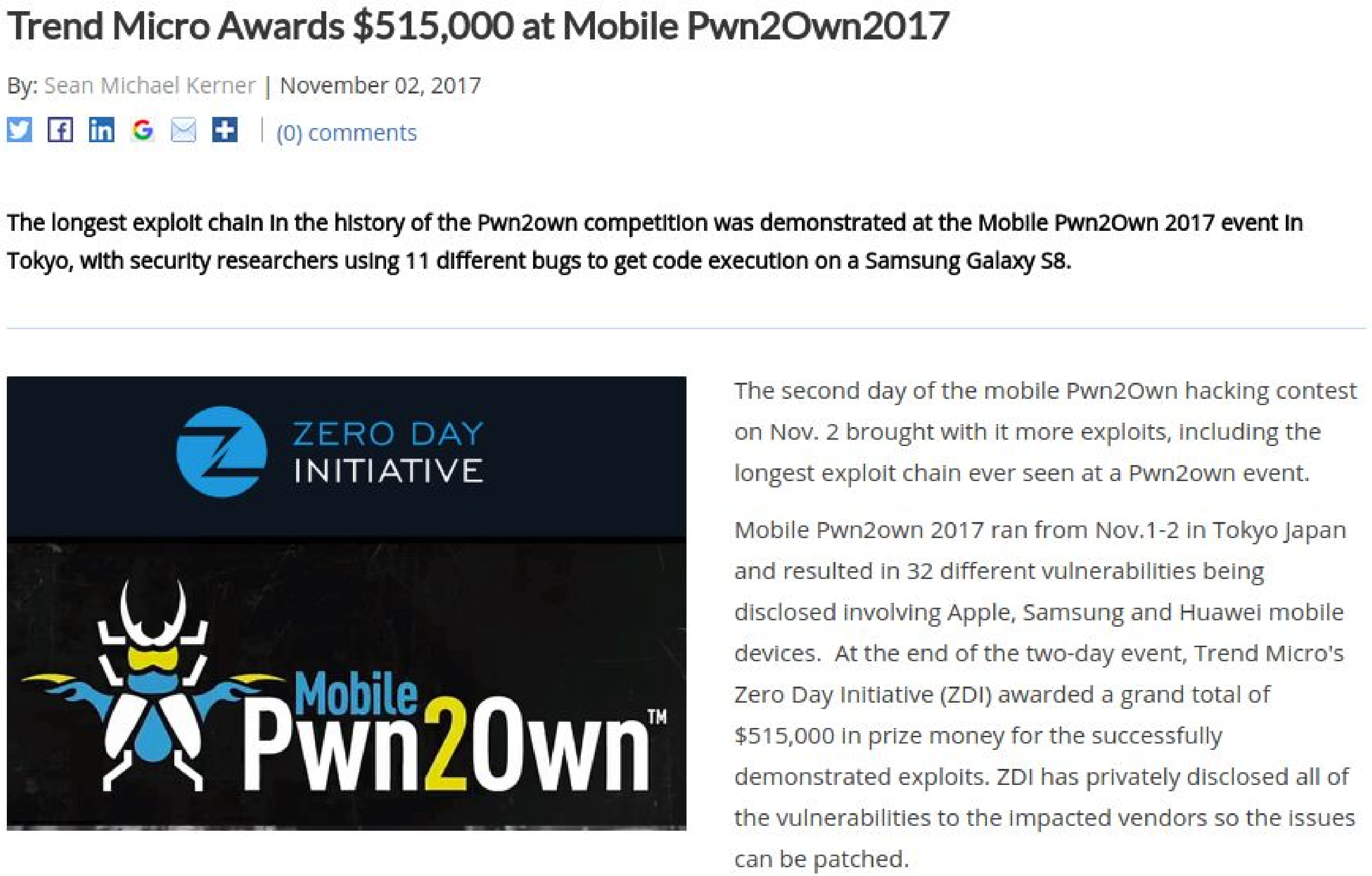
DIFUZE: Interface Aware Fuzzing for Kernel Drivers

Jake Corina, **Aravind Machiry**, Chris Salls, Yan Shoshitaishvili, Shuang Hao, Christopher Kruegel, and Giovanni Vigna

THE COMPUTER SECURITY GROUP AT UC SANTA BARBARA

Yesterday: 32-Zero days in Smartphones



Trend Micro Awards $515,000 at Mobile Pwn2Own2017

By: Sean Michael Kerner | November 02, 2017

(0) comments

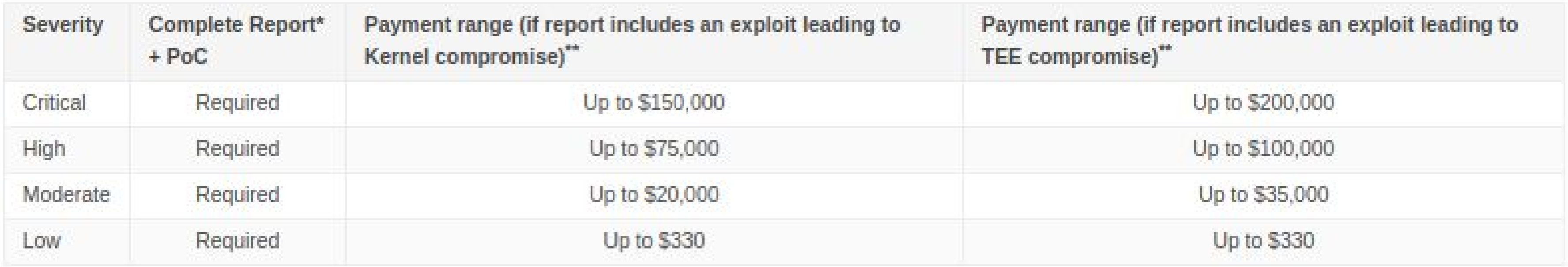
The longest exploit chain in the history of the Pwn2own competition was demonstrated at the Mobile Pwn2Own 2017 event in Tokyo, with security researchers using 11 different bugs to get code execution on a Samsung Galaxy S8.

The second day of the mobile Pwn2Own hacking contest on Nov. 2 brought with it more exploits, including the longest exploit chain ever seen at a Pwn2own event.

Mobile Pwn2own 2017 ran from Nov.1-2 in Tokyo Japan and resulted in 32 different vulnerabilities being disclosed involving Apple, Samsung and Huawei mobile devices. At the end of the two-day event, Trend Micro's Zero Day Initiative (ZDI) awarded a grand total of $515,000 in prize money for the successfully demonstrated exploits. ZDI has privately disclosed all of the vulnerabilities to the impacted vendors so the issues can be patched.

