SyzDescribe: Principled, Automated, Static Generation of Syscall Descriptions for Kernel Drivers

Yu Hao[1], Guoren Li[1], Xiaochen Zou[1], Weiteng Chen[1], Shitong Zhu[1], Zhiyun Qian[1], Ardalan Amiri Sani[2]

[1]University of California, Riverside, [2]University of California, Irvine

[1]{yhao016, gli076, xzou017, wchen130, szhu014}@ucr.edu, zhiyunq@cs.ucr.edu, [2]ardalan@uci.edu

Linux Kernel, Fuzzing, Syzkaller

Linux kernel

is widely used in servers (Linux Distribution), cell phones (Android) in the world.

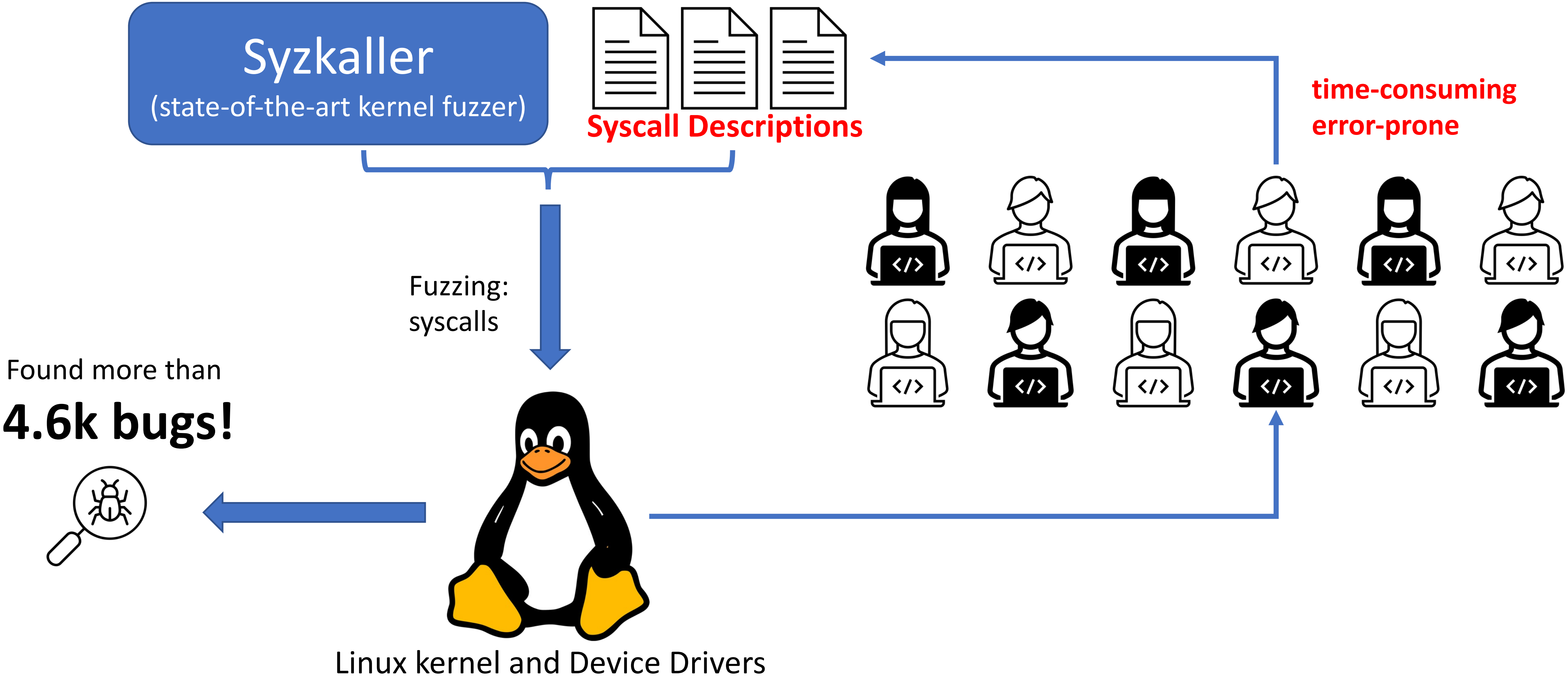
Fuzzing

has become one of the most popular and essential methods for uncovering bugs and vulnerabilities.

Syzkaller

which is the state-of-the-art kernel fuzzer, has found or fixed more than 6k bugs in the Linux kernel.

Kernel Fuzzing, Syzkaller & Syscall Descriptions



Syzkaller

(state-of-the-art kernel fuzzer)

Syscall Descriptions

time-consuming
error-prone

Fuzzing:
syscalls

Found more than

4.6k bugs!

Linux kernel and Device Drivers

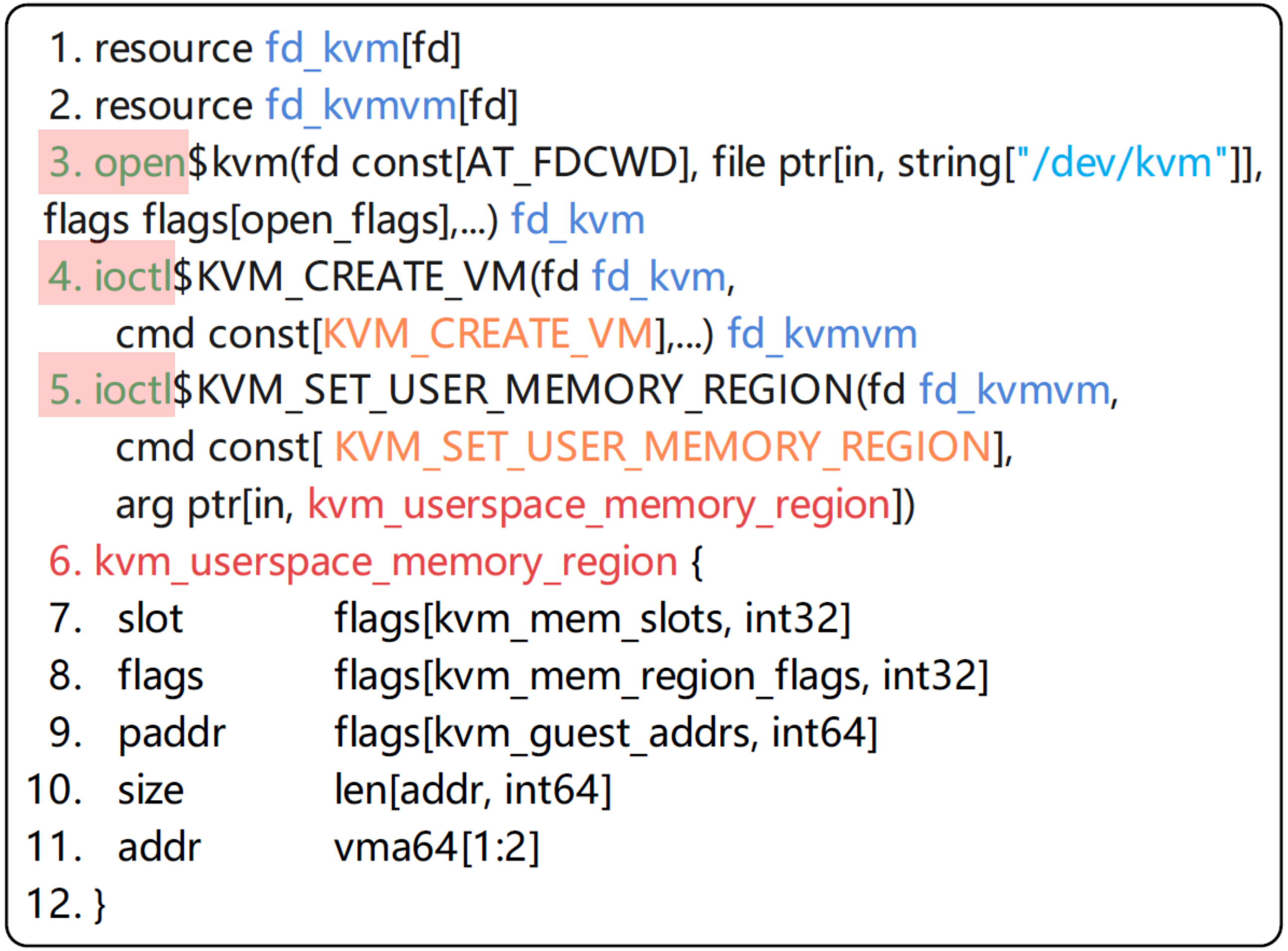
Syscall Descriptions

🟩 Syscalls interface 🟦 Device file name

🟧 Command value 🟥 Argument type

🟦 Explicit dependency

non-open file descriptor dependency:
e.g., fd_kvmvm



```
resource fd_kvm[fd]
resource fd_kvmvm[fd]
open$kvm(fd const[AT_FDCWD], file ptr[in, string["/dev/kvm"]],
flags flags[open_flags],...) fd_kvm
ioctl$KVM_CREATE_VM(fd fd_kvm,
    cmd const[KVM_CREATE_VM],...) fd_kvmvm
ioctl$KVM_SET_USER_MEMORY_REGION(fd fd_kvmvm,
    cmd const[ KVM_SET_USER_MEMORY_REGION],
    arg ptr[in, kvm_userspace_memory_region])
kvm_userspace_memory_region {
    slot        flags[kvm_mem_slots, int32]
    flags       flags[kvm_mem_region_flags, int32]
    paddr       flags[kvm_guest_addrs, int64]
    size        len[addr, int64]
    addr        vma64[1:2]
}
```

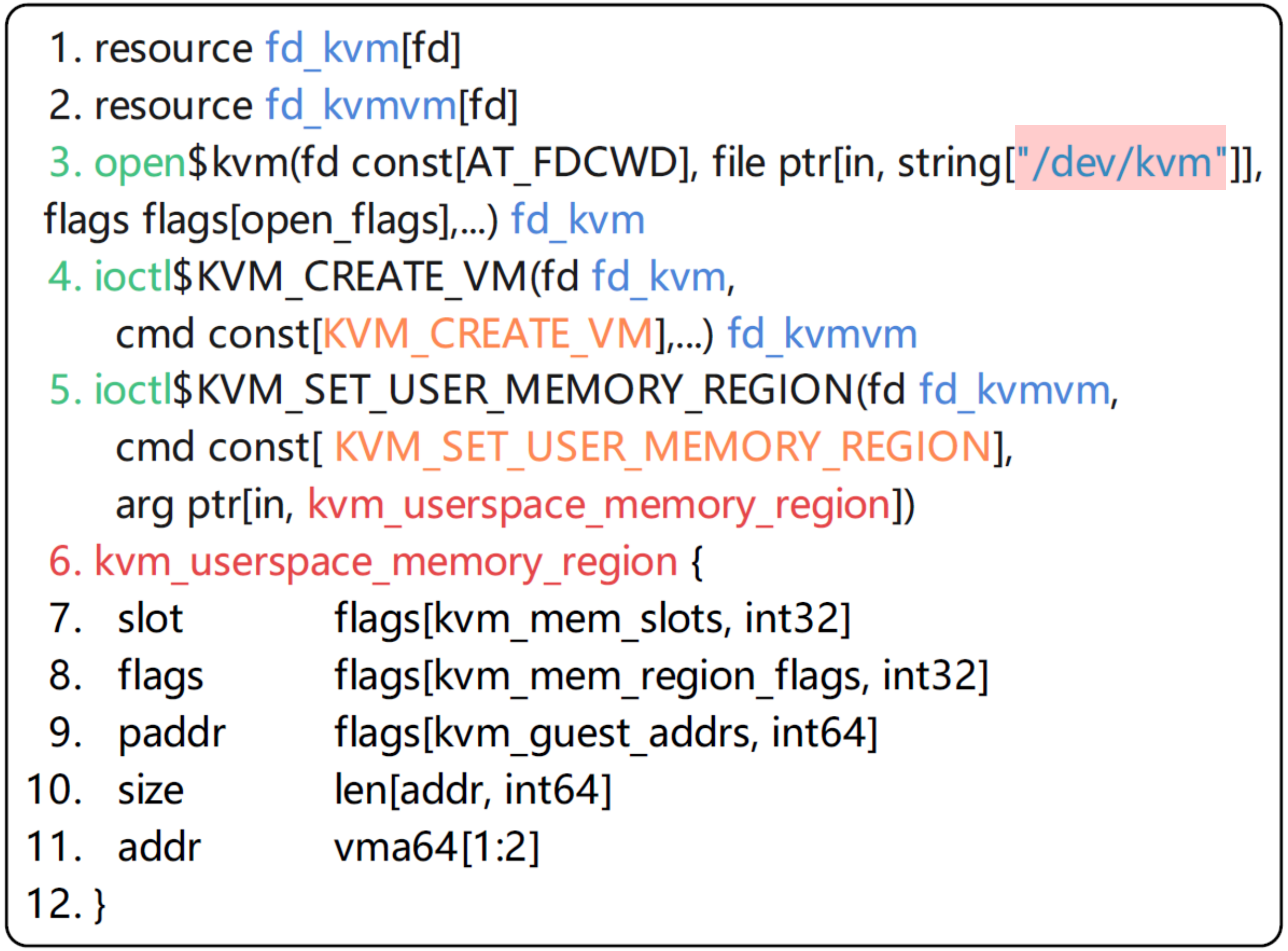
Syscall Descriptions

🟩 **Syscalls interface** 🟦 **Device file name**

🟧 **Command value** 🟥 **Argument type**

🟦 **Explicit dependency**

non-open file descriptor dependency:
e.g., fd_kvmvm



```
resource fd_kvm[fd]
resource fd_kvmvm[fd]
open$kvm(fd const[AT_FDCWD], file ptr[in, string["/dev/kvm"]],
flags flags[open_flags],...) fd_kvm
ioctl$KVM_CREATE_VM(fd fd_kvm,
    cmd const[KVM_CREATE_VM],...) fd_kvmvm
ioctl$KVM_SET_USER_MEMORY_REGION(fd fd_kvmvm,
    cmd const[ KVM_SET_USER_MEMORY_REGION],
    arg ptr[in, kvm_userspace_memory_region])
kvm_userspace_memory_region {
    slot        flags[kvm_mem_slots, int32]
    flags       flags[kvm_mem_region_flags, int32]
    paddr       flags[kvm_guest_addrs, int64]
    size        len[addr, int64]
    addr        vma64[1:2]
}
```

