```

Syzkaller Overview

coverage - KCOV

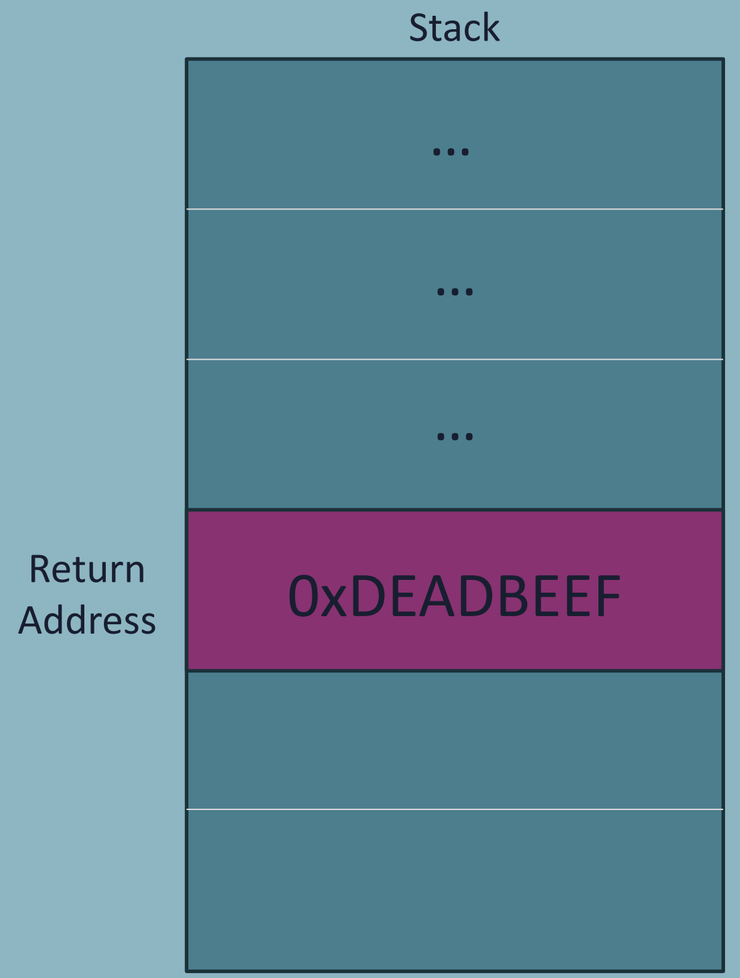
- During the kernel build, the compiler adds a call to `__sanitizer_cov_trace_pc` to every basic block
- When a basic block is executed, the kernel saves the return address when `__sanitizer_cov_trace_pc` returns.