Bad day for Smartphones!!!

- We found **36-bugs in 7 different devices**.



Severity	Complete Report* + PoC	Payment range (if report includes an exploit leading to Kernel compromise)**	Payment range (if report includes an exploit leading to TEE compromise)**
Critical	Required	Up to $150,000	Up to $200,000
High	Required	Up to $75,000	Up to $100,000
Moderate	Required	Up to $20,000	Up to $35,000
Low	Required	Up to $330	Up to $330

Kernel Drivers

- Provide interface to interact with hardware.
- Written by hardware vendors.
- Run in kernel space.

Linux Kernel Drivers

- A file in the file system.
    - /dev/qseecom

- Read, Write, **ioctl**

- Run in kernel space.

Linux Kernel Drivers

- Tightly coupled with the device.
  - Written by vendors.
- Self-contained module.
- Allows for easy customization:
  - Android devices, Amazon Echo. IoT.

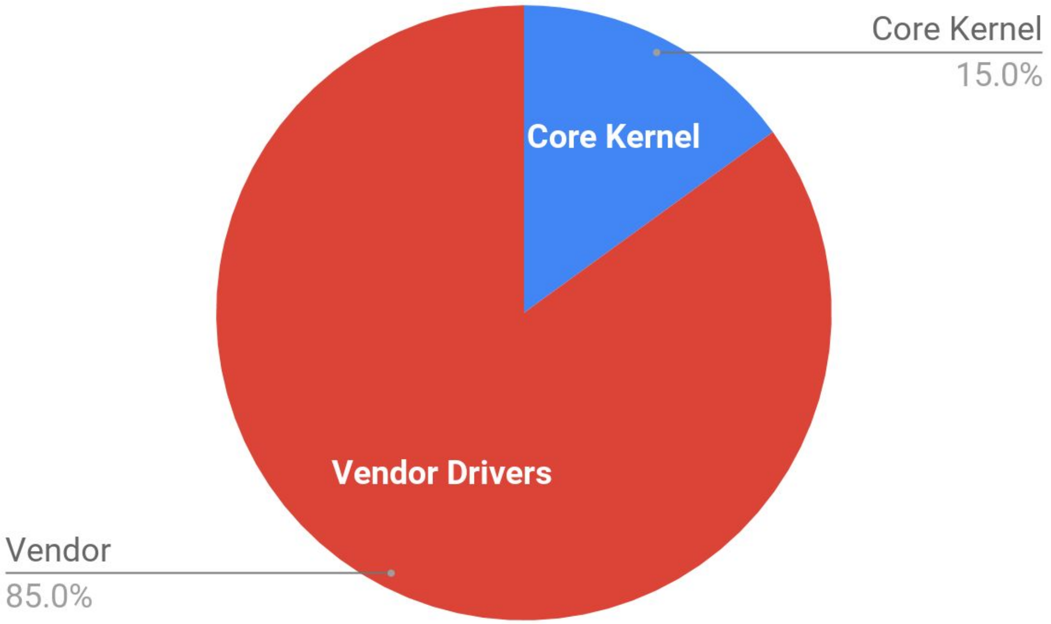
Android Kernel

- Based on Linux.

- Dominates the smartphone OS market.
  - 86.8% of the market in 2016 Q3 (Source: IDC, Nov 2016).

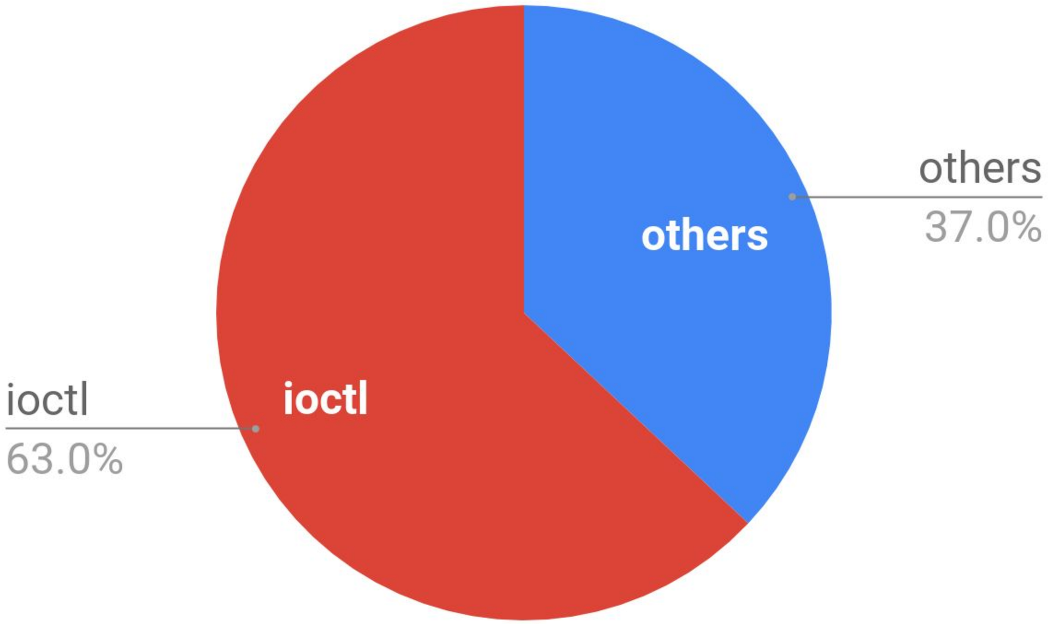
- Lots of Vendors → Lots of Hardware → Lots of drivers.

Where are the Android kernel bugs?



android: Protecting the kernel, Jeff Vander Stoep, Linux Foundation 2016

How are these bugs reached from user space?



android: Protecting the kernel, Jeff Vander Stoep, Linux Foundation 2016

Ioctl

- **I**nput **O**utput **C**on**t**ro**l**.

- System call to allow device operations that can’t be well modeled as a “normal” syscall.

- Bound to a file, requires a valid file descriptor.

Ioctl

```
ioctl(
    int fd,
    unsigned long command,
    unsigned long param
   );
```

Ioctl

```
ioctl(
    int fd,
    unsigned long command,
    unsigned long param
);
```

← Valid file descriptor.