Syscall Descriptions

🟩 Syscalls interface

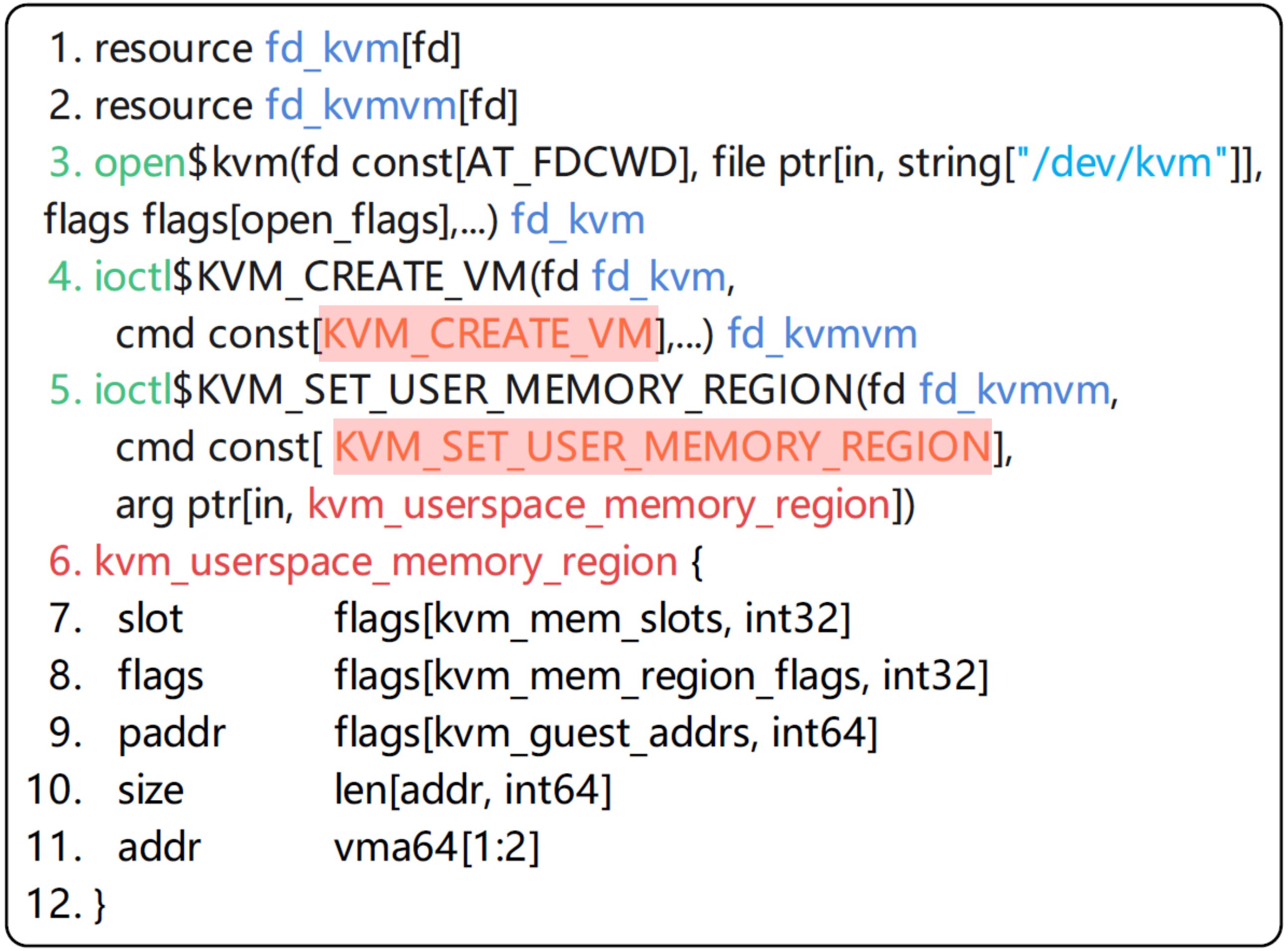
🟦 Device file name

🟧 Command value

🟥 Argument type

🟦 Explicit dependency

non-open file descriptor dependency:
e.g., fd_kvmvm

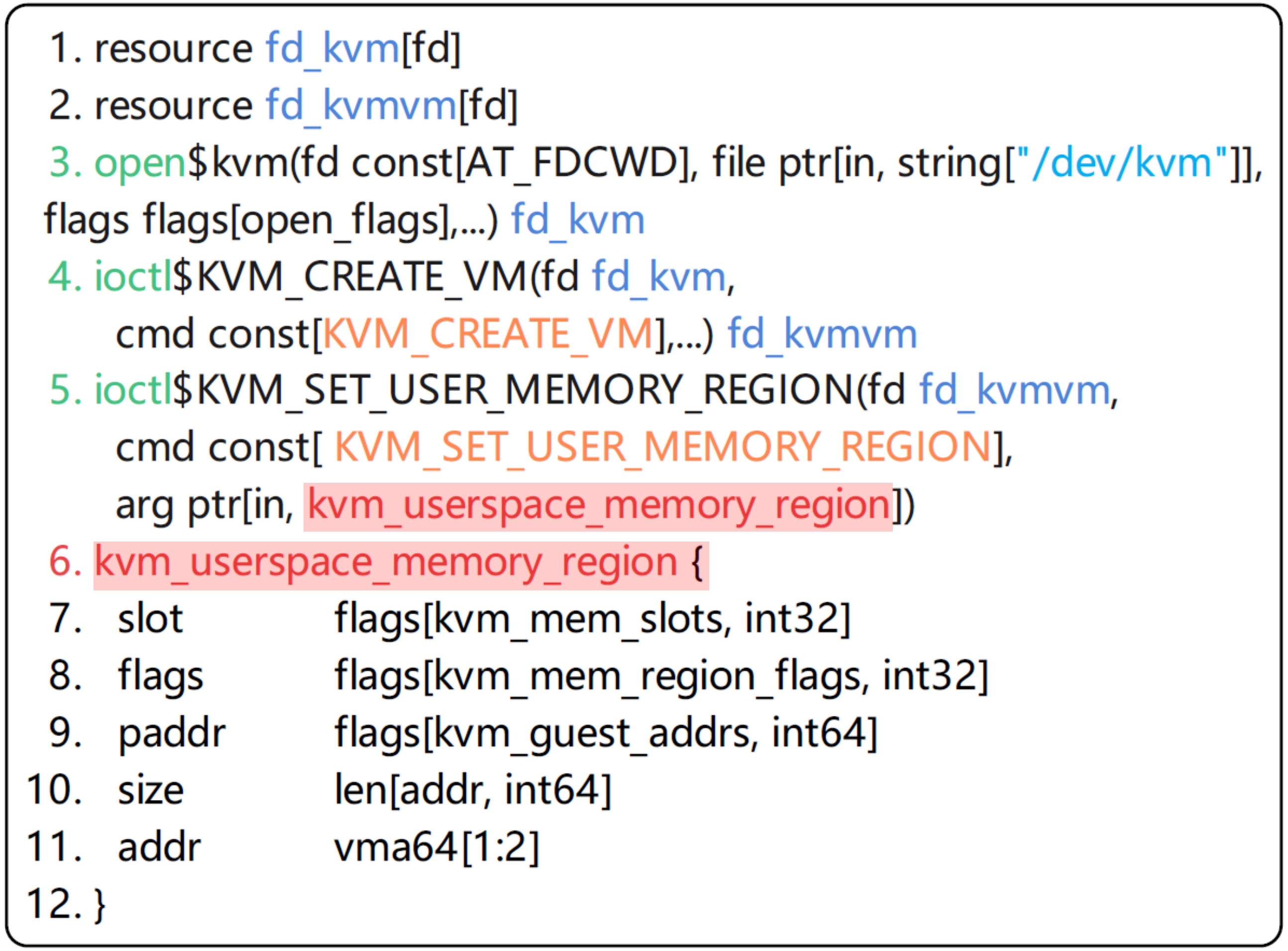


```
resource fd_kvm[fd]
resource fd_kvmvm[fd]
open$kvm(fd const[AT_FDCWD], file ptr[in, string["/dev/kvm"]],
flags flags[open_flags],...) fd_kvm
ioctl$KVM_CREATE_VM(fd fd_kvm,
    cmd const[KVM_CREATE_VM],...) fd_kvmvm
ioctl$KVM_SET_USER_MEMORY_REGION(fd fd_kvmvm,
    cmd const[ KVM_SET_USER_MEMORY_REGION],
    arg ptr[in, kvm_userspace_memory_region])
kvm_userspace_memory_region {
    slot        flags[kvm_mem_slots, int32]
    flags       flags[kvm_mem_region_flags, int32]
    paddr       flags[kvm_guest_addrs, int64]
    size        len[addr, int64]
    addr        vma64[1:2]
}
```

Syscall Descriptions

- 🟩 Syscalls interface
- 🟦 Device file name
- 🟧 Command value
- 🟥 Argument type
- 🟦 Explicit dependency

non-open file descriptor dependency:
e.g., fd_kvmvm



```
resource fd_kvm[fd]
resource fd_kvmvm[fd]
open$kvm(fd const[AT_FDCWD], file ptr[in, string["/dev/kvm"]],
flags flags[open_flags],...) fd_kvm
ioctl$KVM_CREATE_VM(fd fd_kvm,
    cmd const[KVM_CREATE_VM],...) fd_kvmvm
ioctl$KVM_SET_USER_MEMORY_REGION(fd fd_kvmvm,
    cmd const[ KVM_SET_USER_MEMORY_REGION],
    arg ptr[in, kvm_userspace_memory_region])
kvm_userspace_memory_region {
    slot          flags[kvm_mem_slots, int32]
    flags         flags[kvm_mem_region_flags, int32]
    paddr         flags[kvm_guest_addrs, int64]
    size          len[addr, int64]
    addr          vma64[1:2]
}
```

Syscall Descriptions

🟩 Syscalls interface

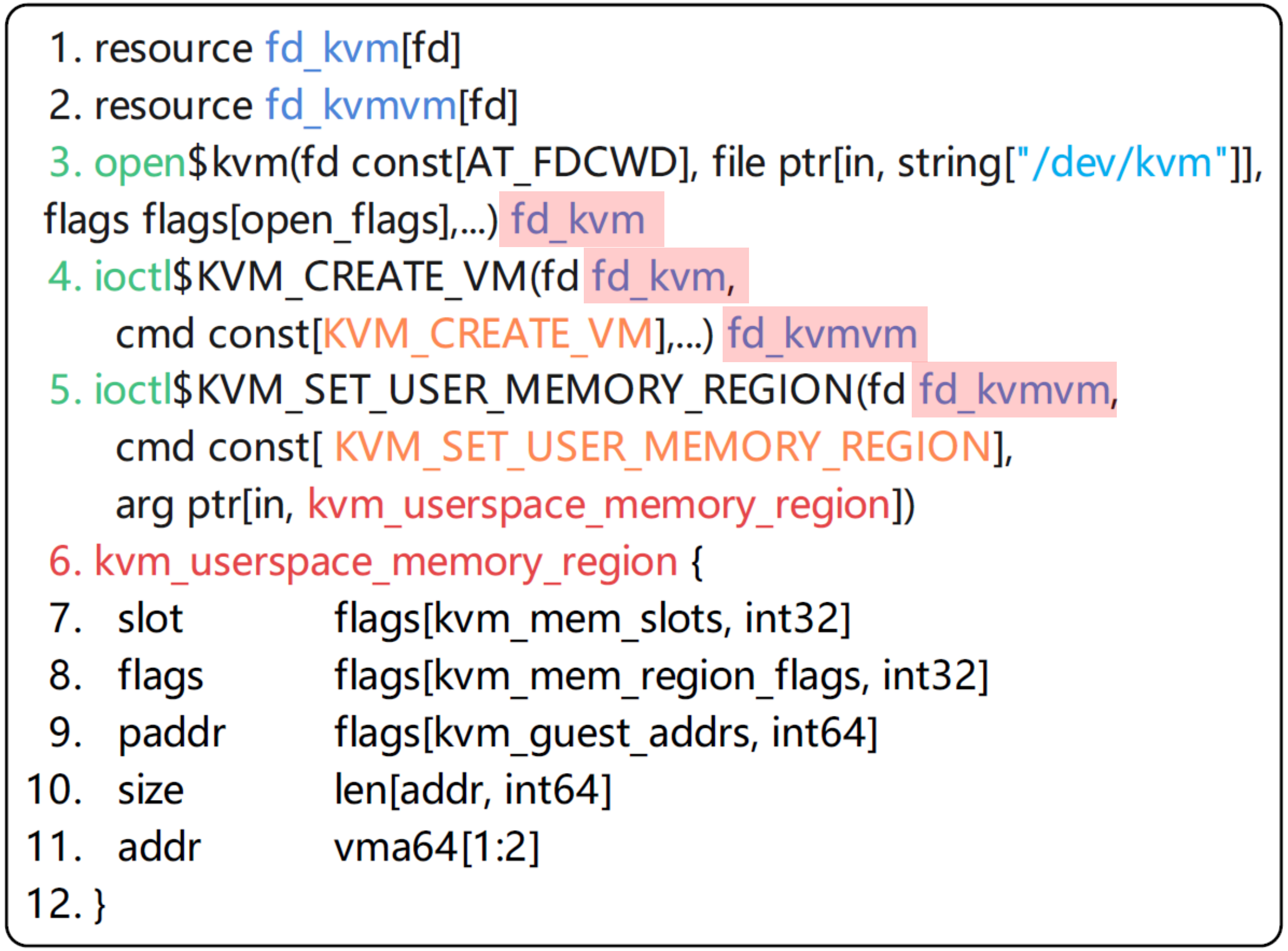
🟦 Device file name

🟧 Command value

🟥 Argument type

🟦 Explicit dependency

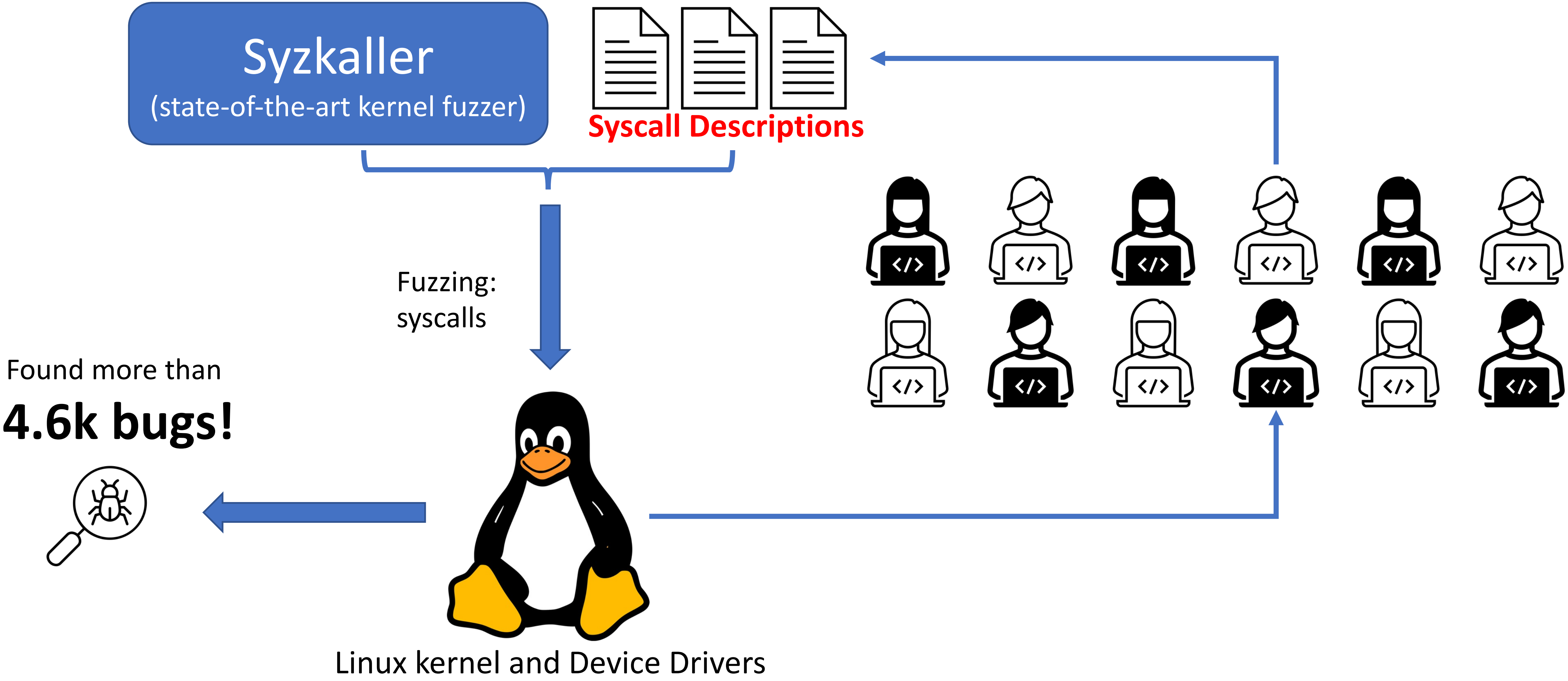
non-open file descriptor dependency:
e.g., fd_kvmvm



```
resource fd_kvm[fd]
resource fd_kvmvm[fd]
open$kvm(fd const[AT_FDCWD], file ptr[in, string["/dev/kvm"]],
flags flags[open_flags],...) fd_kvm
ioctl$KVM_CREATE_VM(fd fd_kvm,
     cmd const[KVM_CREATE_VM],...) fd_kvmvm
ioctl$KVM_SET_USER_MEMORY_REGION(fd fd_kvmvm,
     cmd const[ KVM_SET_USER_MEMORY_REGION],
     arg ptr[in, kvm_userspace_memory_region])
kvm_userspace_memory_region {
  slot         flags[kvm_mem_slots, int32]
  flags        flags[kvm_mem_region_flags, int32]
  paddr        flags[kvm_guest_addrs, int64]
  size         len[addr, int64]
  addr         vma64[1:2]
}
```

Our Goal

- **miss syscall descriptions**
- **need maintenance**



Syzkaller

(state-of-the-art kernel fuzzer)