```
int my_kern_func {
    __sanitizer_cov_trace_pc();
    if (...) {
        __sanitizer_cov_trace_pc();
        ...
    }
     __sanitizer_cov_trace_pc();
    return 0;
}
```

Syzkaller Overview

coverage - KCOV

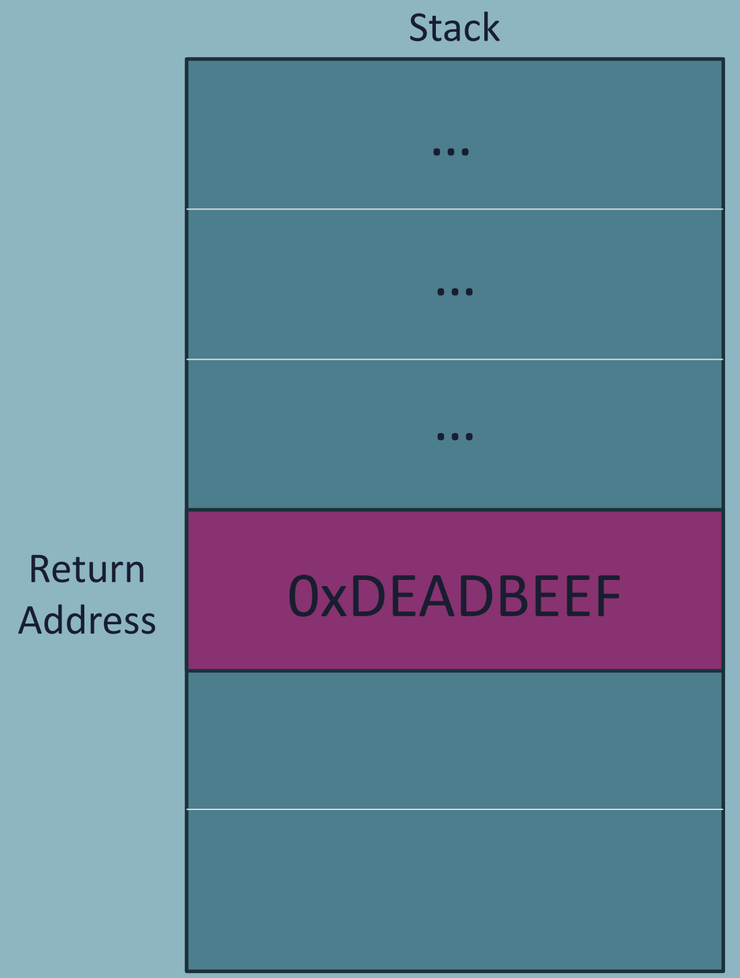
- During the kernel build, the compiler adds a call to `__sanitizer_cov_trace_pc` to every basic block
- When a basic block is executed, the kernel saves the return address when `__sanitizer_cov_trace_pc` returns.



Syzkaller Overview

coverage - KCOV

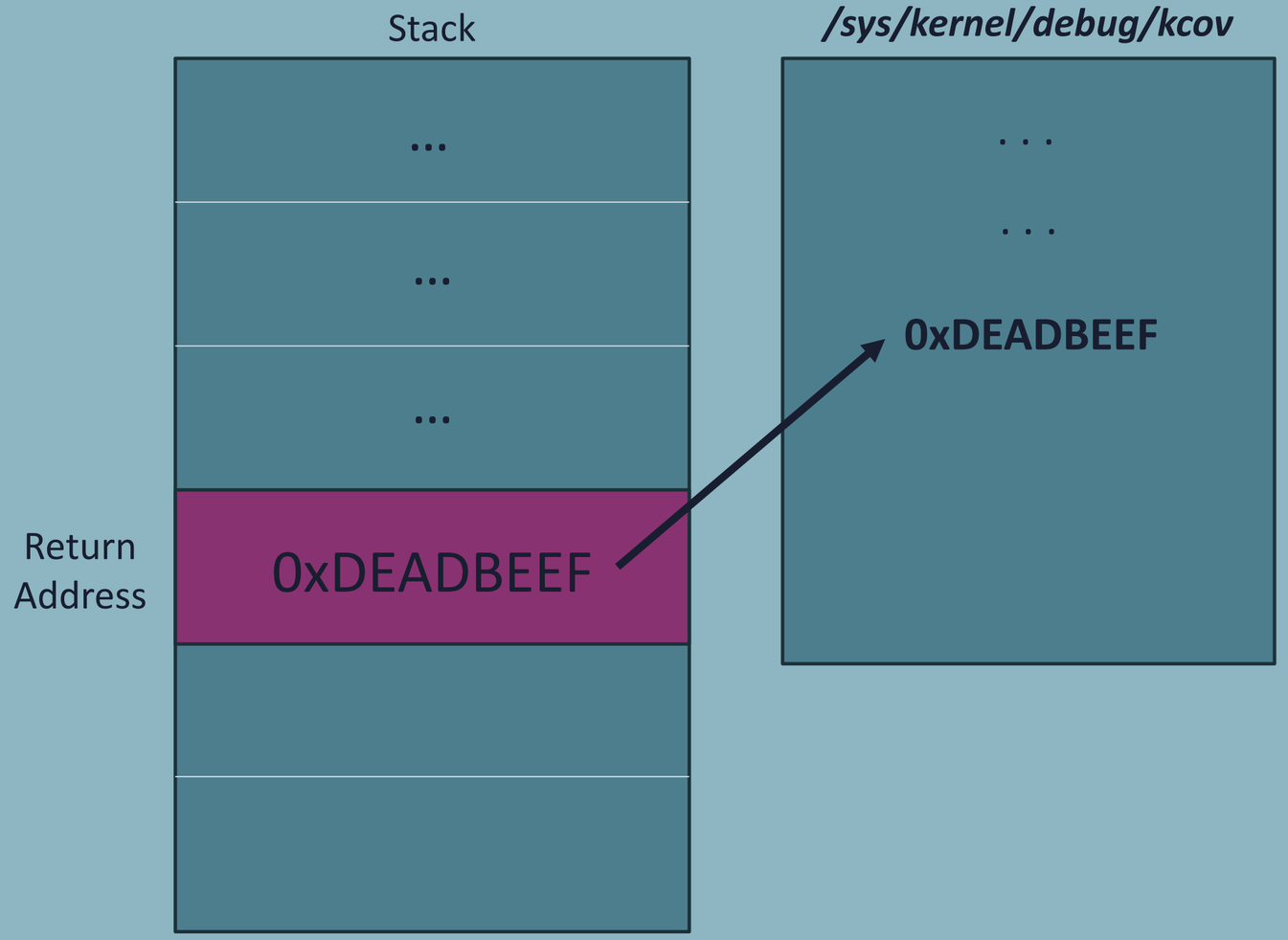
- During the kernel build, the compiler adds a call to `__sanitizer_cov_trace_pc` to every basic block
- When a basic block is executed, the kernel saves the return address when `__sanitizer_cov_trace_pc` returns.
- The addresses are saved in the ***`/sys/kernel/debug/kcov`*** *file*



Syzkaller Overview

coverage - KCOV

- During the kernel build, the compiler adds a call to `__sanitizer_cov_trace_pc` to every basic block
- When a basic block is executed, the kernel saves the return address when `__sanitizer_cov_trace_pc` returns.
- The addresses are saved in the ***`/sys/kernel/debug/kcov`*** *file*



Syzkaller Overview

coverage - KCOV

Remote KCOV

Syzkaller Overview

coverage - Remote KCOV

Syzkaller Overview

coverage - Remote KCOV

- Standard KCOV cannot collect coverage of tasks other than the current one (such as asynchronous kernel tasks)

Syzkaller Overview

coverage - Remote KCOV

- Standard KCOV cannot collect coverage of tasks other than the current one (such as asynchronous kernel tasks)
- To collect kernel task coverage data, we can use **remote KCOV**

Syzkaller Overview

coverage - Remote KCOV

- Standard KCOV cannot collect coverage of tasks other than the current one (such as asynchronous kernel tasks)
- To collect kernel task coverage data, we can use **remote KCOV**
- The kernel needs to be modified and compiled with remote KCOV annotations

Syzkaller Overview

coverage - Remote KCOV

- Standard KCOV cannot collect coverage of tasks other than the current one (such as asynchronous kernel tasks)
- To collect kernel task coverage data, we can use **remote KCOV**
- The kernel needs to be modified and compiled with remote KCOV annotations

Syzkaller Overview

coverage - Remote KCOV

- Standard KCOV cannot collect coverage of tasks other than the current one (such as asynchronous kernel tasks)
- To collect kernel task coverage data, we can use **remote KCOV**
- The kernel needs to be modified and compiled with remote KCOV annotations

---

```
kcov_remote_start(REMOTE_HANDLE);
```

- Start collecting coverage for remote task
- The coverage data will be saved to a specific location, determined by a unique u64 `REMOTE_HANDLE`

Syzkaller Overview

coverage - Remote KCOV

- Standard KCOV cannot collect coverage of tasks other than the current one (such as asynchronous kernel tasks)
- To collect kernel task coverage data, we can use **remote KCOV**
- The kernel needs to be modified and compiled with remote KCOV annotations

---

`kcov_remote_start(REMOTE_HANDLE);`

- Start collecting coverage for remote task
- The coverage data will be saved to a specific location, determined by a unique u64 `REMOTE_HANDLE`

`kcov_remote_stop();`

- Stop collecting coverage for remote task

Syzkaller Overview

coverage - Remote KCOV

```
void my_kernel_task(int a) {
    if (a < 10) {
        foo();
    }
    else {
        boo();
    }
}
```

Syzkaller Overview

coverage - Remote KCOV