Ioctl

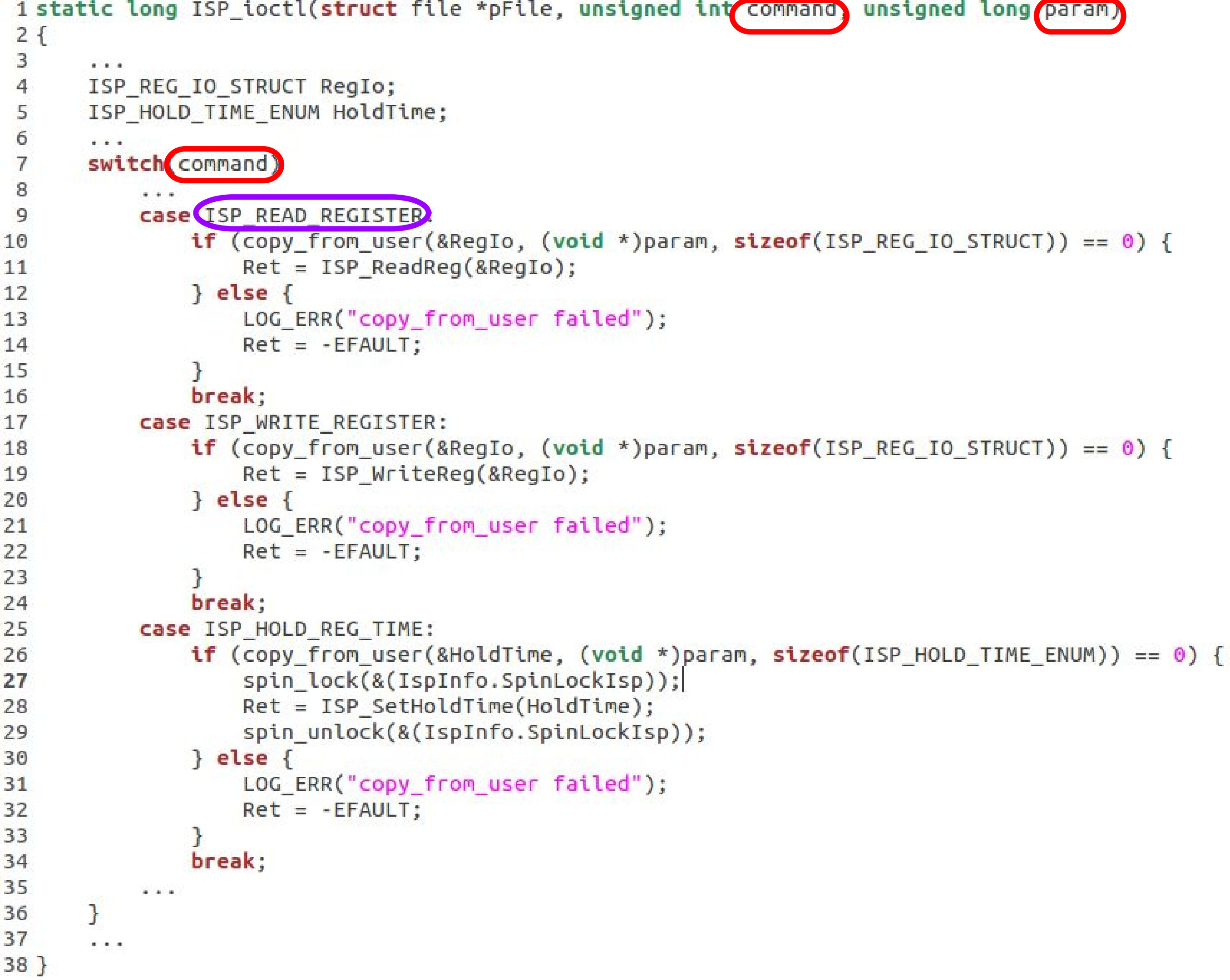
```
ioctl(
    int fd,                          Valid file descriptor.
    unsigned long command,
    unsigned long param
);
```

Unverified user data.

```c
static long ISP_ioctl(struct file *pFile, unsigned int command, unsigned long param)
{
    ...
    ISP_REG_IO_STRUCT RegIo;
    ISP_HOLD_TIME_ENUM HoldTime;
    ...
    switch(command)
        ...
        case ISP_READ_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_ReadReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_WRITE_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_WriteReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_HOLD_REG_TIME:
            if (copy_from_user(&HoldTime, (void *)param, sizeof(ISP_HOLD_TIME_ENUM)) == 0) {
                spin_lock(&(IspInfo.SpinLockIsp));
                Ret = ISP_SetHoldTime(HoldTime);
                spin_unlock(&(IspInfo.SpinLockIsp));
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        ...
    }
    ...
}
```

```c
static long ISP_ioctl(struct file *pFile, unsigned int command, unsigned long param)
{
    ...
    ISP_REG_IO_STRUCT RegIo;
    ISP_HOLD_TIME_ENUM HoldTime;
    ...
    switch(command)
        ...
        case ISP_READ_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_ReadReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_WRITE_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_WriteReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_HOLD_REG_TIME:
            if (copy_from_user(&HoldTime, (void *)param, sizeof(ISP_HOLD_TIME_ENUM)) == 0) {
                spin_lock(&(IspInfo.SpinLockIsp));
                Ret = ISP_SetHoldTime(HoldTime);
                spin_unlock(&(IspInfo.SpinLockIsp));
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        ...
    }
    ...
}
```