Syscall Descriptions

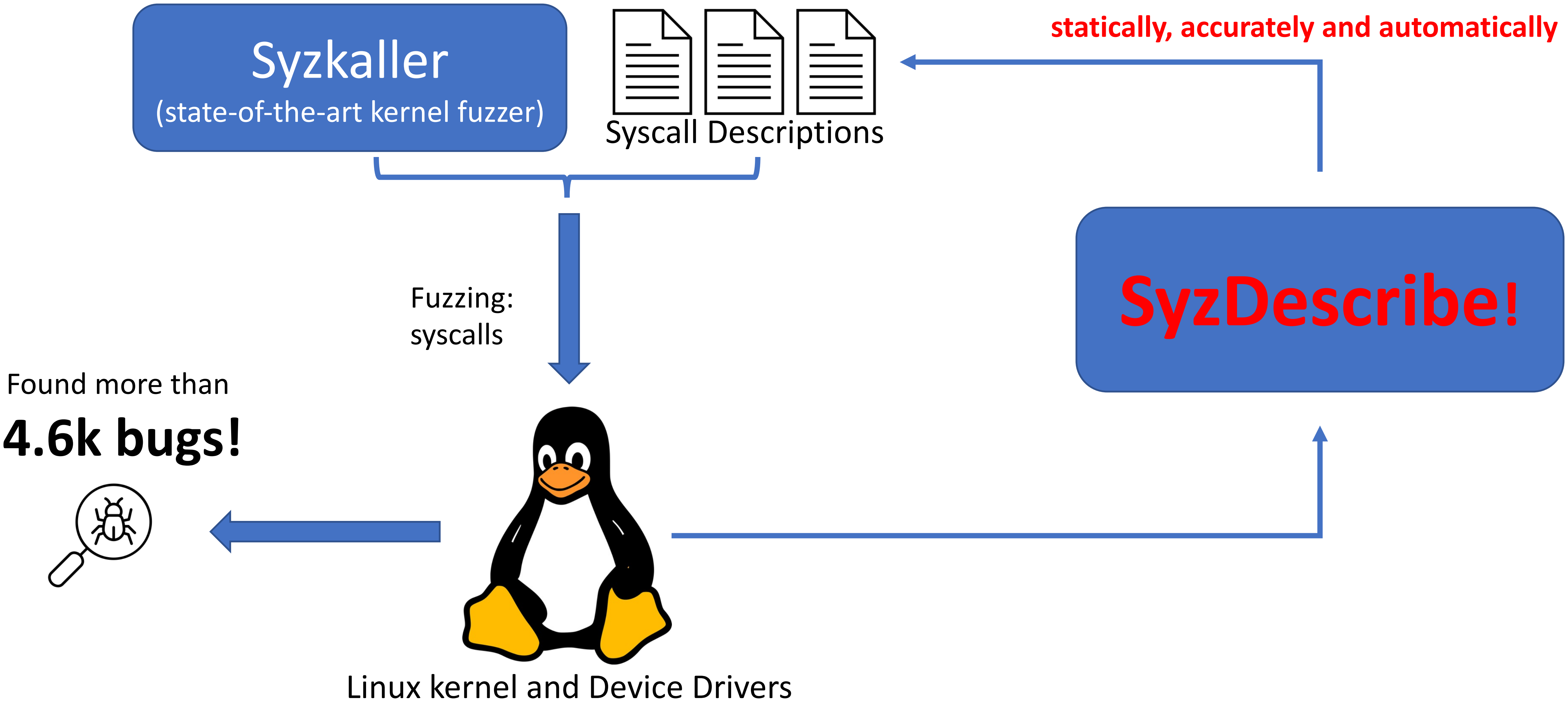
Fuzzing:
syscalls

Found more than

4.6k bugs!

Linux kernel and Device Drivers

Our Goal



Syzkaller

(state-of-the-art kernel fuzzer)

Syscall Descriptions

statically, accurately and automatically

SyzDescribe!

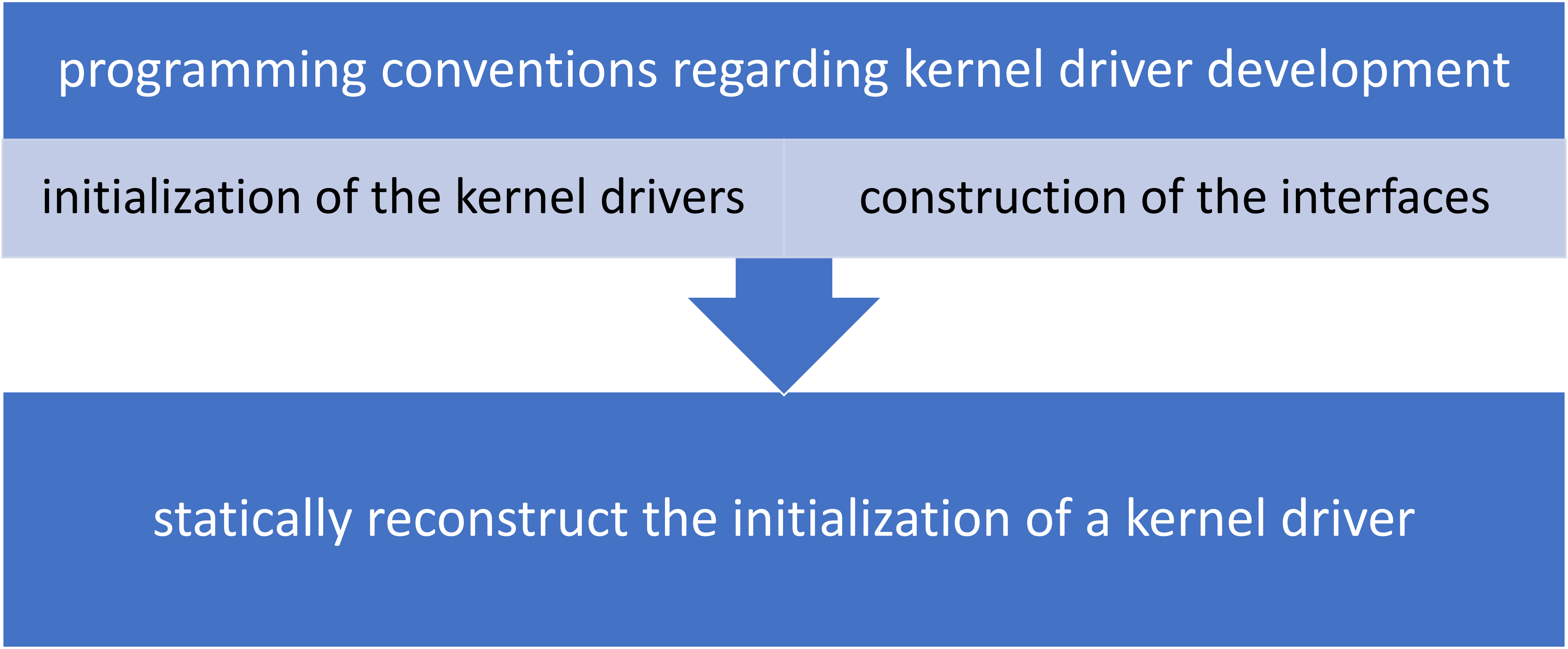
Fuzzing: syscalls

Found more than

4.6k bugs!

Linux kernel and Device Drivers

Key Insight



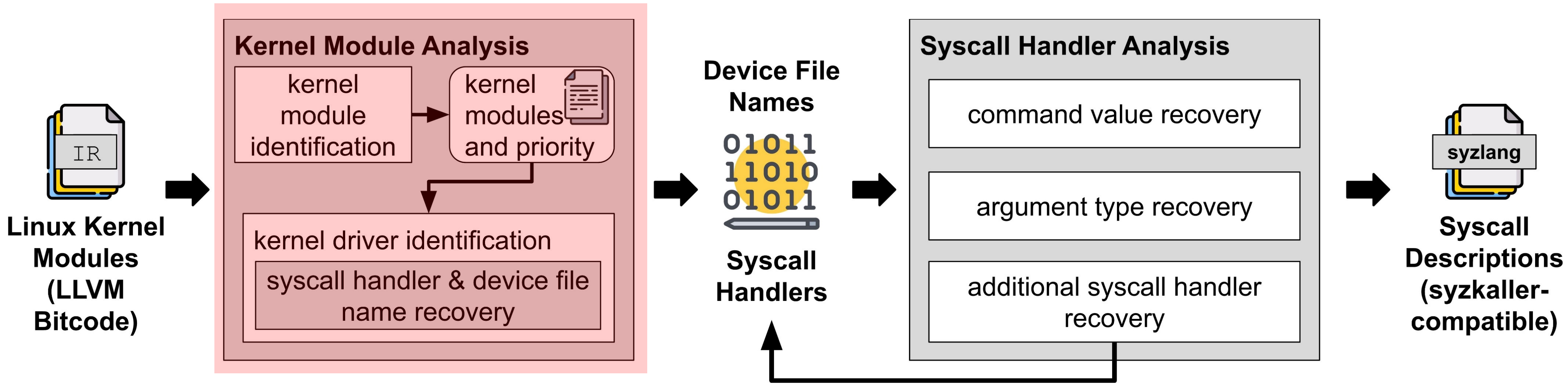
programming conventions regarding kernel driver development

initialization of the kernel drivers

construction of the interfaces

statically reconstruct the initialization of a kernel driver

SyzDescribe Design



Linux Kernel Modules (LLVM Bitcode)

Kernel Module Analysis

kernel module identification

kernel modules and priority

kernel driver identification

syscall handler & device file name recovery

Device File Names

Syscall Handlers

Syscall Handler Analysis

command value recovery

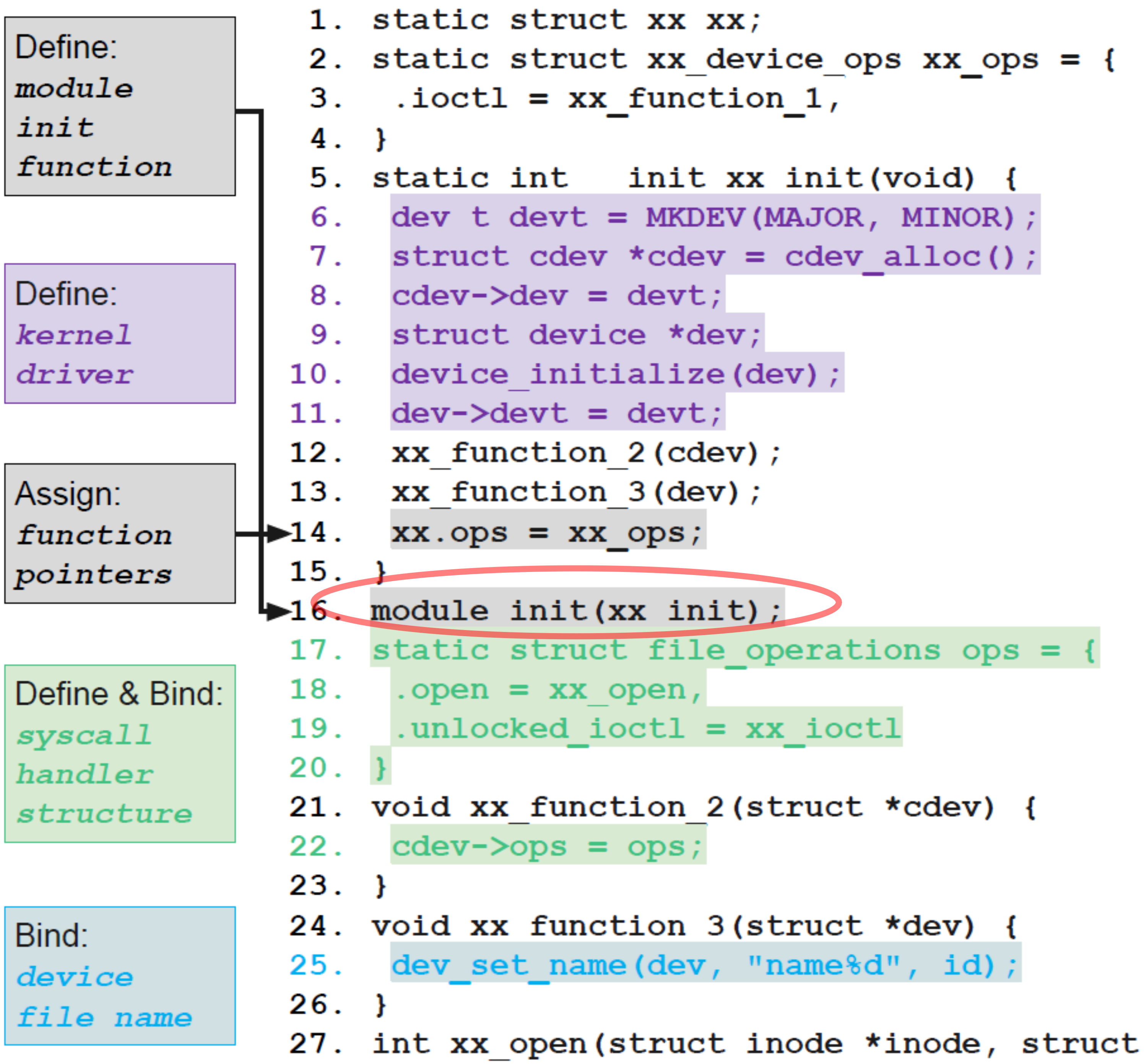
argument type recovery

additional syscall handler recovery

Syscall Descriptions (syzkaller-compatible)

Kernel Module Analysis

1. **Kernel Module Identification**
2. Kernel Driver Identification



Define: *module init function*

Define: *kernel driver*

Assign: *function pointers*

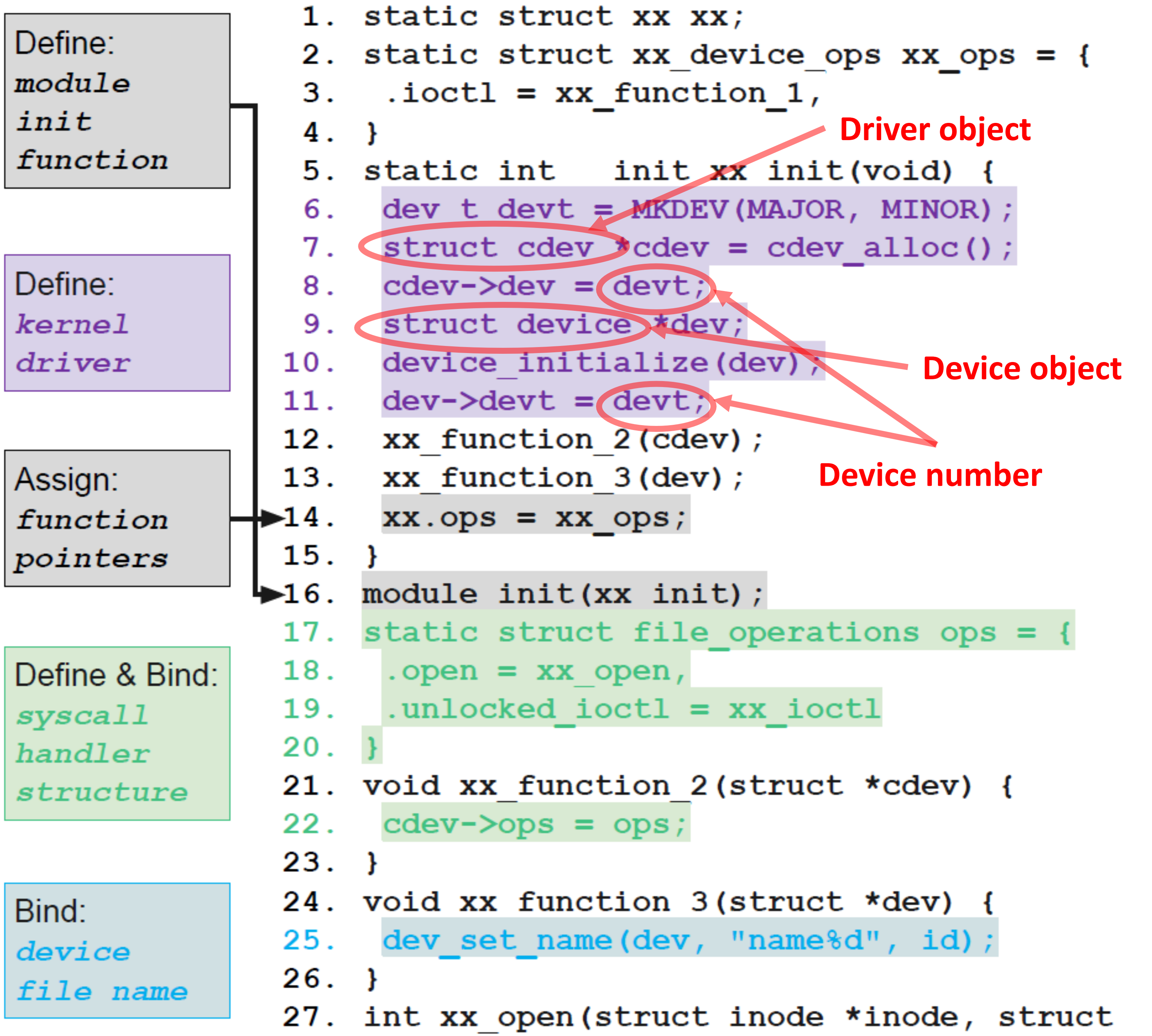
Define & Bind: *syscall handler structure*