```
void my_kernel_task(int a) {
    kcov_remote_start(REMOTE_HANDLE);
    if (a < 10) {
        foo();
    }
    else {
        boo();
    }
    kcov_remote_stop();
}
```

2

NVMe-oF/TCP Overview

NVMe-oF/TCP Overview

- NVMe is a protocol used to control PCI devices like SSDs)
- NVMe-oF (NVMe over Fabrics) is an extension of NVMe that allows remote storage access
- NVMe-oF/TCP allows NVMe-oF using a TCP connection
- NVMe-oF/TCP is mainly used in data centers and cloud environments

NVMe-oF/TCP Attack Surface

Gets inputs through network

Parse data in kernel space

May lead up to remote kernel code execution

3

Adding NVMe to syzkaller

Adding NVMe to syzkaller

- [ ] Define syscalls for syzkaller to send NVMe-oF/TCP payloads
- [ ] Define NVMe-oF/TCP structure
- [ ] Define syscalls for syzkaller to open sockets for NVMe-oF/TCP

Adding NVMe to syzkaller

- [ ] **Define syscalls for syzkaller to send NVMe-oF/TCP payloads**
- [ ] Define NVMe-oF/TCP structure
- [ ] Define syscalls for syzkaller to open sockets for NVMe-oF/TCP

Adding NVMe to syzkaller

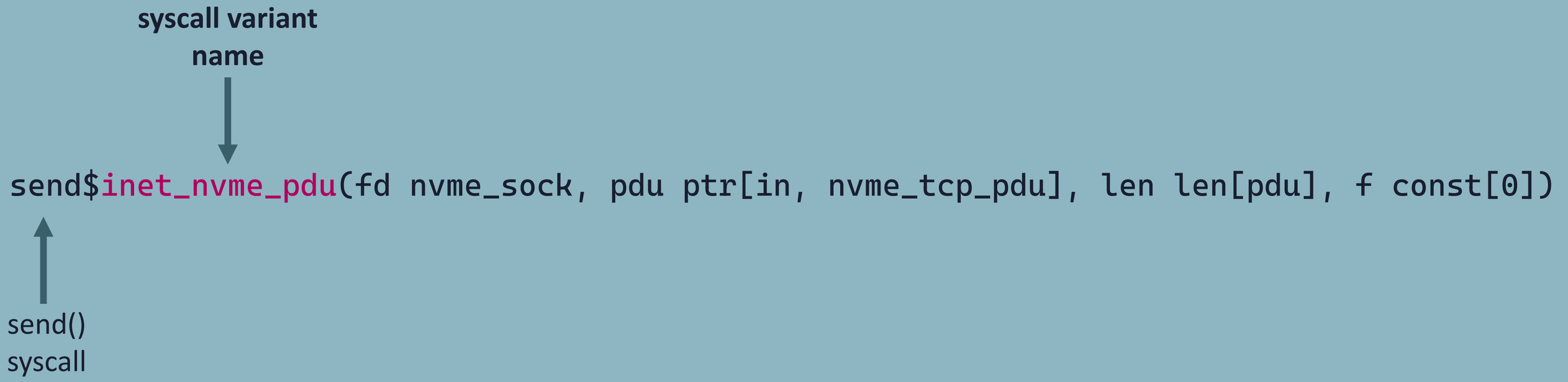
Defining syscalls for syzkaller to send NVMe-oF/TCP payloads