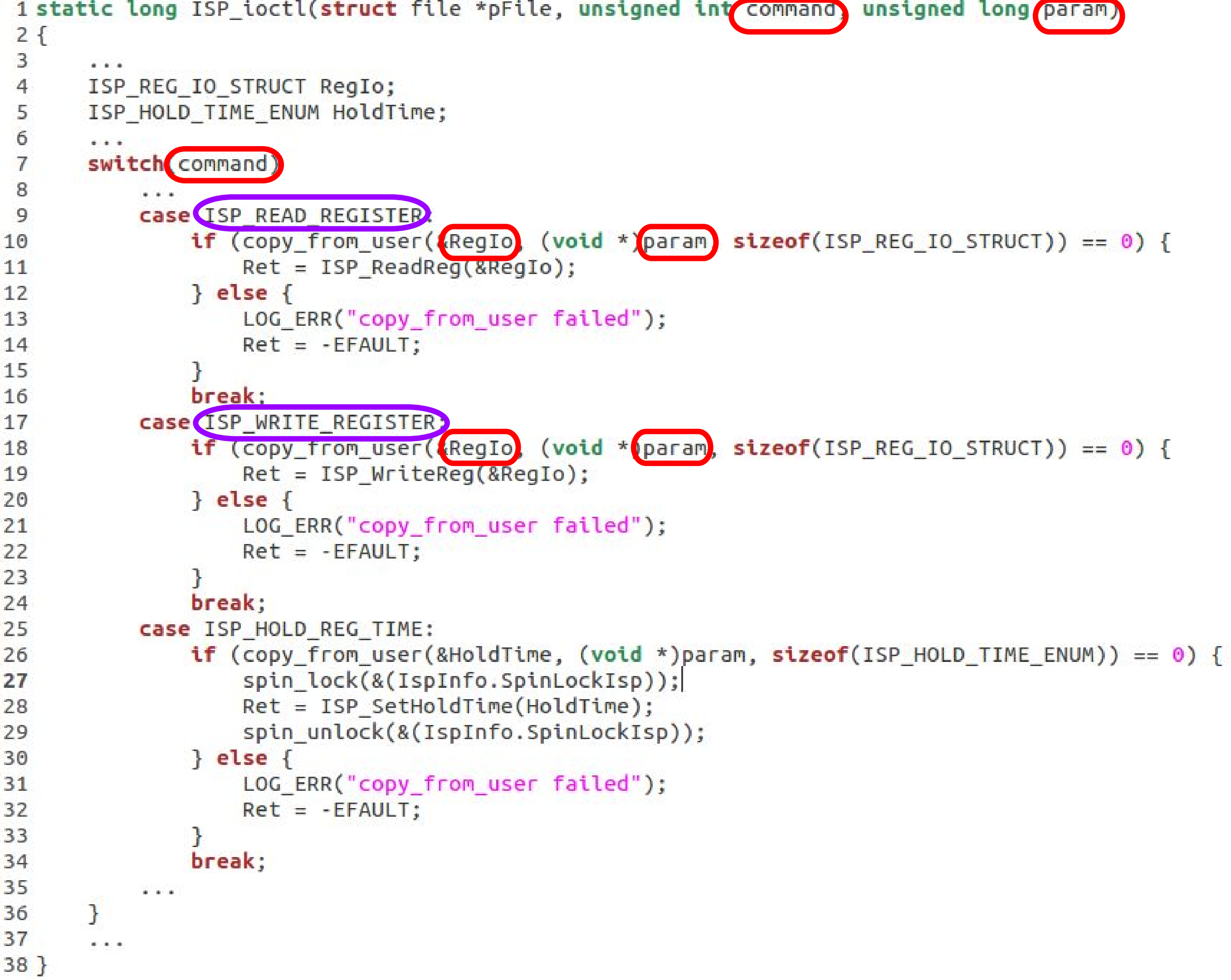
```c
static long ISP_ioctl(struct file *pFile, unsigned int command, unsigned long param)
{
    ...
    ISP_REG_IO_STRUCT RegIo;
    ISP_HOLD_TIME_ENUM HoldTime;
    ...
    switch(command)
        ...
        case ISP_READ_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_ReadReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_WRITE_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_WriteReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_HOLD_REG_TIME:
            if (copy_from_user(&HoldTime, (void *)param, sizeof(ISP_HOLD_TIME_ENUM)) == 0) {
                spin_lock(&(IspInfo.SpinLockIsp));
                Ret = ISP_SetHoldTime(HoldTime);
                spin_unlock(&(IspInfo.SpinLockIsp));
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        ...
    }
    ...
}
```



```c
static long ISP_ioctl(struct file *pFile, unsigned int command, unsigned long param)
{
    ...
    ISP_REG_IO_STRUCT RegIo;
    ISP_HOLD_TIME_ENUM HoldTime;
    ...
    switch (command)
        ...
        case ISP_READ_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_ReadReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_WRITE_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_WriteReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_HOLD_REG_TIME:
            if (copy_from_user(&HoldTime, (void *)param, sizeof(ISP_HOLD_TIME_ENUM)) == 0) {
                spin_lock(&(IspInfo.SpinLockIsp));
                Ret = ISP_SetHoldTime(HoldTime);
                spin_unlock(&(IspInfo.SpinLockIsp));
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        ...
    }
    ...
}
```

```
static long ISP_ioctl(struct file *pFile, unsigned int command, unsigned long param)
{
    ...
    ISP_REG_IO_STRUCT RegIo;
    ISP_HOLD_TIME_ENUM HoldTime;
    ...
    switch(command)
        ...
        case ISP_READ_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_ReadReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_WRITE_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_WriteReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_HOLD_REG_TIME:
            if (copy_from_user(&HoldTime, (void *)param, sizeof(ISP_HOLD_TIME_ENUM)) == 0) {
                spin_lock(&(IspInfo.SpinLockIsp));
                Ret = ISP_SetHoldTime(HoldTime);
                spin_unlock(&(IspInfo.SpinLockIsp));
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        ...
    }
    ...
}
```

```c
static long ISP_ioctl(struct file *pFile, unsigned int command, unsigned long param)
{
    ...
    ISP_REG_IO_STRUCT RegIo;
    ISP_HOLD_TIME_ENUM HoldTime;
    ...
    switch(command)
        ...
        case ISP_READ_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_ReadReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_WRITE_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_WriteReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_HOLD_REG_TIME:
            if (copy_from_user(&HoldTime, (void *)param, sizeof(ISP_HOLD_TIME_ENUM)) == 0) {
                spin_lock(&(IspInfo.SpinLockIsp));
                Ret = ISP_SetHoldTime(HoldTime);
                spin_unlock(&(IspInfo.SpinLockIsp));
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        ...
    }
    ...
}
```



```c
static long ISP_ioctl(struct file *pFile, unsigned int command, unsigned long param)
{
    ...
    ISP_REG_IO_STRUCT RegIo;
    ISP_HOLD_TIME_ENUM HoldTime;
    ...
    switch(command)
        ...
        case ISP_READ_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_ReadReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_WRITE_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_WriteReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_HOLD_REG_TIME:
            if (copy_from_user(&HoldTime, (void *)param, sizeof(ISP_HOLD_TIME_ENUM)) == 0) {
                spin_lock(&(IspInfo.SpinLockIsp));
                Ret = ISP_SetHoldTime(HoldTime);
                spin_unlock(&(IspInfo.SpinLockIsp));
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        ...
    }
    ...
}
```

```
static long ISP_ioctl(struct file *pFile, unsigned int command, unsigned long param)
{
    ...
    ISP_REG_IO_STRUCT RegIo;
    ISP_HOLD_TIME_ENUM HoldTime;
    ...
    switch(command)
        ...
        case ISP_READ_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_ReadReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_WRITE_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_WriteReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_HOLD_REG_TIME:
            if (copy_from_user(&HoldTime, (void *)param, sizeof(ISP_HOLD_TIME_ENUM)) == 0) {
                spin_lock(&(IspInfo.SpinLockIsp));
                Ret = ISP_SetHoldTime(HoldTime);
                spin_unlock(&(IspInfo.SpinLockIsp));
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        ...
    }
    ...
}
```

How can we find vulnerabilities in ioctls?