Bind: *device file name*

```
static struct xx xx;
static struct xx_device_ops xx_ops = {
 .ioctl = xx_function_1,
}
static int   init xx init(void) {
 dev t devt = MKDEV(MAJOR, MINOR);
 struct cdev *cdev = cdev_alloc();
 cdev->dev = devt;
 struct device *dev;
 device_initialize(dev);
 dev->devt = devt;
 xx_function_2(cdev);
 xx_function_3(dev);
 xx.ops = xx_ops;
}
module init(xx init);
static struct file_operations ops = {
 .open = xx_open,
 .unlocked_ioctl = xx_ioctl
}
void xx_function_2(struct *cdev) {
 cdev->ops = ops;
}
void xx function 3(struct *dev) {
 dev_set_name(dev, "name%d", id);
}
int xx_open(struct inode *inode, struct
```

Kernel Driver Identification

1. **Driver and device object identification and pairing**
2. Syscall handler and device file name recovery



Define: *module init function*

Define: *kernel driver*

Assign: *function pointers*

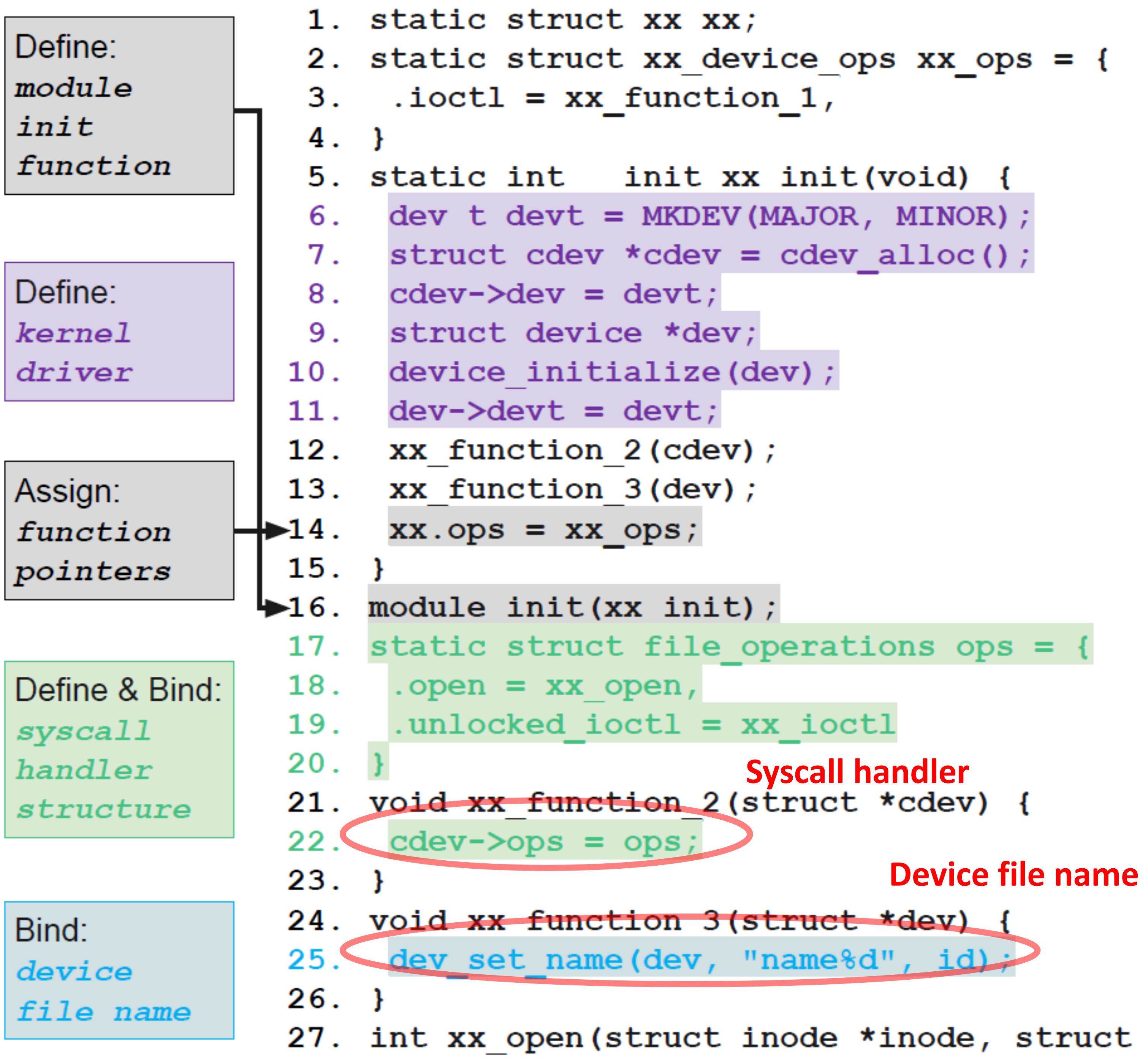
Define & Bind: *syscall handler structure*

Bind: *device file name*

```
static struct xx xx;
static struct xx_device_ops xx_ops = {
 .ioctl = xx_function_1,
}
static int   init xx init(void) {
 dev t devt = MKDEV(MAJOR, MINOR);
 struct cdev *cdev = cdev_alloc();
 cdev->dev = devt;
 struct device *dev;
 device_initialize(dev);
 dev->devt = devt;
 xx_function_2(cdev);
 xx_function_3(dev);
 xx.ops = xx_ops;
}
module init(xx init);
static struct file_operations ops = {
 .open = xx_open,
 .unlocked_ioctl = xx_ioctl
}
void xx_function_2(struct *cdev) {
 cdev->ops = ops;
}
void xx function 3(struct *dev) {
 dev_set_name(dev, "name%d", id);
}
int xx_open(struct inode *inode, struct
```

Kernel Driver Identification

1. Driver and device object identification and pairing
2. **Syscall handler and device file name recovery**



Define: *module init function*

Define: *kernel driver*

Assign: *function pointers*

Define & Bind: *syscall handler structure*

Bind: *device file name*

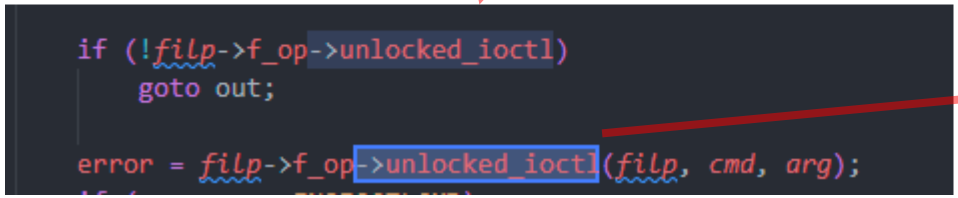
```
static struct xx xx;
static struct xx_device_ops xx_ops = {
 .ioctl = xx_function_1,
}
static int   init xx init(void) {
 dev t devt = MKDEV(MAJOR, MINOR);
 struct cdev *cdev = cdev_alloc();
 cdev->dev = devt;
 struct device *dev;
 device_initialize(dev);
 dev->devt = devt;
 xx_function_2(cdev);
 xx_function_3(dev);
 xx.ops = xx_ops;
}
module init(xx init);
static struct file_operations ops = {
 .open = xx_open,
 .unlocked_ioctl = xx_ioctl
}
void xx_function_2(struct *cdev) {
 cdev->ops = ops;
}
void xx_function_3(struct *dev) {
 dev_set_name(dev, "name%d", id);
}
int xx_open(struct inode *inode, struct
```

Syscall handler

Device file name

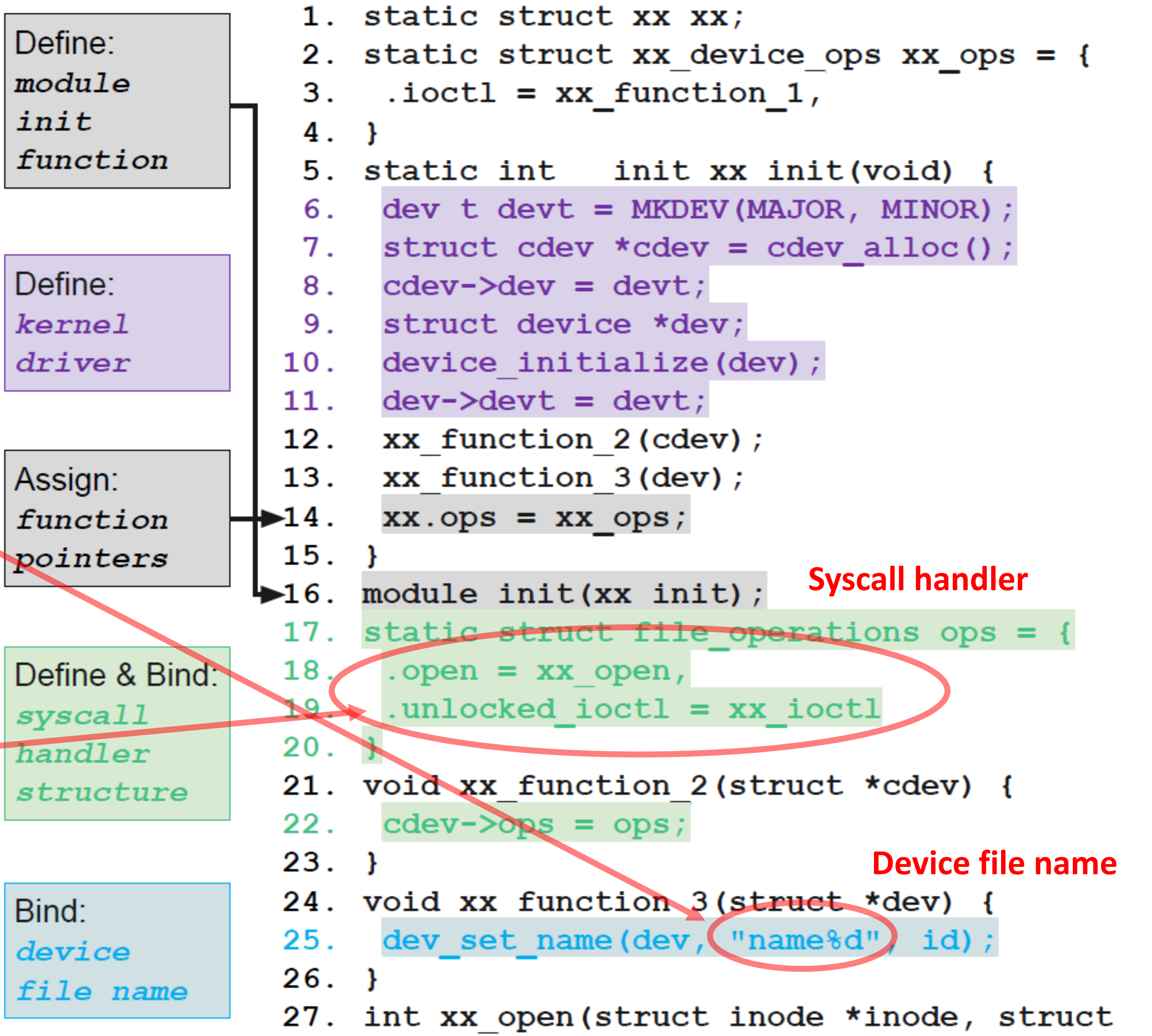
Kernel Driver Identification

- fd = open(“device file name”)
- Ioctl(fd, cmd, arg)



```
if (!filp->f_op->unlocked_ioctl)
    goto out;

error = filp->f_op->unlocked_ioctl(filp, cmd, arg);
```



Define: *module init function*

Define: *kernel driver*

Assign: *function pointers*

Define & Bind: *syscall handler structure*

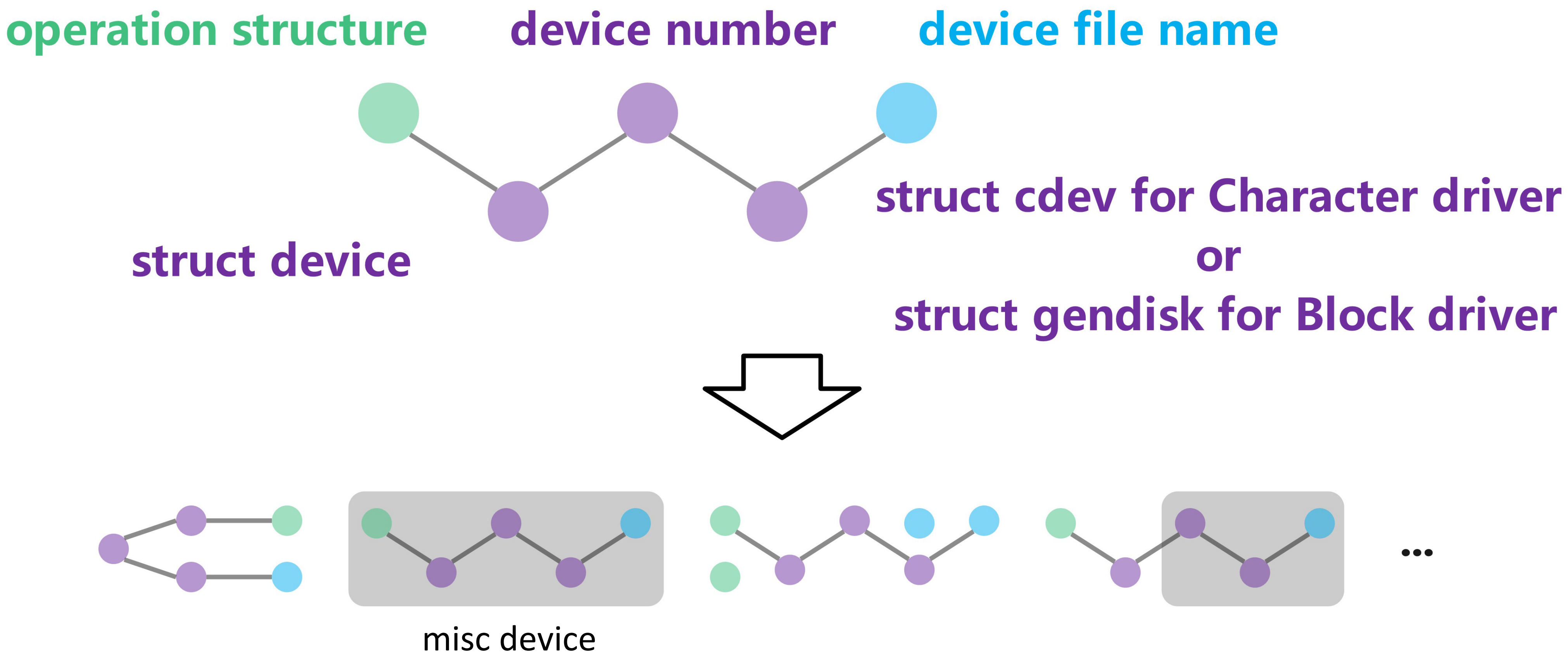
Bind: *device file name*

```
static struct xx xx;
static struct xx_device_ops xx_ops = {
 .ioctl = xx_function_1,
}
static int __init xx_init(void) {
 dev_t devt = MKDEV(MAJOR, MINOR);
 struct cdev *cdev = cdev_alloc();
 cdev->dev = devt;
 struct device *dev;
 device_initialize(dev);
 dev->devt = devt;
 xx_function_2(cdev);
 xx_function_3(dev);
 xx.ops = xx_ops;
}
module_init(xx_init);
static struct file_operations ops = {
 .open = xx_open,
 .unlocked_ioctl = xx_ioctl
}
void xx_function_2(struct *cdev) {
 cdev->ops = ops;
}
void xx_function_3(struct *dev) {
 dev_set_name(dev, "name%d", id);
}
int xx_open(struct inode *inode, struct
```