```
send$inet_nvme_pdu(fd nvme_sock, pdu ptr[in, nvme_tcp_pdu], len len[pdu], f const[0])
```

send()
syscall

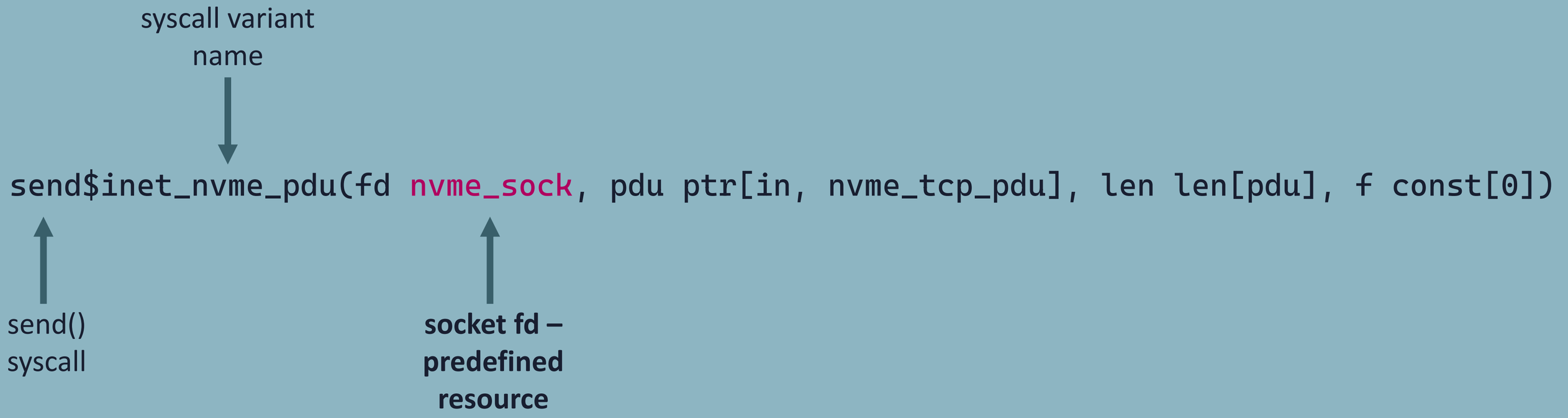
Adding NVMe to syzkaller

Defining syscalls for syzkaller to send NVMe-oF/TCP payloads



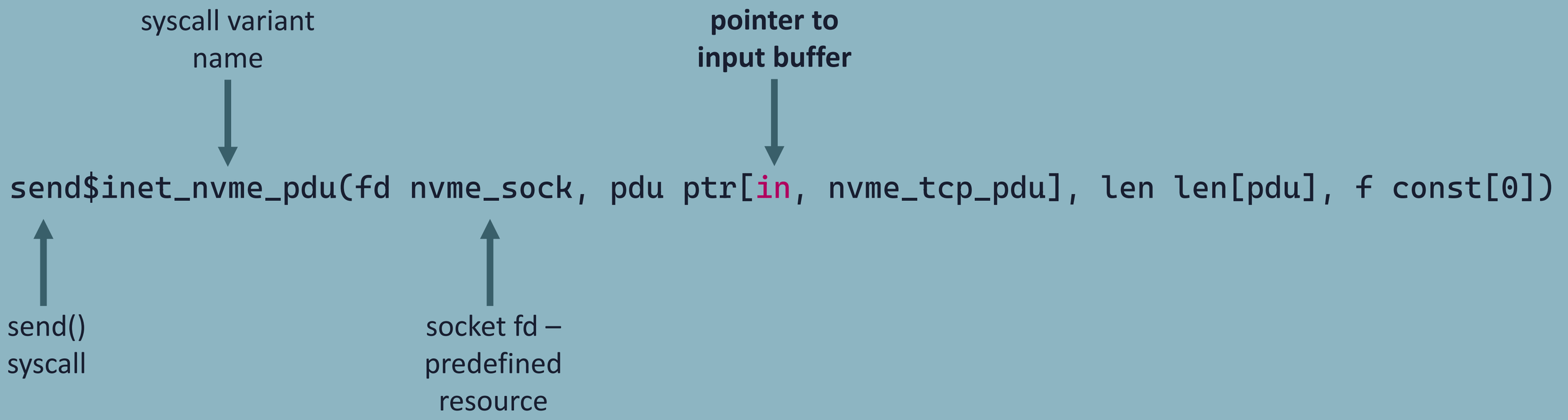
Adding NVMe to syzkaller

Defining syscalls for syzkaller to send NVMe-oF/TCP payloads



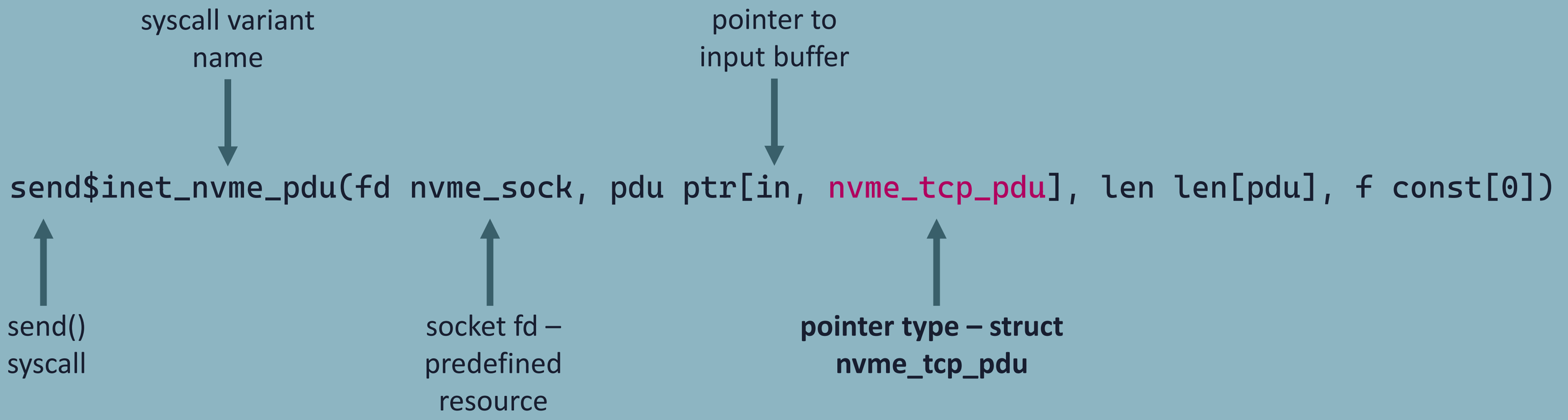
Adding NVMe to syzkaller

Defining syscalls for syzkaller to send NVMe-oF/TCP payloads



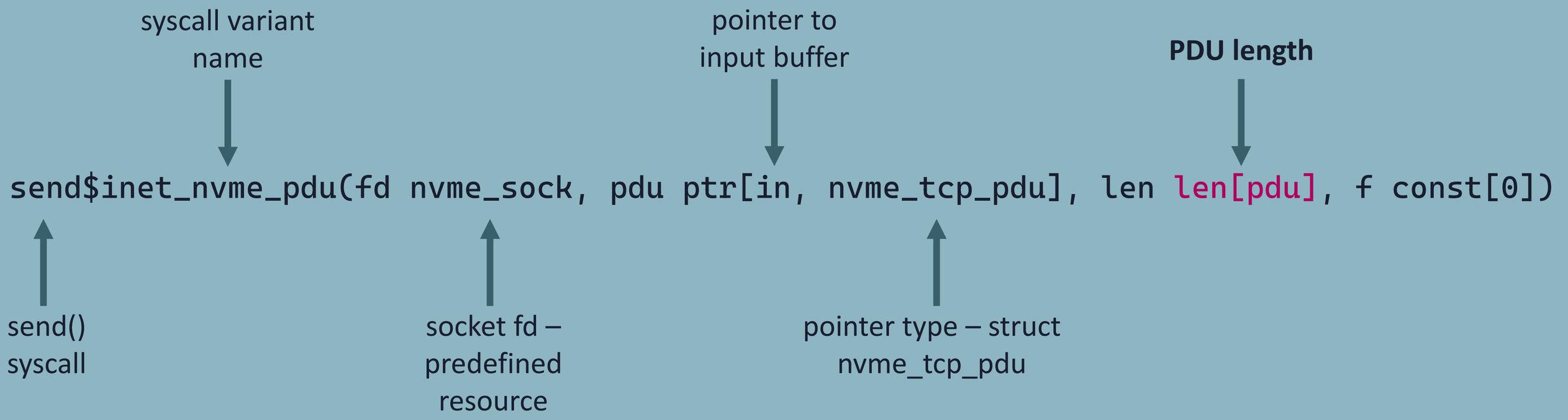
Adding NVMe to syzkaller

Defining syscalls for syzkaller to send NVMe-oF/TCP payloads



Adding NVMe to syzkaller

Defining syscalls for syzkaller to send NVMe-oF/TCP payloads



Adding NVMe to syzkaller

- [x] **Define syscalls for syzkaller to send NVMe-oF/TCP payloads**
- [ ] Define NVMe-oF/TCP structure
- [ ] Define syscalls for syzkaller to open sockets for NVMe-oF/TCP

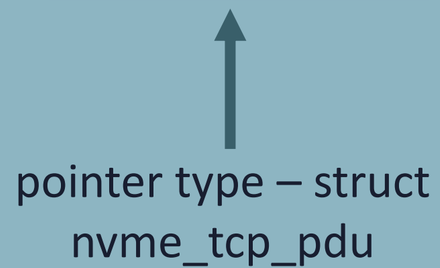
Adding NVMe to syzkaller

- [x] Define syscalls for syzkaller to send NVMe-oF/TCP payloads
- [ ] **Define NVMe-oF/TCP structure**
- [ ] Define syscalls for syzkaller to open sockets for NVMe-oF/TCP

Adding NVMe to syzkaller

Defining structs for syzkaller to send NVMe-oF/TCP payloads