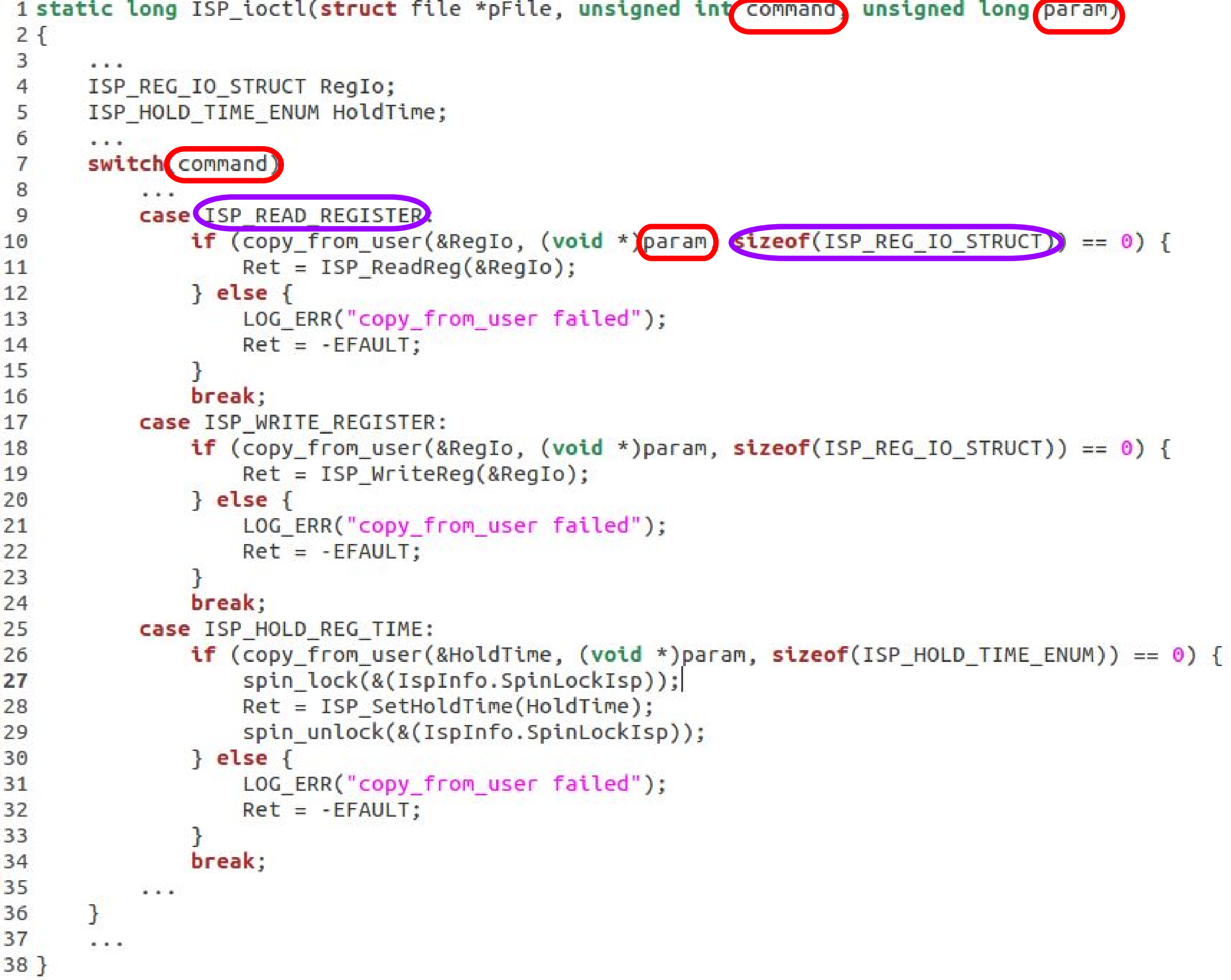
- Static Analysis:
  - False positives.
  - Hard to find common bug classes like: use-after-free.

- Dynamic Analysis:
  - Fuzzing
  - Easy to triage.
  - Can find complex bugs.

Let's Fuzz

- Generate random input and hope the kernel panics.

- Ioctls have highly structured input.



```c
static long ISP_ioctl(struct file *pFile, unsigned int command, unsigned long param)
{
    ...
    ISP_REG_IO_STRUCT RegIo;
    ISP_HOLD_TIME_ENUM HoldTime;
    ...
    switch(command)
        ...
        case ISP_READ_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_ReadReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_WRITE_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_WriteReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_HOLD_REG_TIME:
            if (copy_from_user(&HoldTime, (void *)param, sizeof(ISP_HOLD_TIME_ENUM)) == 0) {
                spin_lock(&(IspInfo.SpinLockIsp));
                Ret = ISP_SetHoldTime(HoldTime);
                spin_unlock(&(IspInfo.SpinLockIsp));
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        ...
    }
    ...
}
```

Highly Constrained Input

- If command == ISP_READ_REGISTER then param should be a valid user pointer to ISP_REG_IO_STRUCT.

Highly Constrained Input

- If command == ISP_READ_REGISTER then param should be a valid user pointer to ISP_REG_IO_STRUCT.

- Idea!! Always make param a valid user pointer.

Highly Constrained Input

- If command == ISP_READ_REGISTER then param should be a valid user pointer to ISP_REG_IO_STRUCT.

- Idea!! Always make param a valid user pointer.

What if there are pointers inside the expected struct?

```
typedef struct {
    ISP_REG_STRUCT *pData;
    unsigned int Count;
} ISP_REG_IO_STRUCT;
```

```c
static long ISP_ioctl(struct file *pFile, unsigned int command, unsigned long param)
{
    ...
    ISP_REG_IO_STRUCT RegIo;
    ISP_HOLD_TIME_ENUM HoldTime;
    ...
    switch(command)
        ...
        case ISP_READ_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_ReadReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_WRITE_REGISTER:
            if (copy_from_user(&RegIo, (void *)param, sizeof(ISP_REG_IO_STRUCT)) == 0) {
                Ret = ISP_WriteReg(&RegIo);
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        case ISP_HOLD_REG_TIME:
            if (copy_from_user(&HoldTime, (void *)param, sizeof(ISP_HOLD_TIME_ENUM)) == 0) {
                spin_lock(&(IspInfo.SpinLockIsp));
                Ret = ISP_SetHoldTime(HoldTime);
                spin_unlock(&(IspInfo.SpinLockIsp));
            } else {
                LOG_ERR("copy_from_user failed");
                Ret = -EFAULT;
            }
            break;
        ...
    }
    ...
}
```

Highly Constrained Input

- If command == ISP_READ_REGISTER then param should be a valid user pointer to ISP_REG_IO_STRUCT.