Syscall handler

Device file name

Kernel Driver Identification



operation structure

device number

device file name

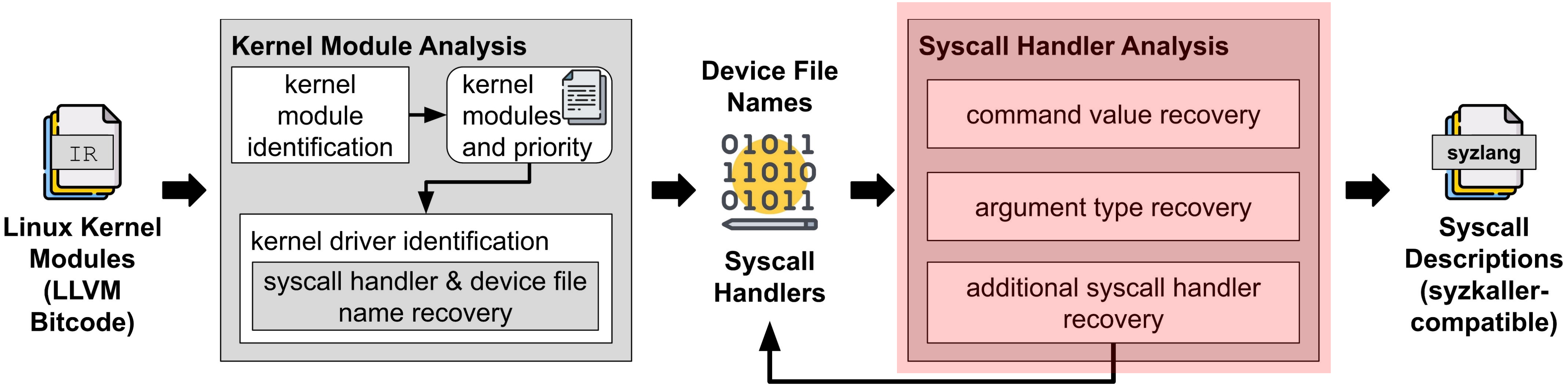
struct device

struct cdev for Character driver
or
struct gendisk for Block driver

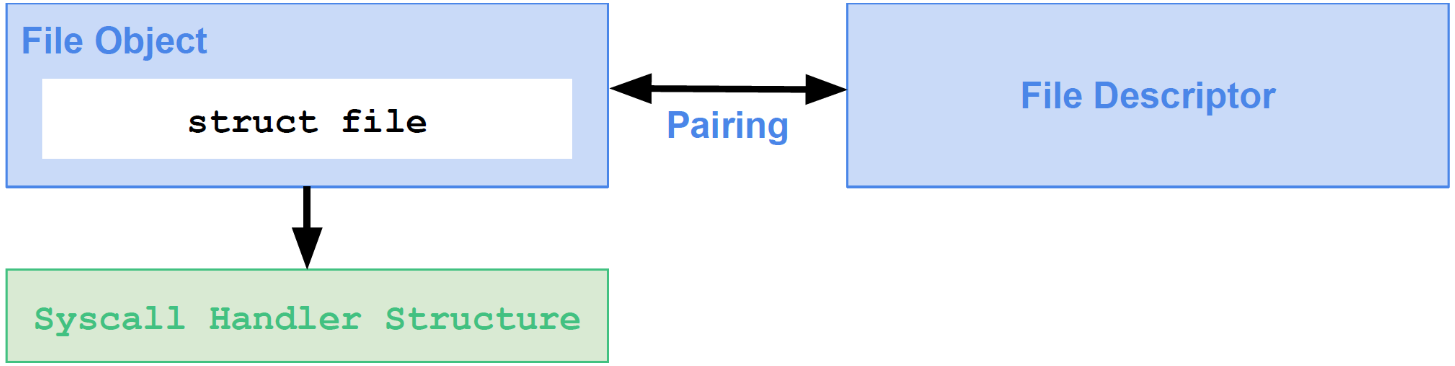
misc device

...

SyzDescribe Design


IR
Linux Kernel Modules (LLVM Bitcode)
Kernel Module Analysis
kernel module identification
kernel modules and priority
kernel driver identification
syscall handler & device file name recovery
Device File Names
01011
11010
01011
Syscall Handlers
Syscall Handler Analysis
command value recovery
argument type recovery
additional syscall handler recovery
syzlang
Syscall Descriptions (syzkaller-compatible)

Syscall Handler Analysis

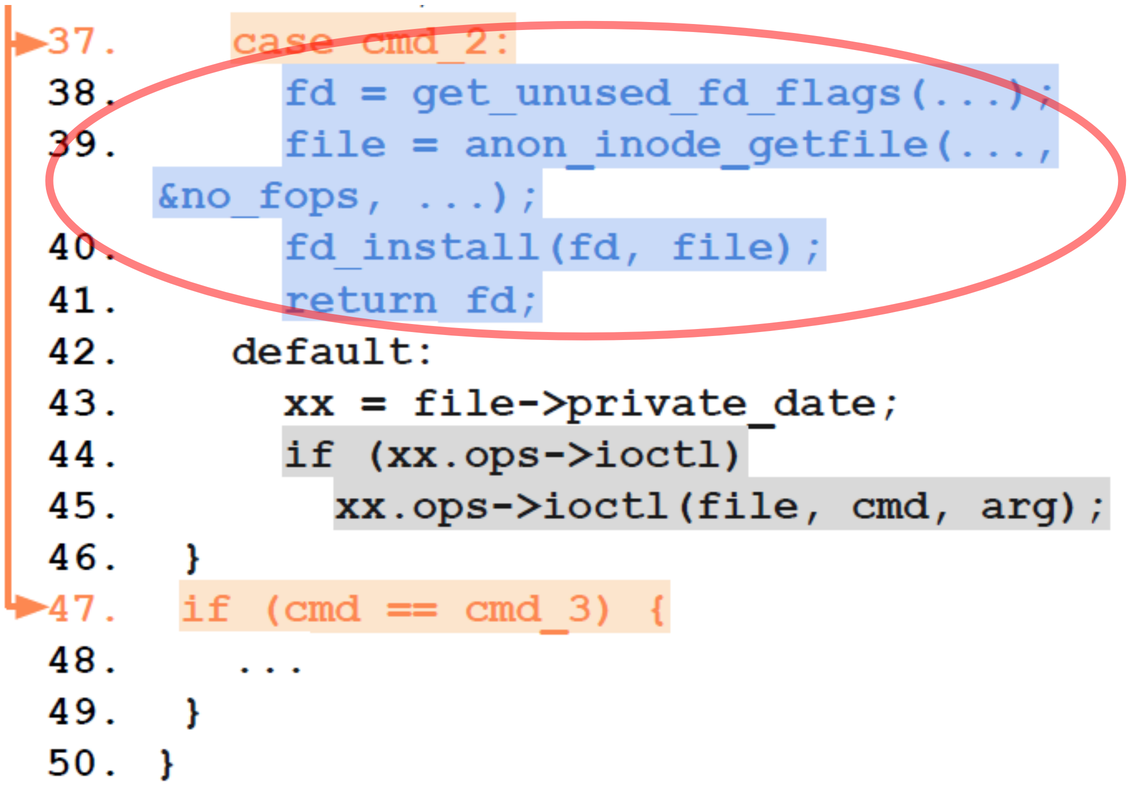


1. Command Value Recovery
2. Argument Type Recovery
3. **Additional Syscall Handler Recovery**

(non-open file descriptor dependency)

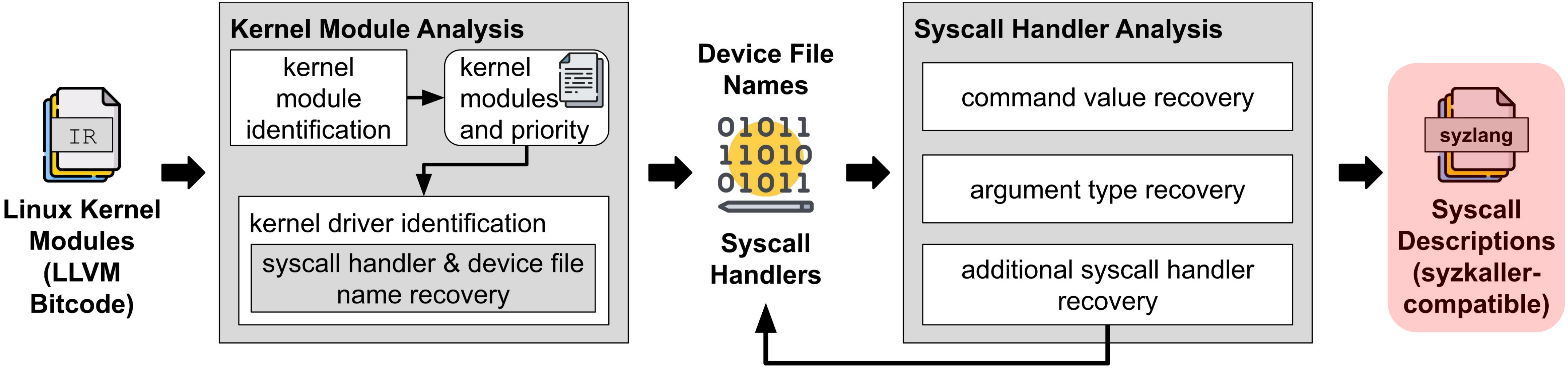
Define: **additional** ***syscall handler***

Use: ***indirect function call***



```
    case cmd_2:
      fd = get_unused_fd_flags(...);
      file = anon_inode_getfile(...,
&no_fops, ...);
      fd_install(fd, file);
      return fd;
    default:
      xx = file->private_date;
      if (xx.ops->ioctl)
        xx.ops->ioctl(file, cmd, arg);
  }
  if (cmd == cmd_3) {
    ...
  }
}
```

SyzDescribe Design



Linux Kernel Modules (LLVM Bitcode)

Kernel Module Analysis

kernel module identification

kernel modules and priority

kernel driver identification

syscall handler & device file name recovery

Device File Names

Syscall Handlers

Syscall Handler Analysis

command value recovery

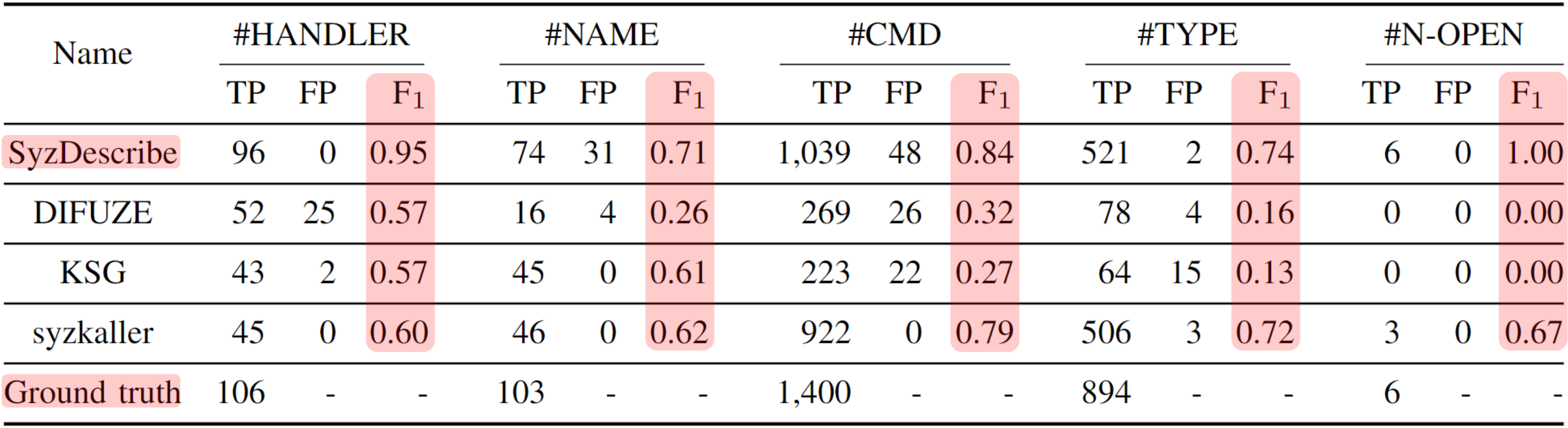
argument type recovery

additional syscall handler recovery

Syscall Descriptions (syzkaller-compatible)

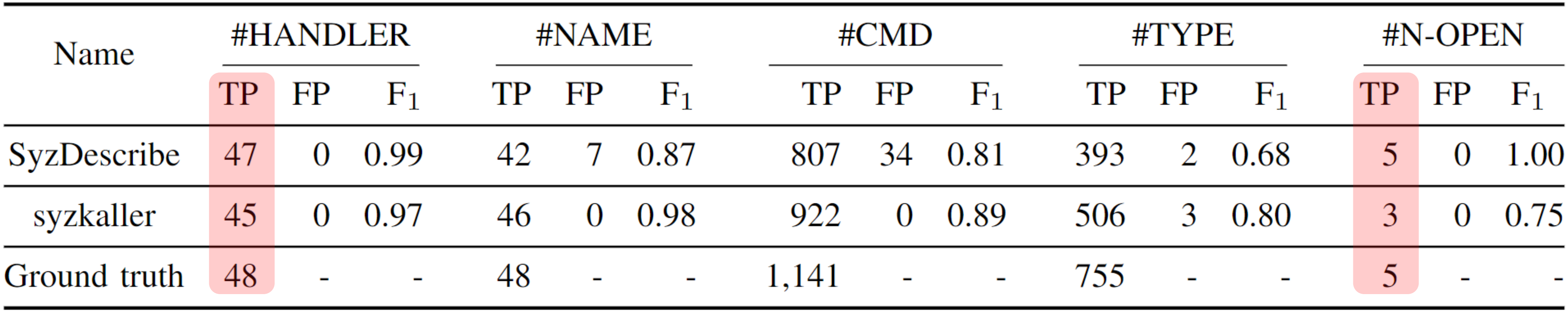
SyzDescribe vs. Ground Truth (Final Goal) : Accuracy

- 100 drivers: randomly picked and cross-validate
- **Closest to the ground truth compared against any other solutions**



Name	#HANDLER			#NAME			#CMD			#TYPE			#N-OPEN		
	TP	FP	F_1	TP	FP	F_1	TP	FP	F_1	TP	FP	F_1	TP	FP	F_1
SyzDescribe	96	0	0.95	74	31	0.71	1,039	48	0.84	521	2	0.74	6	0	1.00
DIFUZE	52	25	0.57	16	4	0.26	269	26	0.32	78	4	0.16	0	0	0.00
KSG	43	2	0.57	45	0	0.61	223	22	0.27	64	15	0.13	0	0	0.00
syzkaller	45	0	0.60	46	0	0.62	922	0	0.79	506	3	0.72	3	0	0.67
Ground truth	106	-	-	103	-	-	1,400	-	-	894	-	-	6	-	-

SyzDescribe vs. Syzkaller Description (Manual)



Name	#HANDLER			#NAME			#CMD			#TYPE			#N-OPEN		
	TP	FP	F_1	TP	FP	F_1	TP	FP	F_1	TP	FP	F_1	TP	FP	F_1
SyzDescribe	47	0	0.99	42	7	0.87	807	34	0.81	393	2	0.68	5	0	1.00
syzkaller	45	0	0.97	46	0	0.98	922	0	0.89	506	3	0.80	3	0	0.75
Ground truth	48	-	-	48	-	-	1,141	-	-	755	-	-	5	-	-

- 57 drivers: missed in the syzkaller descriptions
- 43 drivers: covered by syzkaller descriptions
- 2 non-open dependency: still missed in the syzkaller descriptions
- 13 drivers: more CMDs or TYPEs generated by SyzDescribe

“Bugs” in Human-Generated Descriptions

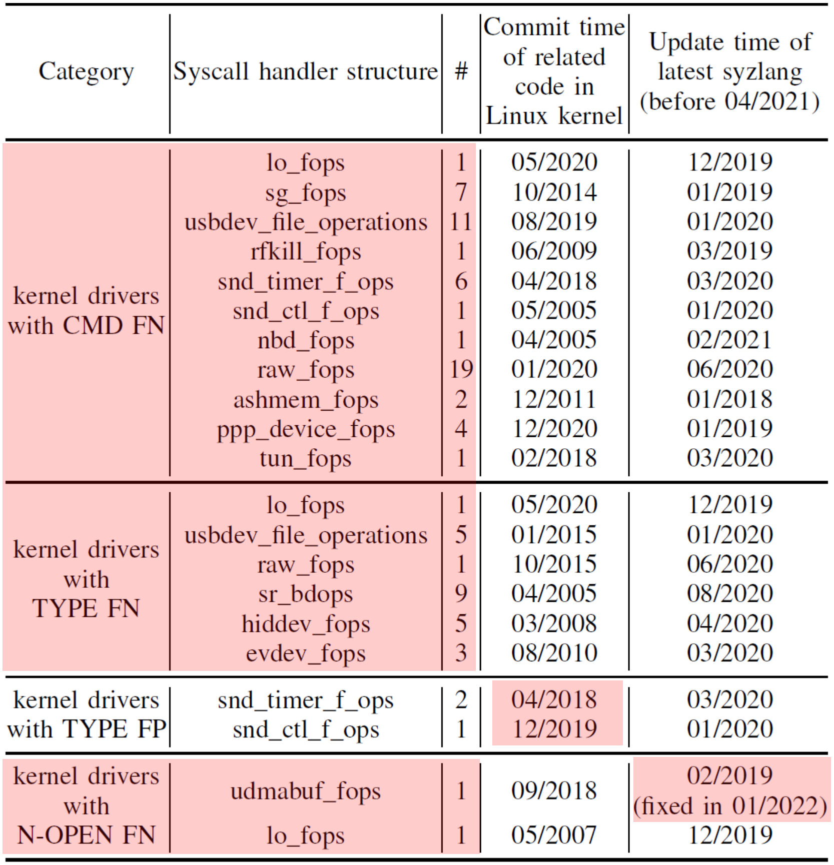
- **78 missed command values or argument types**
- **Two FN of the additional syscall handlers**
- **FP mainly because of evolution of the kernel code**
- **Only one (i.e., udmabuf_fops) of these “bugs” is fixed in January 2022.**

  ➔Ongoing maintenance is needed.