```
send$inet_nvme_pdu(fd nvme_sock, pdu ptr[in, nvme_tcp_pdu], len len[pdu], f const[0])
```



Adding NVMe to syzkaller

Defining structs for syzkaller to send NVMe-oF/TCP payloads

```
nvme_tcp_pdu [
    icreq       nvme_tcp_icreq_pdu
    icresp      nvme_tcp_icresp_pdu
    cmd         nvme_tcp_cmd_pdu
    rsp         nvme_tcp_rsp_pdu
    r2t         nvme_tcp_r2t_pdu
    data        nvme_tcp_data_pdu
]
```

Adding NVMe to syzkaller

Defining structs for syzkaller to send NVMe-oF/TCP payloads

```
nvme_tcp_data_pdu {
    hdr             nvme_tcp_hdr_data_pdu_union
    command_id      int16
    ttag            int16
    data_offset     int32
    data_length     int32
    rsvd            array[int8, 4]
} [size[NVME_TCP_DATA_PDU_SIZE]]
```

Adding NVMe to syzkaller

Defining structs for syzkaller to send NVMe-oF/TCP payloads

```
nvme_tcp_data_pdu {
    hdr             nvme_tcp_hdr_data_pdu_union
    command_id      int16
    ttag            int16
    data_offset     int32
    data_length     int32
    rsvd            array[int8, 4]
} [size[NVME_TCP_DATA_PDU_SIZE]]
```

Adding NVMe to syzkaller

- [x] Define syscalls for syzkaller to send NVMe-oF/TCP payloads
- [x] **Define NVMe-oF/TCP structure**
- [ ] Define syscalls for syzkaller to open sockets for NVMe-oF/TCP

Adding NVMe to syzkaller

- [x] Define syscalls for syzkaller to send NVMe-oF/TCP payloads
- [x] Define NVMe-oF/TCP structure
- [ ] **Define syscalls for syzkaller to open sockets for NVMe-oF/TCP**

Adding NVMe to syzkaller

Defining syscalls for syzkaller to open NVMe-oF/TCP payloads

```
send$inet_nvme_pdu(fd nvme_sock, pdu ptr[in, nvme_tcp_pdu], len len[pdu], f const[0])
```

socket fd – predefined resource

Adding NVMe to syzkaller

Defining syscalls for syzkaller to open NVMe-oF/TCP payloads

socket()

Syzkaller inputs run inside a network ns sandbox

Adding NVMe to syzkaller

Defining syscalls for syzkaller to open NVMe-oF/TCP payloads

New Sandbox

Syzkaller’s
maintainers didn’t
like this idea

Adding NVMe to syzkaller

Defining syscalls for syzkaller to open NVMe-oF/TCP payloads

syz_init_net_socket()

This pseudo-syscall doesn’t allow the creation of TCP sockets

Adding NVMe to syzkaller

Defining syscalls for syzkaller to open NVMe-oF/TCP payloads

New pseudo-syscall

Adding NVMe to syzkaller

Defining syscalls for syzkaller to open NVMe-oF/TCP payloads

```
static long syz_socket_connect_nvme_tcp()
{
    struct sockaddr_in nvme_local_address;
    int netns = open("/proc/self/ns/net", O_RDONLY);
    if (netns == -1)
        return netns;
    if (setns(kInitNetNsFd, 0))
        return -1;
    int sock = syscall(__NR_socket, AF_INET, SOCK_STREAM, 0x0);
    int err = errno;
    if (setns(netns, 0)) {
        // The operation may fail if the fd is closed by
        // a syscall from another thread.
        exitf("setns(netns) failed");
    }
    close(netns);
    errno = err;
    // We only connect to an NVMe-oF/TCP server on 127.0.0.1:4420
    nvme_local_address.sin_family = AF_INET;
    nvme_local_address.sin_port = htobe16(4420);
    nvme_local_address.sin_addr.s_addr = htobe32(0x7f000001);
    err = syscall(__NR_connect, sock, &nvme_local_address, sizeof(nvme_local_address));
    if (err != 0) {
        close(sock);
        return -1;
    }
    return sock;
}
```

Adding NVMe to syzkaller

Defining syscalls for syzkaller to open NVMe-oF/TCP payloads

```
static long syz_socket_connect_nvme_tcp()
{
    struct sockaddr_in nvme_local_address;
    int netns = open("/proc/self/ns/net", O_RDONLY);
    if (netns == -1)
        return netns;
    if (setns(kInitNetNsFd, 0))
        return -1;
    int sock = syscall(__NR_socket, AF_INET, SOCK_STREAM, 0x0);
    int err = errno;
```

Adding NVMe to syzkaller

Defining syscalls for syzkaller to open NVMe-oF/TCP payloads

```
int sock = syscall(__NR_socket, AF_INET, SOCK_STREAM, 0x0);
int err = errno;
if (setns(netns, 0)) {
    // The operation may fail if the fd is closed by
    // a syscall from another thread.
    exitf("setns(netns) failed");
}
close(netns);
errno = err;
```

Adding NVMe to syzkaller

Defining syscalls for syzkaller to open NVMe-oF/TCP payloads

```
    // We only connect to an NVMe-oF/TCP server on 127.0.0.1:4420
    nvme_local_address.sin_family = AF_INET;
    nvme_local_address.sin_port = htobe16(4420);
    nvme_local_address.sin_addr.s_addr = htobe32(0x7f000001);
    err = syscall(__NR_connect, sock, &nvme_local_address, sizeof(nvme_local_address));
    if (err != 0) {
        close(sock);
        return -1;
    }
    return sock;
}
```

Adding NVMe to syzkaller

- [x] Define syscalls for syzkaller to send NVMe-oF/TCP payloads
- [x] Define NVMe-oF/TCP structure
- [x] **Define syscalls for syzkaller to open sockets for NVMe-oF/TCP**

Adding NVMe to syzkaller

coverage

Adding NVMe to syzkaller

coverage

Adding NVMe to syzkaller

coverage

```
static void nvmet_tcp_io_work(struct work_struct *w) {

    ...