Highly Constrained Input

- If command == ISP_READ_REGISTER then param should be a valid user pointer to ISP_REG_IO_STRUCT.

- If command == ISP_WRITE_REGISTER then param should be a valid user pointer to ISP_REG_IO_STRUCT.

Highly Constrained Input

- If command == ISP_READ_REGISTER then param should be a valid user pointer to ISP_REG_IO_STRUCT.

- If command == ISP_WRITE_REGISTER then param should be a valid user pointer to ISP_REG_IO_STRUCT.

- If command == ISP_HOLD_REG_TIME then param should be a valid user pointer to ISP_HOLD_TIME_ENUM.

```c
int gTable[128];

ioctl_handler(struct file *pFile, unsigned int cmd, unsigned long param) {
    int idx;
    foo_t foo;
    switch(cmd) {
        case 1337:
            if (copy_from_user(&foo, (void *)param, sizeof(foo_t)) != 0)
                return -1;

            /* WRITE */
            if (foo.type == 77)
                gTable[foo.idx] = foo.val;

            /* CLEAR */
            else if (foo.type == 78)
                kmemset(gTable, 0, sizeof(gTable));

            else
                return -1;

            break;

        default:
            return -1;
    }
}
```

```c
int gTable[128];

ioctl_handler(struct file *pFile, unsigned int cmd, unsigned long param) {
    int idx;
    foo_t foo;
    switch(cmd) {
        case 1337:
            if (copy_from_user(&foo, (void *)param, sizeof(foo_t)) != 0)
                return -1;

            /* WRITE */
            if (foo.type == 77)
                gTable[foo.idx] = foo.val;

            /* CLEAR */
            else if (foo.type == 78)
                kmemset(gTable, 0, sizeof(gTable));

            else
                return -1;

            break;

        default:
            return -1;
    }
}
```

Arbitrary kernel heap write.

```
int gTable[128];

ioctl_handler(struct file *pFile, unsigned int cmd, unsigned long param){
    int idx;
    foo_t foo;
    switch(cmd) {
        case 1337:
            if (copy_from_user(&foo, (void *)param, sizeof(foo_t)) != 0)
                return -1;

            /* WRITE */
            if (foo.type == 77)
                gTable[foo.idx] = foo.val;

            /* CLEAR */
            else if (foo.type == 78)
                kmemset(gTable, 0, sizeof(gTable));

            else
                return -1;

            break;

        default:
            return -1;
    }
}
```

Arbitrary kernel heap write.

Highly Constrained Input

- You can trigger the bug, only if command **==** 1337 and param is **a valid pointer** to the structure:

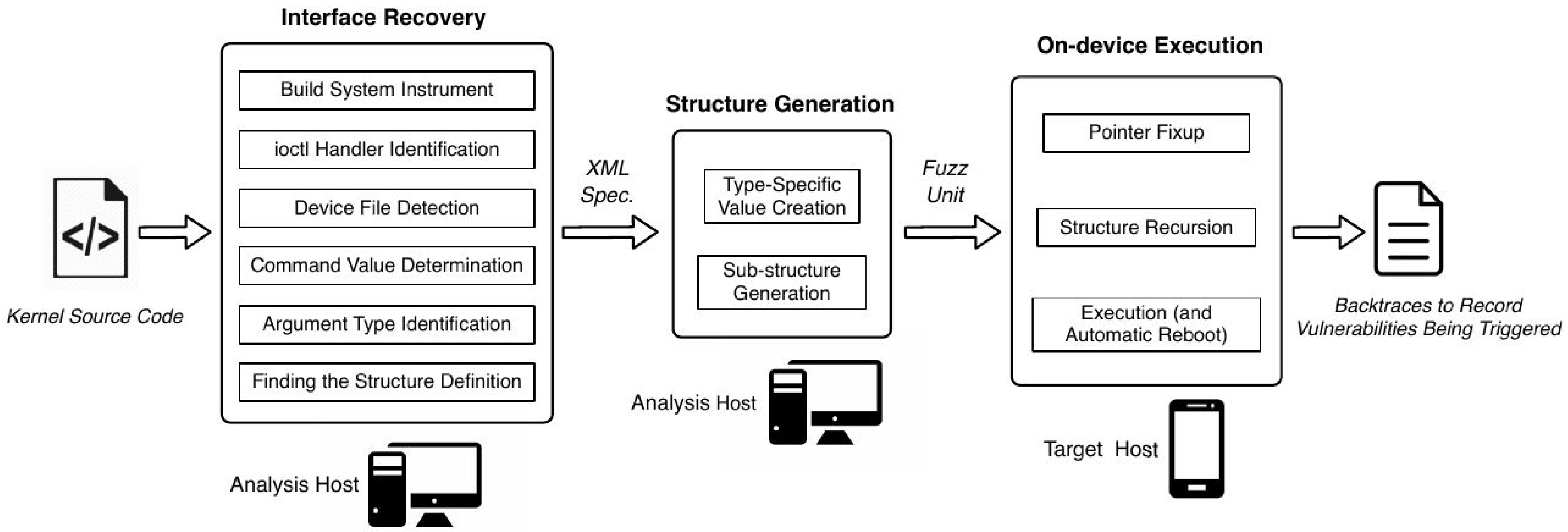
```
typedef struct {
    type_enum type; ( == 77)
    int idx; ( >= 128)
    int val;
} foo_t
```

DIFUZE: Interface Aware Fuzzing

- Recover all the command values, corresponding param types automatically.