- FN after human experts updated
- These “bugs” have existing for a long time.

  ➔Human experts are not enough.

- We have reported all the bugs to syzkaller (all of which are fixed).



Category	Syscall handler structure	#	Commit time of related code in Linux kernel	Update time of latest syzlang (before 04/2021)
kernel drivers with CMD FN	lo_fops	1	05/2020	12/2019
	sg_fops	7	10/2014	01/2019
	usbdev_file_operations	11	08/2019	01/2020
	rfkill_fops	1	06/2009	03/2019
	snd_timer_f_ops	6	04/2018	03/2020
	snd_ctl_f_ops	1	05/2005	01/2020
	nbd_fops	1	04/2005	02/2021
	raw_fops	19	01/2020	06/2020
	ashmem_fops	2	12/2011	01/2018
	ppp_device_fops	4	12/2020	01/2019
	tun_fops	1	02/2018	03/2020
kernel drivers with TYPE FN	lo_fops	1	05/2020	12/2019
	usbdev_file_operations	5	01/2015	01/2020
	raw_fops	1	10/2015	06/2020
	sr_bdops	9	04/2005	08/2020
	hiddev_fops	5	03/2008	04/2020
	evdev_fops	3	08/2010	03/2020
kernel drivers with TYPE FP	snd_timer_f_ops	2	04/2018	03/2020
	snd_ctl_f_ops	1	12/2019	01/2020
kernel drivers with N-OPEN FN	udmabuf_fops	1	09/2018	02/2019 (fixed in 01/2022)
	lo_fops	1	05/2007	12/2019

“Bugs” in Human-Generated Descriptions

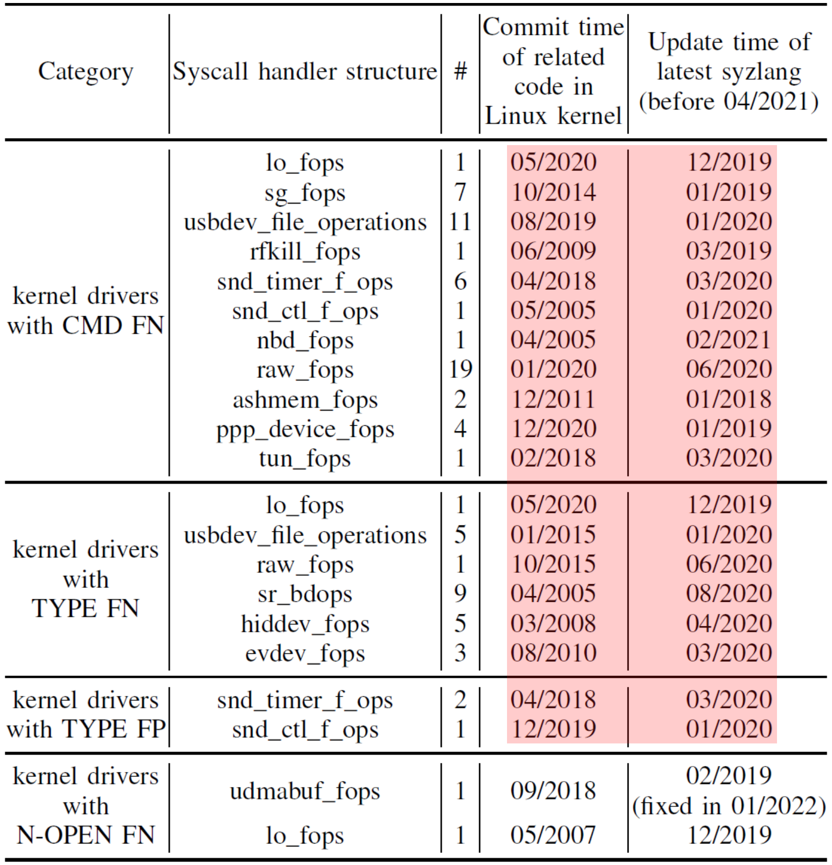
- 78 missed command values or argument types
- Two FN of the additional syscall handlers
- FP mainly because of evolution of the kernel code
- Only one (i.e., udmabuf_fops) of these “bugs” is fixed in January 2022.

  ➔Ongoing maintenance is needed.

- **FN after human experts updated**
- **These “bugs” have existing for a long time.**

  ➔Human experts are not enough.

- We have reported all the bugs to syzkaller (all of which are fixed).



Category	Syscall handler structure	#	Commit time of related code in Linux kernel	Update time of latest syzlang (before 04/2021)
kernel drivers with CMD FN	lo_fops	1	05/2020	12/2019
	sg_fops	7	10/2014	01/2019
	usbdev_file_operations	11	08/2019	01/2020
	rfkill_fops	1	06/2009	03/2019
	snd_timer_f_ops	6	04/2018	03/2020
	snd_ctl_f_ops	1	05/2005	01/2020
	nbd_fops	1	04/2005	02/2021
	raw_fops	19	01/2020	06/2020
	ashmem_fops	2	12/2011	01/2018
	ppp_device_fops	4	12/2020	01/2019
	tun_fops	1	02/2018	03/2020
kernel drivers with TYPE FN	lo_fops	1	05/2020	12/2019
	usbdev_file_operations	5	01/2015	01/2020
	raw_fops	1	10/2015	06/2020
	sr_bdops	9	04/2005	08/2020
	hiddev_fops	5	03/2008	04/2020
	evdev_fops	3	08/2010	03/2020
kernel drivers with TYPE FP	snd_timer_f_ops	2	04/2018	03/2020
	snd_ctl_f_ops	1	12/2019	01/2020
kernel drivers with N-OPEN FN	udmabuf_fops	1	09/2018	02/2019 (fixed in 01/2022)
	lo_fops	1	05/2007	12/2019

“Bugs” in Human-Generated Descriptions

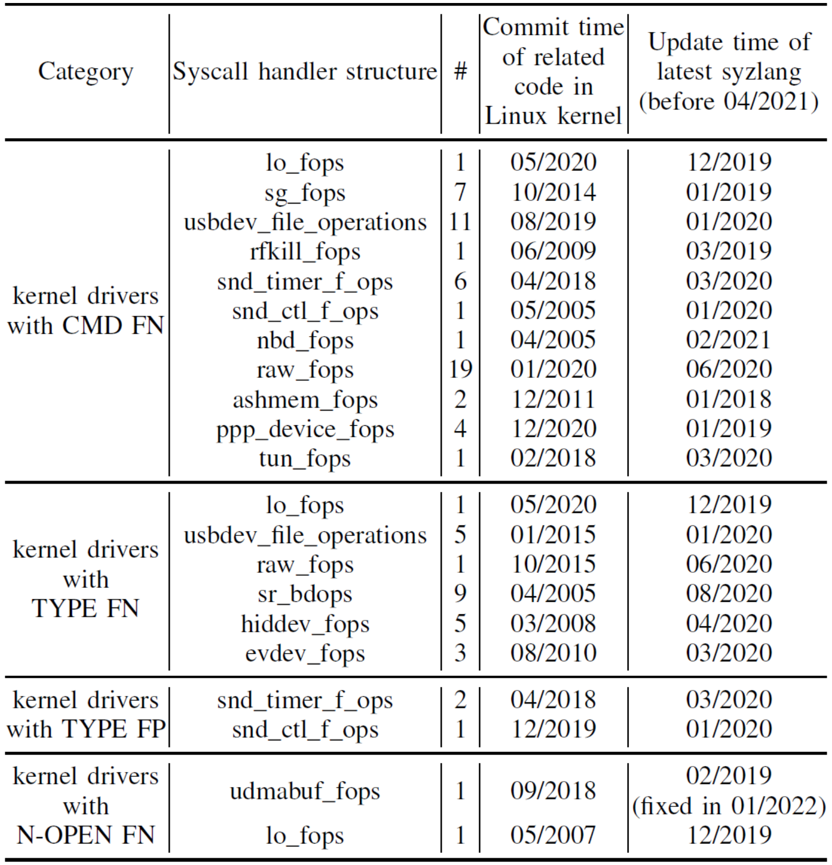
- 78 missed command values or argument types
- Two FN of the additional syscall handlers
- FP mainly because of evolution of the kernel code
- Only one (i.e., udmabuf_fops) of these “bugs” is fixed in January 2022.

  ➔Ongoing maintenance is needed.

- FN after human experts updated
- These “bugs” have existing for a long time.

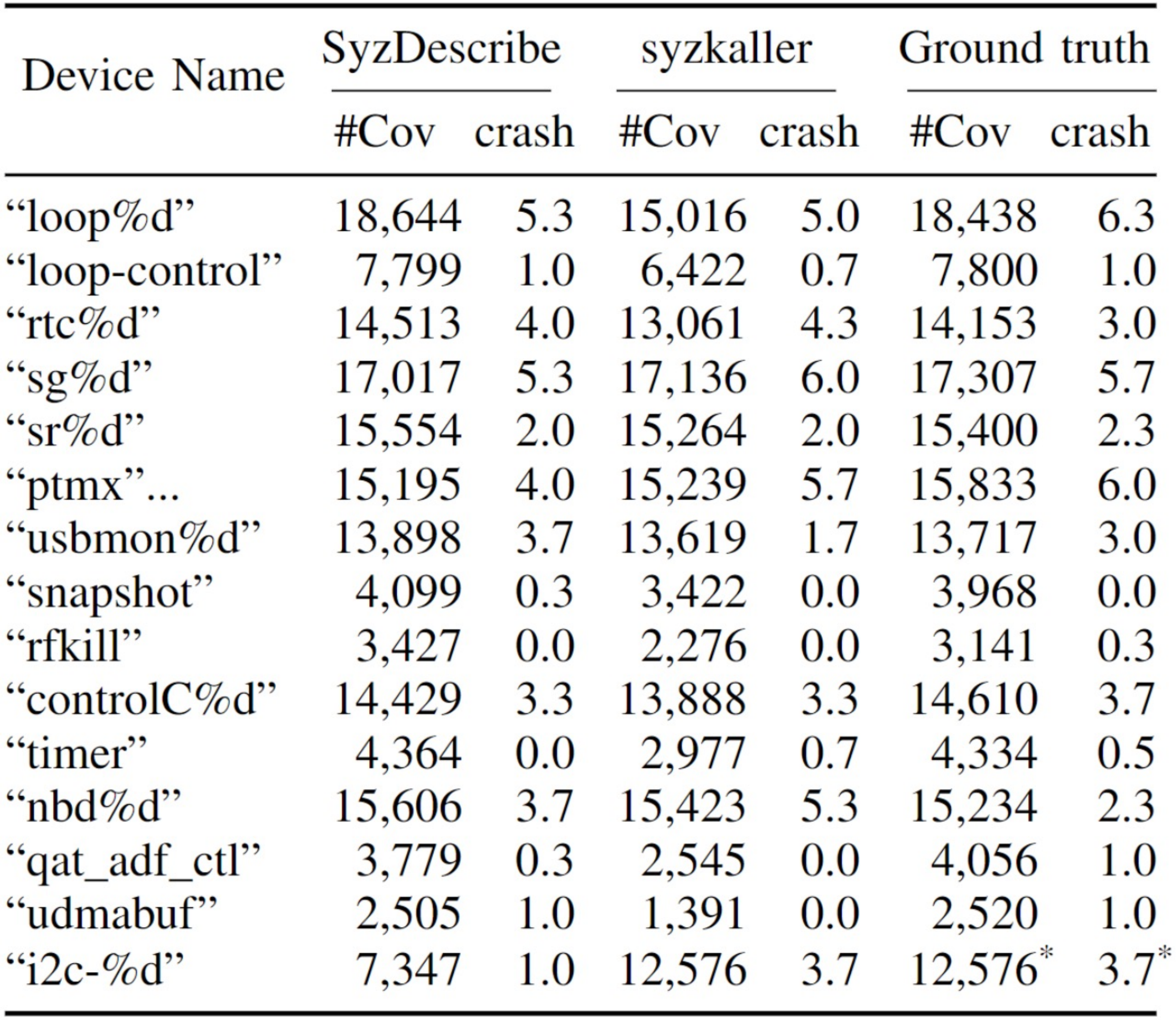
  ➔Human experts are not enough.

- **We have reported all the bugs to syzkaller (all of which are fixed).**

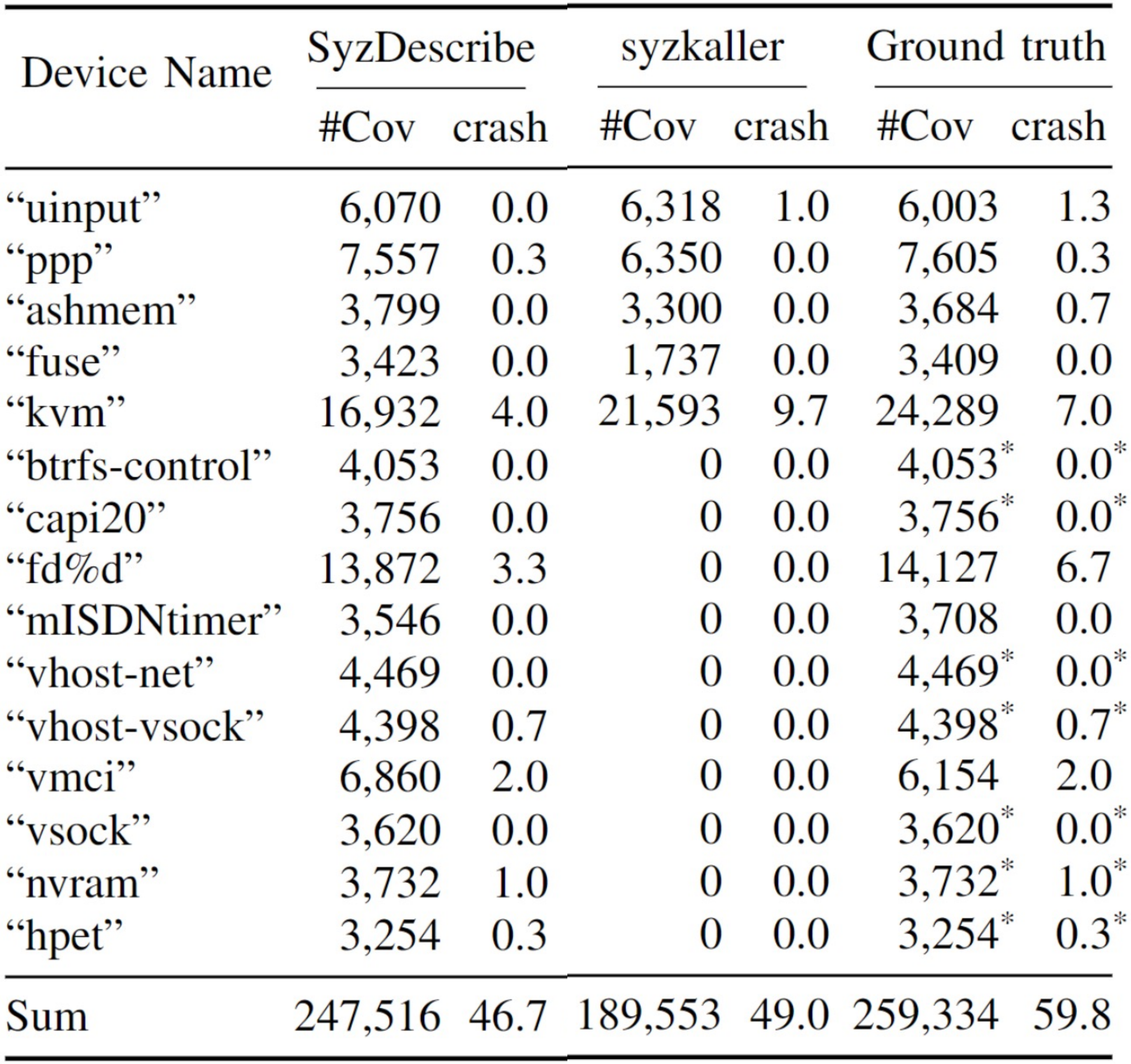


Category	Syscall handler structure	#	Commit time of related code in Linux kernel	Update time of latest syzlang (before 04/2021)
kernel drivers with CMD FN	lo_fops	1	05/2020	12/2019
	sg_fops	7	10/2014	01/2019
	usbdev_file_operations	11	08/2019	01/2020
	rfkill_fops	1	06/2009	03/2019
	snd_timer_f_ops	6	04/2018	03/2020
	snd_ctl_f_ops	1	05/2005	01/2020
	nbd_fops	1	04/2005	02/2021
	raw_fops	19	01/2020	06/2020
	ashmem_fops	2	12/2011	01/2018
	ppp_device_fops	4	12/2020	01/2019
	tun_fops	1	02/2018	03/2020
kernel drivers with TYPE FN	lo_fops	1	05/2020	12/2019
	usbdev_file_operations	5	01/2015	01/2020
	raw_fops	1	10/2015	06/2020
	sr_bdops	9	04/2005	08/2020
	hiddev_fops	5	03/2008	04/2020
	evdev_fops	3	08/2010	03/2020
kernel drivers with TYPE FP	snd_timer_f_ops	2	04/2018	03/2020
	snd_ctl_f_ops	1	12/2019	01/2020
kernel drivers with N-OPEN FN	udmabuf_fops	1	09/2018	02/2019 (fixed in 01/2022)
	lo_fops	1	05/2007	12/2019