}
```

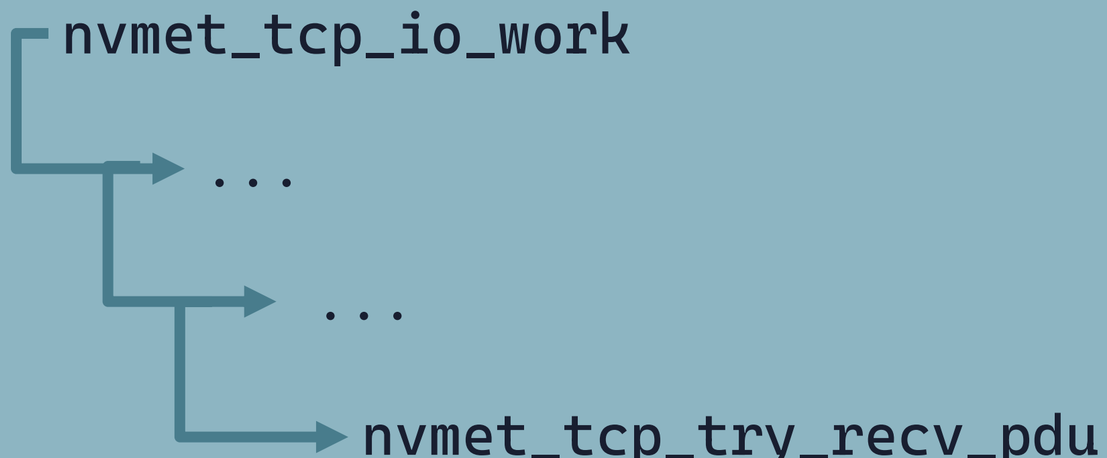
Adding NVMe to syzkaller

coverage

```
struct task_struct {
    ...

#ifdef CONFIG_KCOV
    ...
    u64 kcov_handle;
    ...
#endif
    ...
};
```

Adding NVMe to syzkaller

coverage



Adding NVMe to syzkaller

coverage

```
static int nvmet_tcp_try_recv_pdu(struct nvmet_tcp_queue *queue)
{
        ...

recv:
        ...

        len = kernel_recvmsg(queue->sock, &msg, &iov, 1,
                             iov.iov_len, msg.msg_flags);
        ...
}
```

Adding NVMe to syzkaller

coverage

```
static int nvmet_tcp_try_recv_pdu(struct nvmet_tcp_queue *queue)
{
        ...

recv:
        ...

        len = kernel_recvmsg(queue->sock, &msg, &iov, 1,
                             iov.iov_len, msg.msg_flags);

        kcov_remote_start_common(UNIQUE_KCOV_HANDLE);
        ...
}
```

Adding NVMe to syzkaller

coverage

```
static int nvmet_tcp_try_recv_pdu(struct nvmet_tcp_queue *queue)
{
        ...

recv:
        ...

        len = kernel_recvmsg(queue->sock, &msg, &iov, 1,
                             iov.iov_len, msg.msg_flags);

        kcov_remote_start_common(UNIQUE_KCOV_HANDLE);
        ...
}
```

Adding NVMe to syzkaller

coverage

```
static int nvmet_tcp_try_recv_pdu(struct nvmet_tcp_queue *queue)
{
        ...

recv:
        ...

        len = kernel_recvmsg(queue->sock, &msg, &iov, 1,
                             iov.iov_len, msg.msg_flags);

        kcov_remote_start_common(UNIQUE_KCOV_HANDLE);
        ...
}
```

Adding NVMe to syzkaller

coverage

```
struct msghdr {
        void    *msg_name;
        int     msg_namelen;
        ...

#ifdef CONFIG_KCOV
        u64     kcov_handle;
#endif
};
```

Adding NVMe to syzkaller

coverage

```
static inline void msghdr_set_kcov_handle
    (struct msghdr *msg,
    const u64 kcov_handle)
{
#ifdef CONFIG_KCOV
    msg->kcov_handle = kcov_handle;
#endif
}
```

Adding NVMe to syzkaller

coverage

```
static inline u64 msghdr_get_kcov_handle
    (struct msghdr *msg)
{
#ifdef CONFIG_KCOV
     return msg->kcov_handle;
#else
    return 0;
#endif
}
```

Adding NVMe to syzkaller

coverage

```
static int nvmet_tcp_try_recv_pdu(struct nvmet_tcp_queue *queue)
{
        ...

recv:
        ...

        len = kernel_recvmsg(queue->sock, &msg  &iov, 1,
                             iov.iov_len, msg.msg_flags);

        kcov_remote_start_common(UNIQUE_KCOV_HANDLE);
        ...
}
```

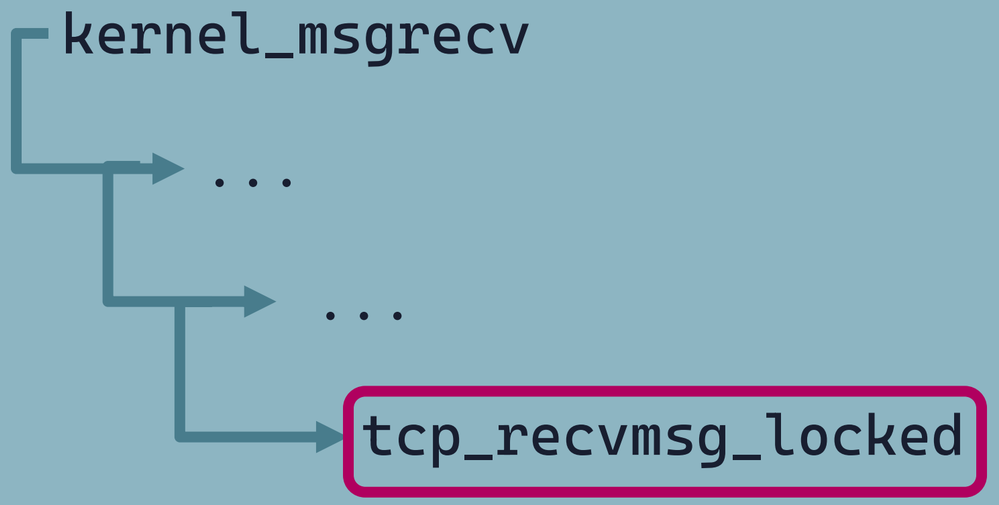
Adding NVMe to syzkaller

coverage