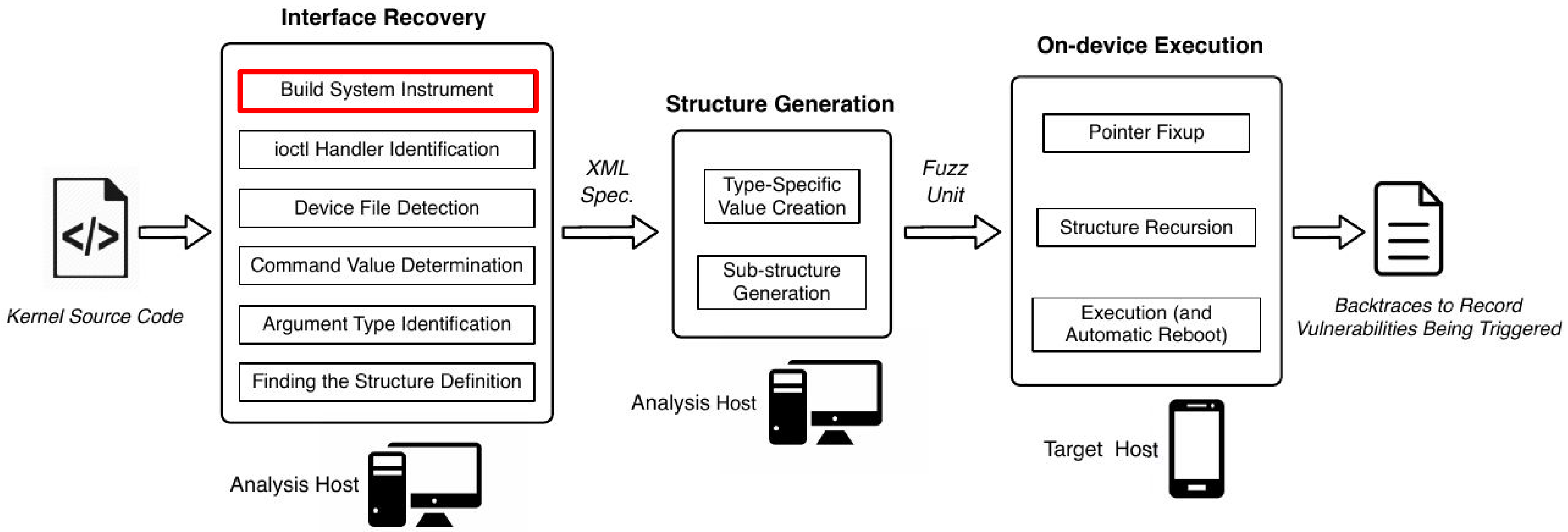
- This will reduce the state space and help in effective fuzzing.

DIFUZE



Kernel Source Code
Interface Recovery
Build System Instrument
ioctl Handler Identification
Device File Detection
Command Value Determination
Argument Type Identification
Finding the Structure Definition
Analysis Host
XML Spec.
Structure Generation
Type-Specific Value Creation
Sub-structure Generation
Analysis Host
Fuzz Unit
On-device Execution
Pointer Fixup
Structure Recursion
Execution (and Automatic Reboot)
Target Host
Backtraces to Record Vulnerabilities Being Triggered

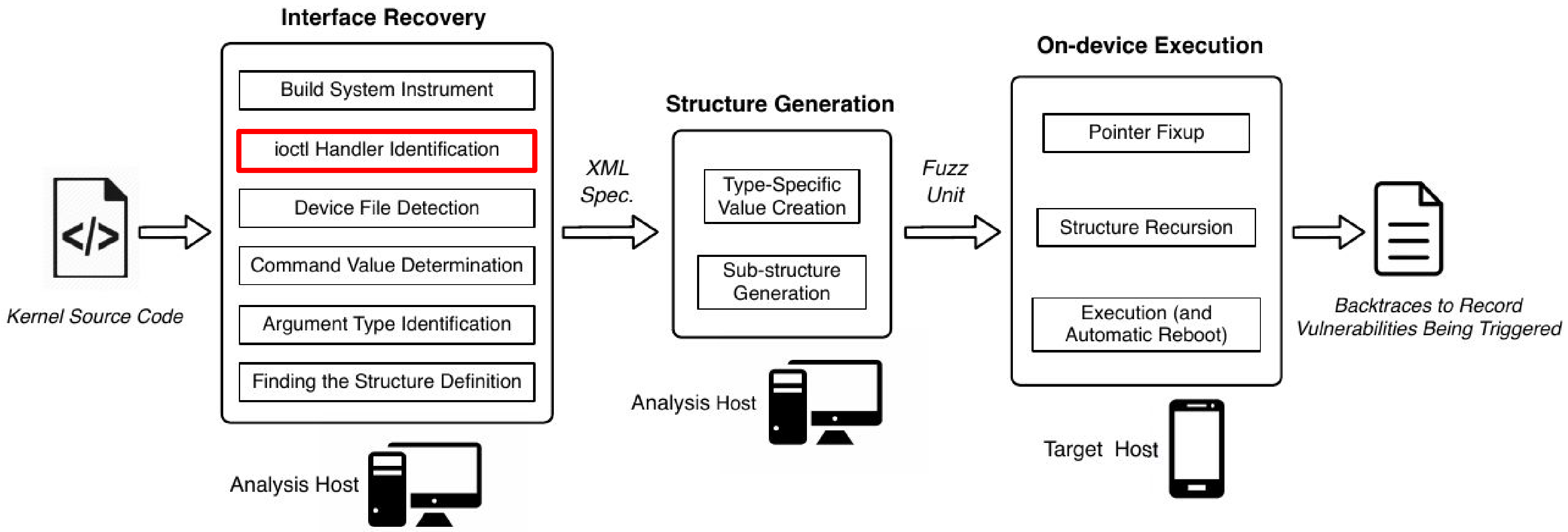
DIFUZE


Interface Recovery
Build System Instrument
ioctl Handler Identification
Device File Detection
Command Value Determination
Argument Type Identification
Finding the Structure Definition
Kernel Source Code
Analysis Host
XML Spec.
Structure Generation
Type-Specific Value Creation
Sub-structure Generation
Analysis Host
Fuzz Unit
On-device Execution
Pointer Fixup
Structure Recursion
Execution (and Automatic Reboot)
Target Host
Backtraces to Record Vulnerabilities Being Triggered