SyzDescribe vs. Syzkaller Description vs. Ground Truth: Fuzzing



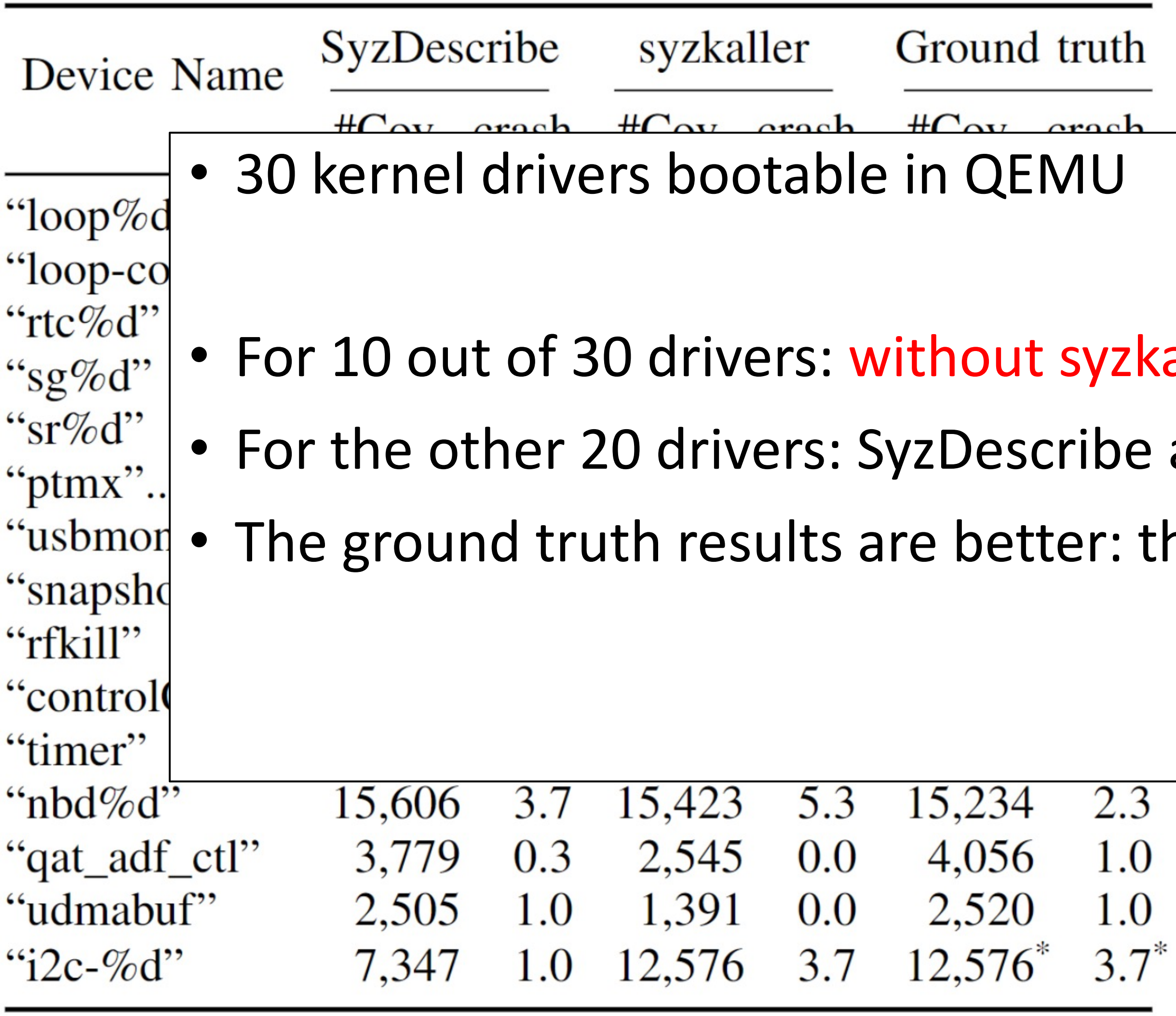
Device Name	SyzDescribe		syzkaller		Ground truth	
	#Cov	crash	#Cov	crash	#Cov	crash
“loop%d”	18,644	5.3	15,016	5.0	18,438	6.3
“loop-control”	7,799	1.0	6,422	0.7	7,800	1.0
“rtc%d”	14,513	4.0	13,061	4.3	14,153	3.0
“sg%d”	17,017	5.3	17,136	6.0	17,307	5.7
“sr%d”	15,554	2.0	15,264	2.0	15,400	2.3
“ptmx”...	15,195	4.0	15,239	5.7	15,833	6.0
“usbmon%d”	13,898	3.7	13,619	1.7	13,717	3.0
“snapshot”	4,099	0.3	3,422	0.0	3,968	0.0
“rfkill”	3,427	0.0	2,276	0.0	3,141	0.3
“controlC%d”	14,429	3.3	13,888	3.3	14,610	3.7
“timer”	4,364	0.0	2,977	0.7	4,334	0.5
“nbd%d”	15,606	3.7	15,423	5.3	15,234	2.3
“qat_adf_ctl”	3,779	0.3	2,545	0.0	4,056	1.0
“udmabuf”	2,505	1.0	1,391	0.0	2,520	1.0
“i2c-%d”	7,347	1.0	12,576	3.7	12,576*	3.7*



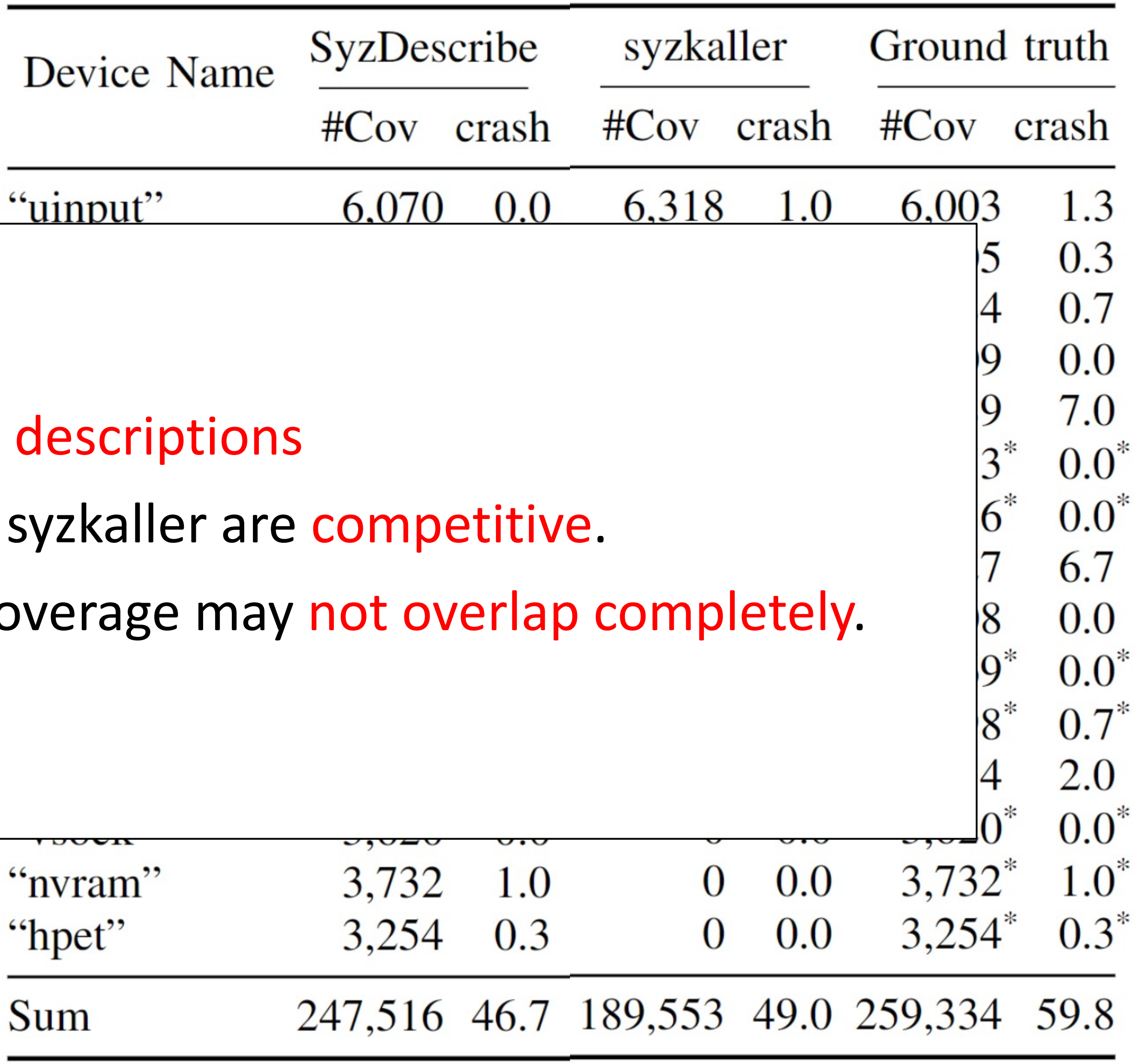
Device Name	SyzDescribe		syzkaller		Ground truth	
	#Cov	crash	#Cov	crash	#Cov	crash
“uinput”	6,070	0.0	6,318	1.0	6,003	1.3
“ppp”	7,557	0.3	6,350	0.0	7,605	0.3
“ashmem”	3,799	0.0	3,300	0.0	3,684	0.7
“fuse”	3,423	0.0	1,737	0.0	3,409	0.0
“kvm”	16,932	4.0	21,593	9.7	24,289	7.0
“btrfs-control”	4,053	0.0	0	0.0	4,053*	0.0*
“capi20”	3,756	0.0	0	0.0	3,756*	0.0*
“fd%d”	13,872	3.3	0	0.0	14,127	6.7
“mISDNtimer”	3,546	0.0	0	0.0	3,708	0.0
“vhost-net”	4,469	0.0	0	0.0	4,469*	0.0*
“vhost-vsock”	4,398	0.7	0	0.0	4,398*	0.7*
“vmci”	6,860	2.0	0	0.0	6,154	2.0
“vsock”	3,620	0.0	0	0.0	3,620*	0.0*
“nvram”	3,732	1.0	0	0.0	3,732*	1.0*
“hpet”	3,254	0.3	0	0.0	3,254*	0.3*
Sum	247,516	46.7	189,553	49.0	259,334	59.8

SyzDescribe vs. Syzkaller Description vs. Ground Truth: Fuzzing

- 30 kernel drivers bootable in QEMU
- For 10 out of 30 drivers: without syzkaller descriptions
- For the other 20 drivers: SyzDescribe and syzkaller are competitive.
- The ground truth results are better: the coverage may not overlap completely.



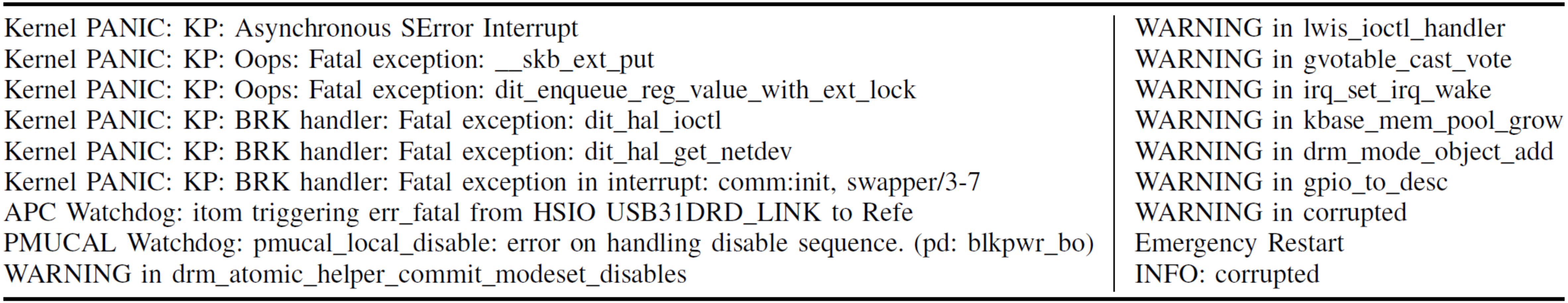
Device Name	SyzDescribe		syzkaller		Ground truth	
	#Cov	crash	#Cov	crash	#Cov	crash
"loop%d						
"loop-co						
"rtc%d"						
"sg%d"						
"sr%d"						
"ptmx"..						
"usbmon						
"snapsho						
"rfkill"						
"control						
"timer"						
"nbd%d"	15,606	3.7	15,423	5.3	15,234	2.3
"qat_adf_ctl"	3,779	0.3	2,545	0.0	4,056	1.0
"udmabuf"	2,505	1.0	1,391	0.0	2,520	1.0
"i2c-%d"	7,347	1.0	12,576	3.7	12,576*	3.7*



Device Name	SyzDescribe		syzkaller		Ground truth	
	#Cov	crash	#Cov	crash	#Cov	crash
"uinput"	6,070	0.0	6,318	1.0	6,003	1.3
					5	0.3
					4	0.7
					9	0.0
					9	7.0
					3*	0.0*
					6*	0.0*
					7	6.7
					8	0.0
					9*	0.0*
					8*	0.7*
					4	2.0
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0*	0.0*
"nvram"	3,732	1.0	0	0.0	3,732*	1.0*
"hpet"	3,254	0.3	0	0.0	3,254*	0.3*
Sum	247,516	46.7	189,553	49.0	259,334	59.8

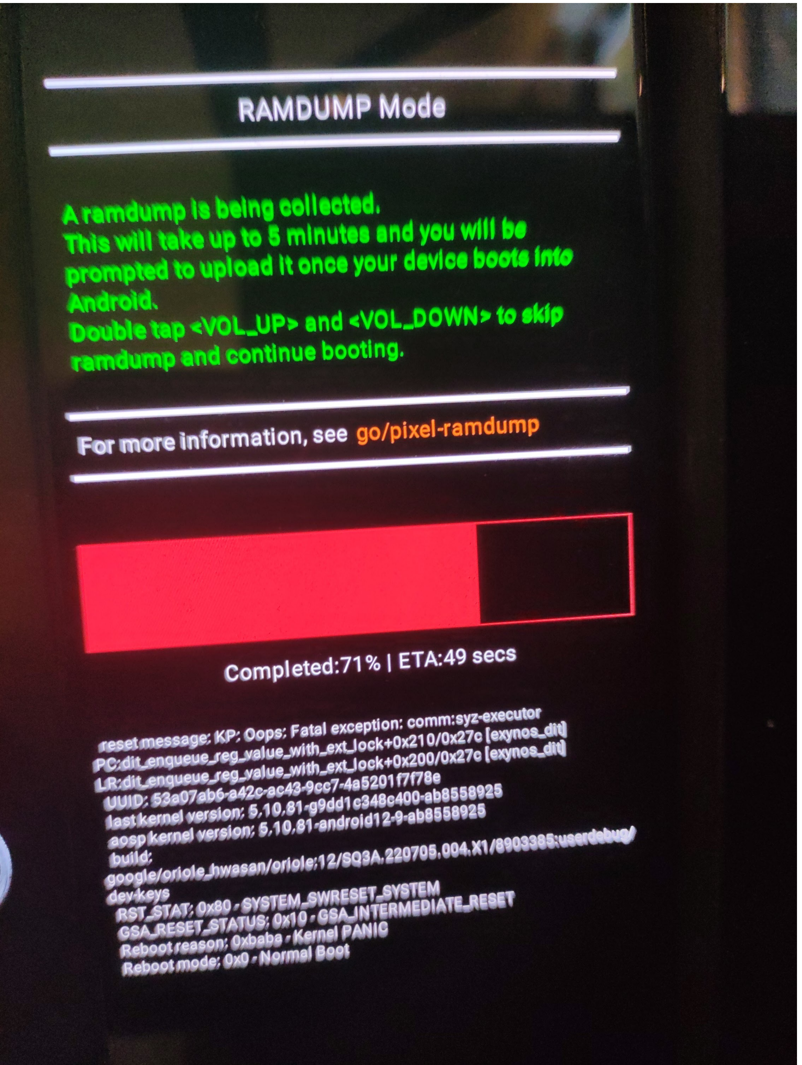
Fuzzing Android Kernel of Pixel 6

- SyzDescribe recovers 154 syscall handlers corresponding to 139 kernel drivers.
- Find 18 crashes