```
static int nvmet_tcp_try_recv_pdu(struct nvmet_tcp_queue *queue)
{
        ...

recv:
        ...

        len = kernel_recvmsg(queue->sock, &msg, &iov, 1,
                             iov.iov_len, msg.msg_flags);

        kcov_remote_start_common(UNIQUE_KCOV_HANDLE);
        ...
}
```

Adding NVMe to syzkaller

coverage



Adding NVMe to syzkaller

coverage

```
static int tcp_recvmsg_locked(struct sock *sk, struct msghdr *msg, ...)
{
    ...

found_ok_skb:
    /* Ok so how much can we use? */
    used = skb->len - offset; if (len < used) used = len;

    ...
}
```

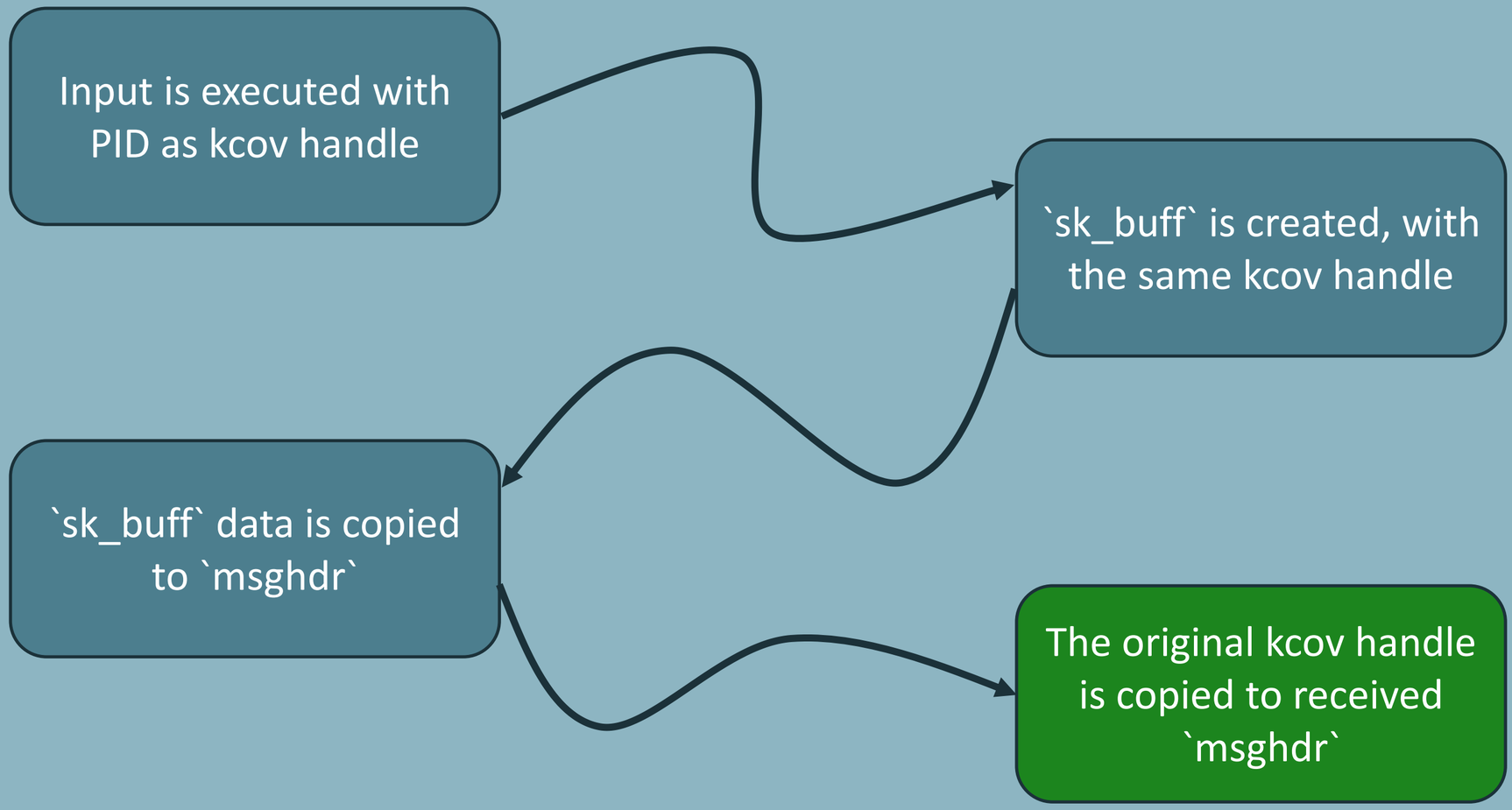
Adding NVMe to syzkaller

coverage

```
static int tcp_recvmsg_locked(struct sock *sk, struct msghdr *msg, ...)
{
    ...

found_ok_skb:
    /* Ok so how much can we use? */
    used = skb->len - offset; if (len < used) used = len;

#ifdef CONFIG_KCOV
    msghdr_set_kcov_handle(msg, skb_get_kcov_handle(skb));
#endif