Kernel PANIC: KP: Asynchronous SError Interrupt
Kernel PANIC: KP: Oops: Fatal exception: __skb_ext_put
Kernel PANIC: KP: Oops: Fatal exception: dit_enqueue_reg_value_with_ext_lock
Kernel PANIC: KP: BRK handler: Fatal exception: dit_hal_ioctl
Kernel PANIC: KP: BRK handler: Fatal exception: dit_hal_get_netdev
Kernel PANIC: KP: BRK handler: Fatal exception in interrupt: comm:init, swapper/3-7
APC Watchdog: itom triggering err_fatal from HSIO USB31DRD_LINK to Refe
PMUCAL Watchdog: pmucal_local_disable: error on handling disable sequence. (pd: blkpwr_bo)
WARNING in drm_atomic_helper_commit_modeset_disables

WARNING in lwis_ioctl_handler
WARNING in gvotable_cast_vote
WARNING in irq_set_irq_wake
WARNING in kbase_mem_pool_grow
WARNING in drm_mode_object_add
WARNING in gpio_to_desc
WARNING in corrupted
Emergency Restart
INFO: corrupted



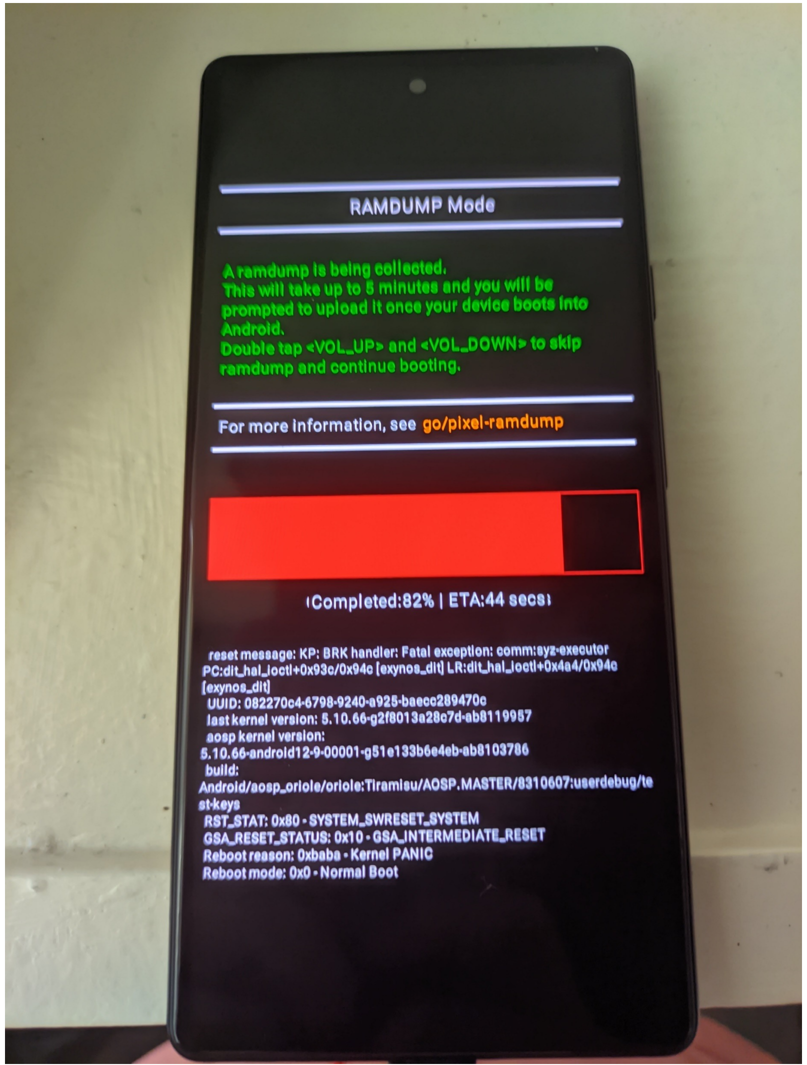
```
RAMDUMP Mode

A ramdump is being collected.
This will take up to 5 minutes and you will be
prompted to upload it once your device boots into
Android.
Double tap <VOL_UP> and <VOL_DOWN> to skip
ramdump and continue booting.

For more information, see go/pixel-ramdump

Completed:71% | ETA:49 secs

 reset message: KP: Oops: Fatal exception: comm:syz-executor
PC:dit_enqueue_reg_value_with_ext_lock+0x210/0x27c [exynos_dit]
LR:dit_enqueue_reg_value_with_ext_lock+0x200/0x27c [exynos_dit]
 UUID: 53a07ab6-a42c-ac43-9cc7-4a5201f7f78e
 last kernel version: 5.10.81-g9dd1c348c400-ab8558925
 aosp kernel version: 5.10.81-android12-9-ab8558925
 build:
google/oriole_hwasan/oriole:12/SQ3A.220705.004.X1/8903385:userdebug/
dev-keys
 RST_STAT: 0x80 - SYSTEM_SWRESET_SYSTEM
 GSA_RESET_STATUS: 0x10 - GSA_INTERMEDIATE_RESET
 Reboot reason: 0xbaba - Kernel PANIC
 Reboot mode: 0x0 - Normal Boot
```



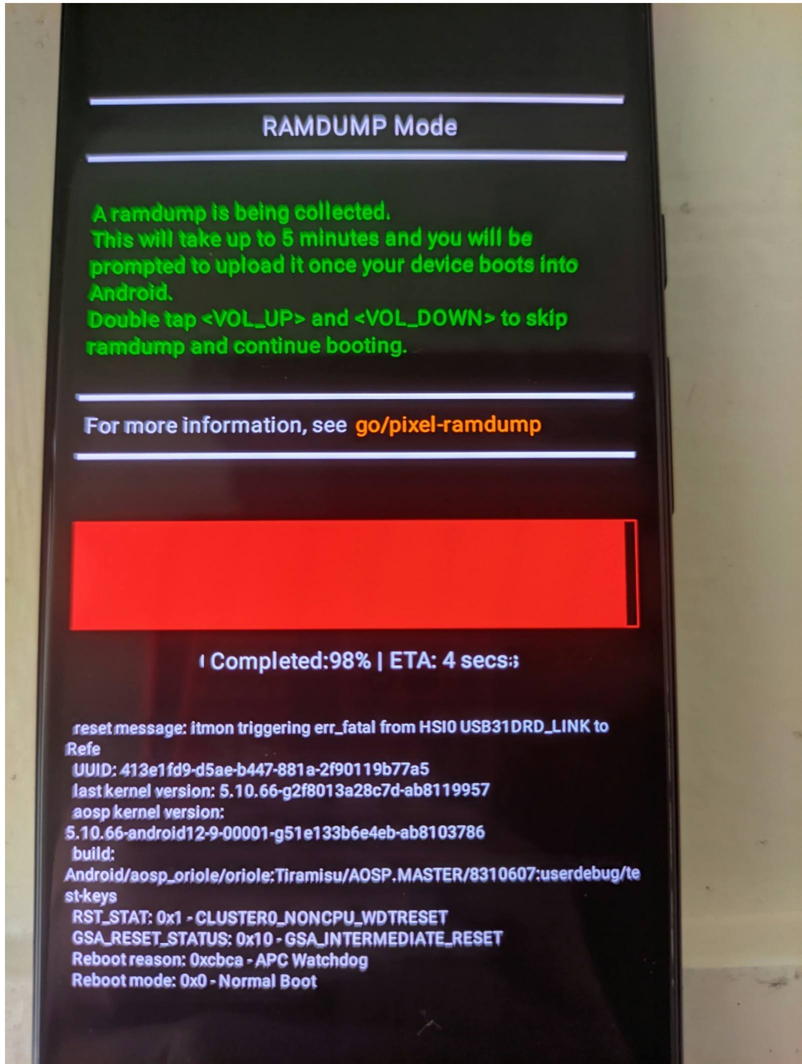
```
RAMDUMP Mode

A ramdump is being collected.
This will take up to 5 minutes and you will be
prompted to upload it once your device boots into
Android.
Double tap <VOL_UP> and <VOL_DOWN> to skip
ramdump and continue booting.

For more information, see go/pixel-ramdump

Completed:82% | ETA:44 secs

 reset message: KP: BRK handler: Fatal exception: comm:syz-executor
PC:dit_hal_ioctl+0x93c/0x94c [exynos_dit] LR:dit_hal_ioctl+0x4a4/0x94c
[exynos_dit]
 UUID: 082270c4-6798-9240-a925-baecc289470c
 last kernel version: 5.10.66-g2f8013a28c7d-ab8119957
 aosp kernel version:
5.10.66-android12-9-00001-g51e133b6e4eb-ab8103786
 build:
Android/aosp_oriole/oriole:Tiramisu/AOSP.MASTER/8310607:userdebug/te
st-keys
 RST_STAT: 0x80 - SYSTEM_SWRESET_SYSTEM
 GSA_RESET_STATUS: 0x10 - GSA_INTERMEDIATE_RESET
 Reboot reason: 0xbaba - Kernel PANIC
 Reboot mode: 0x0 - Normal Boot
```



```
RAMDUMP Mode

A ramdump is being collected.
This will take up to 5 minutes and you will be
prompted to upload it once your device boots into
Android.
Double tap <VOL_UP> and <VOL_DOWN> to skip
ramdump and continue booting.

For more information, see go/pixel-ramdump

Completed:98% | ETA: 4 secs

 reset message: itmon triggering err_fatal from HSI0 USB31DRD_LINK to
Refe
 UUID: 413e1fd9-d5ae-b447-881a-2f90119b77a5
 last kernel version: 5.10.66-g2f8013a28c7d-ab8119957
 aosp kernel version:
5.10.66-android12-9-00001-g51e133b6e4eb-ab8103786
 build:
Android/aosp_oriole/oriole:Tiramisu/AOSP.MASTER/8310607:userdebug/te
st-keys
 RST_STAT: 0x1 - CLUSTER0_NONCPU_WDTRESET
 GSA_RESET_STATUS: 0x10 - GSA_INTERMEDIATE_RESET
 Reboot reason: 0xcbca - APC Watchdog
 Reboot mode: 0x0 - Normal Boot
```

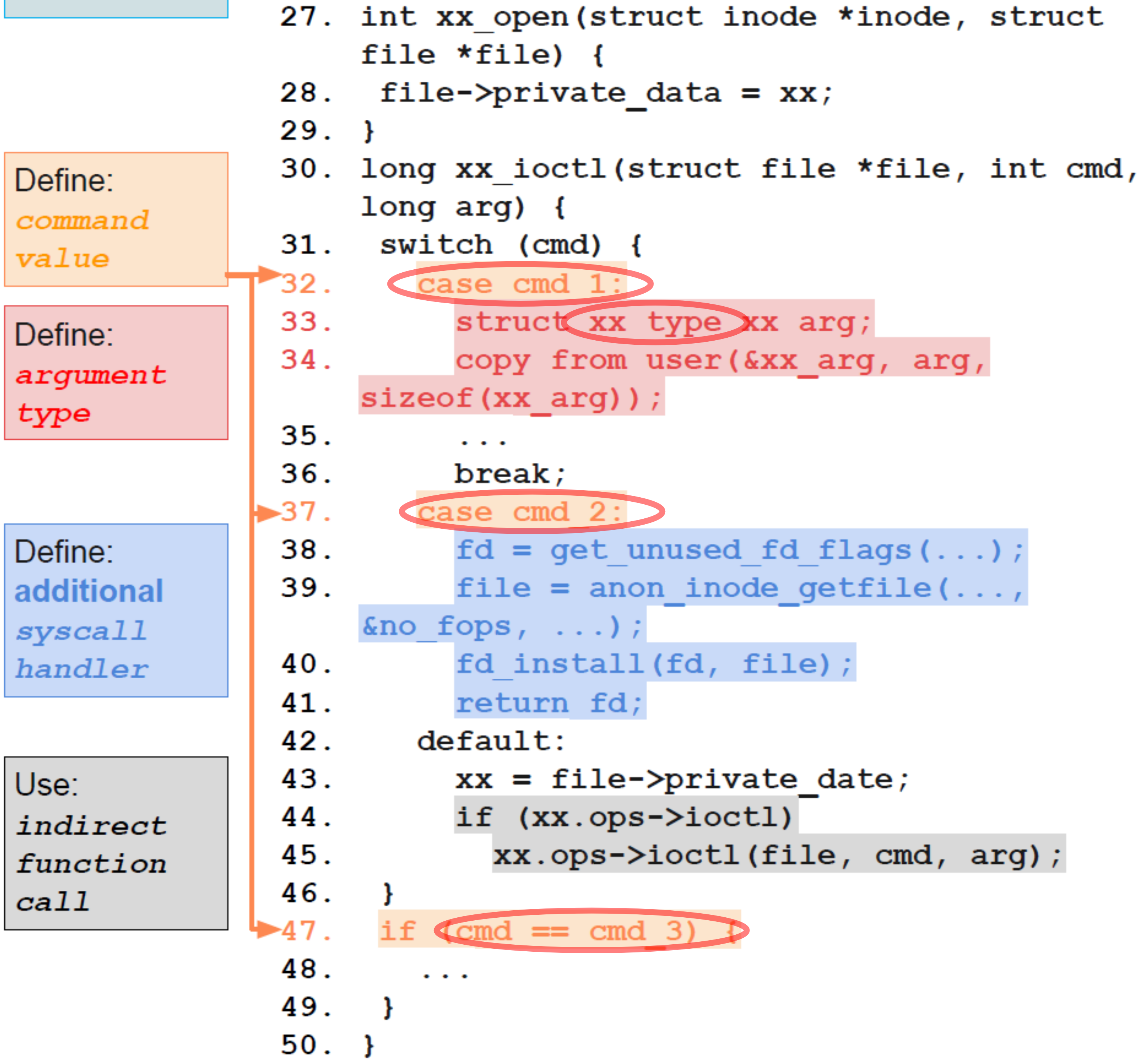
Q&A

- Our open-source repo:
- https://github.com/seclab-ucr/SyzDescribe

- Email: yhao016@ucr.edu
- Twitter: @yuhao2222
- About Yu Hao:
  - A final year Ph.D. Candidate at UC Riverside under the supervision of Professor Zhiyun Qian.
  - Focus on Linux kernel security, kernel fuzzing, symbolic execution and static analysis.
  - Published papers in top academic conferences (S&P, CCS, NDSS, ICSE and FSE).
  - In job market

Syscall Handler Analysis

1. **Command Value Recovery**
2. **Argument Type Recovery**
3. Additional Syscall Handler Recovery



Define: *command value*

Define: *argument type*

Define: **additional** *syscall handler*

Use: *indirect function call*

```
int xx_open(struct inode *inode, struct
file *file) {
 file->private_data = xx;
}
long xx_ioctl(struct file *file, int cmd,
long arg) {
 switch (cmd) {
  case cmd 1:
    struct xx type xx arg;
    copy from user(&xx_arg, arg,
sizeof(xx_arg));
    ...
    break;
  case cmd 2:
    fd = get_unused_fd_flags(...);
    file = anon_inode_getfile(...,
&no_fops, ...);
    fd_install(fd, file);
    return fd;
  default:
    xx = file->private_date;
    if (xx.ops->ioctl)
      xx.ops->ioctl(file, cmd, arg);
 }
 if (cmd == cmd_3) {
   ...
 }
}
```