    ...
}
```

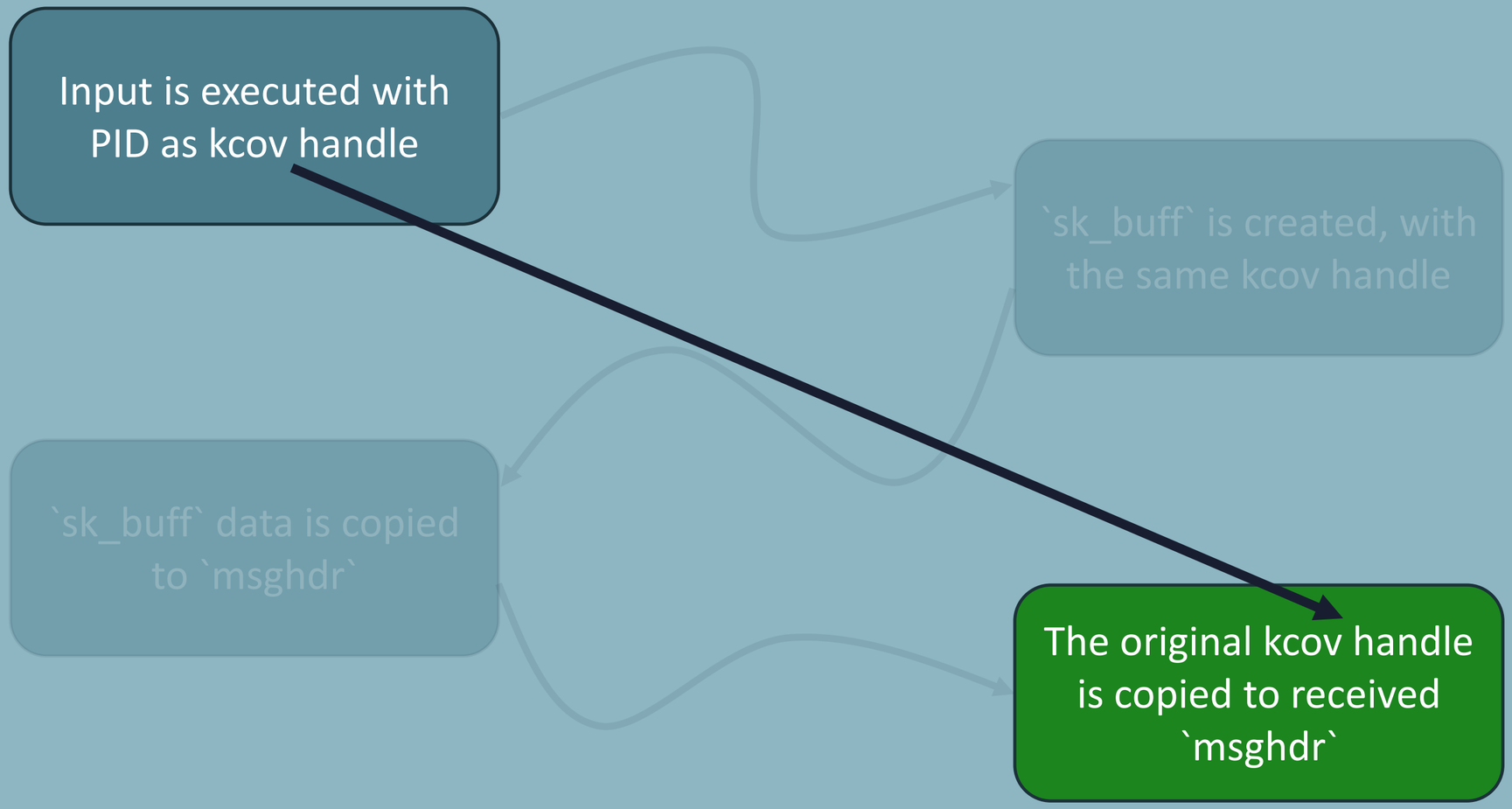
Adding NVMe to syzkaller

coverage



Adding NVMe to syzkaller

coverage



Adding NVMe to syzkaller

coverage

```
static int nvmet_tcp_try_recv_pdu(struct nvmet_tcp_queue *queue)
{
        ...

recv:
        ...

        len = kernel_recvmsg(queue->sock, &msg, &iov, 1,
                             iov.iov_len, msg.msg_flags);

        kcov_remote_start_common(UNIQUE_KCOV_HANDLE);
        ...
}
```

Adding NVMe to syzkaller

coverage

```
static int nvmet_tcp_try_recv_pdu(struct nvmet_tcp_queue *queue)
{
        ...

recv:
        ...

        len = kernel_recvmsg(queue->sock, &msg, &iov, 1,
                             iov.iov_len, msg.msg_flags);

        kcov_remote_start_common(msghdr_get_kcov_handle(&msg));
        ...
}
```

Adding NVMe to syzkaller

coverage

```
static int nvmet_tcp_try_recv_pdu(struct nvmet_tcp_queue *queue)
{
        ...

recv:
        ...

        len = kernel_recvmsg(queue->sock, &msg, &iov, 1,
                             iov.iov_len, msg.msg_flags);

        if (!kcov_started) {
            kcov_started = 1;
            kcov_remote_start_common(msghdr_get_kcov_handle(&msg));
        }
}
```

Demo

4

Conclusion

Conclusion

Conclusion

Findings

Conclusion

Findings

KASAN: slab-use-after-free Read in process_one_work
KASAN: slab-out-of-bounds Read in nvmet_ctrl_find_get
BUG: unable to handle kernel NULL pointer dereference in __nvmet_req_complete
BUG: unable to handle kernel NULL pointer dereference in nvmet_tcp_build_pdu_iovec
BUG: unable to handle kernel NULL pointer dereference in nvmet_tcp_io_work

Conclusion

Findings

KASAN: slab-use-after-free Read in process_one_work	**CVE-2023-5218 (CVSS 9.8)**
KASAN: slab-out-of-bounds Read in nvmet_ctrl_find_get	CVE-2023-6121 (CVSS 4.3)
BUG: unable to handle kernel NULL pointer dereference in __nvmet_req_complete	CVE-2023-6536 (CVSS 7.5)
BUG: unable to handle kernel NULL pointer dereference in nvmet_tcp_build_pdu_iovec	CVE-2023-6356 (CVSS 7.5)
BUG: unable to handle kernel NULL pointer dereference in nvmet_tcp_io_work	CVE-2023-6535 (CVSS 7.5)

Conclusion

- [x] A pull request with those changes was approved
- [x] Syzkaller can now fuzz NVMe-oF/TCP

Conclusion

Future Work

- Send a patch to Linux kernel with KCOV changes
- Improve NVMe-oF/TCP description and coverage
- Add support for new subsystems

Thank you

?

Questions

Your NVMe had Been Syz’ed

Blog post