Build System Instrumentation

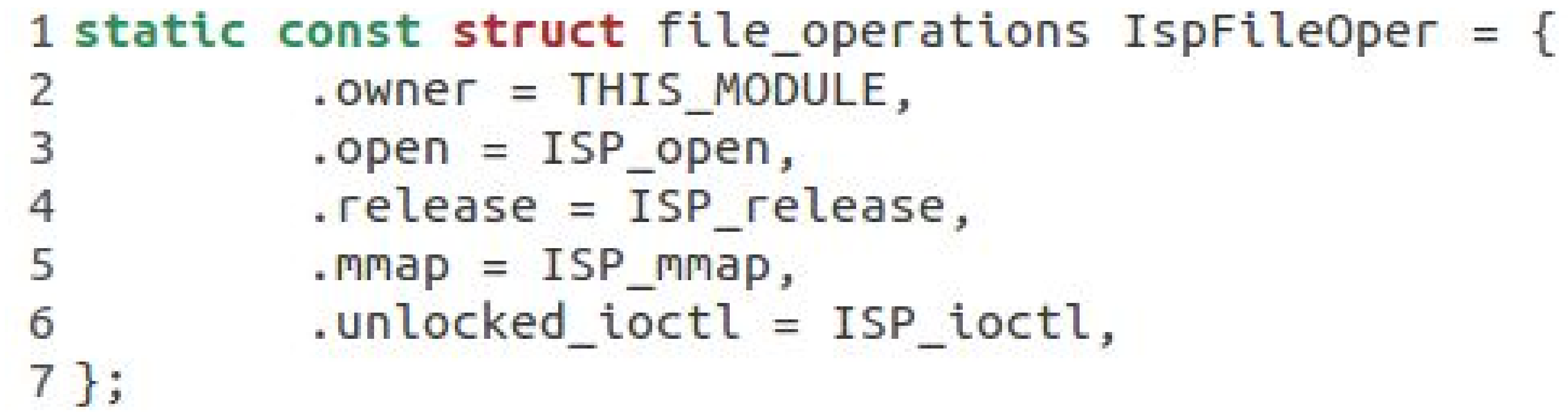
- Goal: LLVM Bitcode file for the entire driver.
  - Compiler kernel using GCC and capture build commands.
  - Transform GCC commands to Clang commands.
  - Link corresponding bitcode files.

DIFUZE



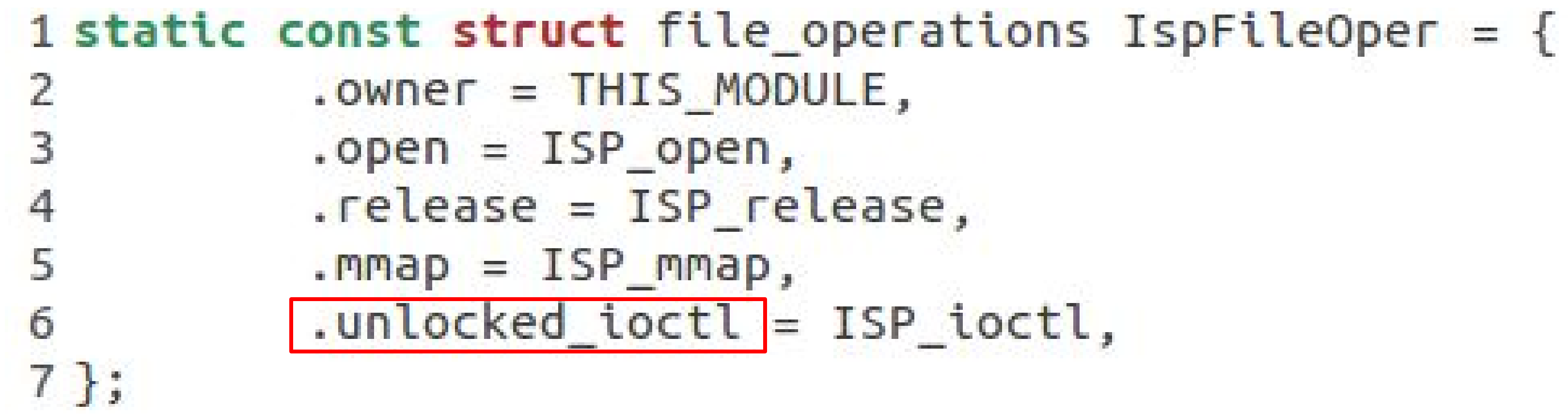
Kernel Source Code
Interface Recovery
Build System Instrument
ioctl Handler Identification
Device File Detection
Command Value Determination
Argument Type Identification
Finding the Structure Definition
Analysis Host
XML Spec.
Structure Generation
Type-Specific Value Creation
Sub-structure Generation
Analysis Host
Fuzz Unit
On-device Execution
Pointer Fixup
Structure Recursion
Execution (and Automatic Reboot)
Target Host
Backtraces to Record Vulnerabilities Being Triggered

Ioctl handler identification



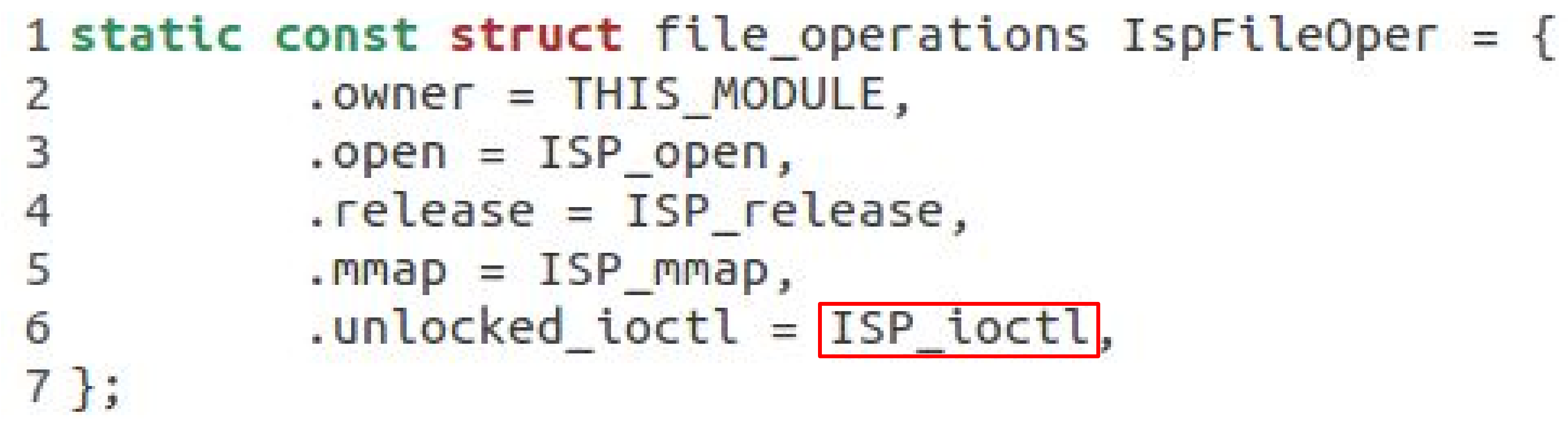
```c
static const struct file_operations IspFileOper = {
        .owner = THIS_MODULE,
        .open = ISP_open,
        .release = ISP_release,
        .mmap = ISP_mmap,
        .unlocked_ioctl = ISP_ioctl,
};
```

Ioctl handler identification



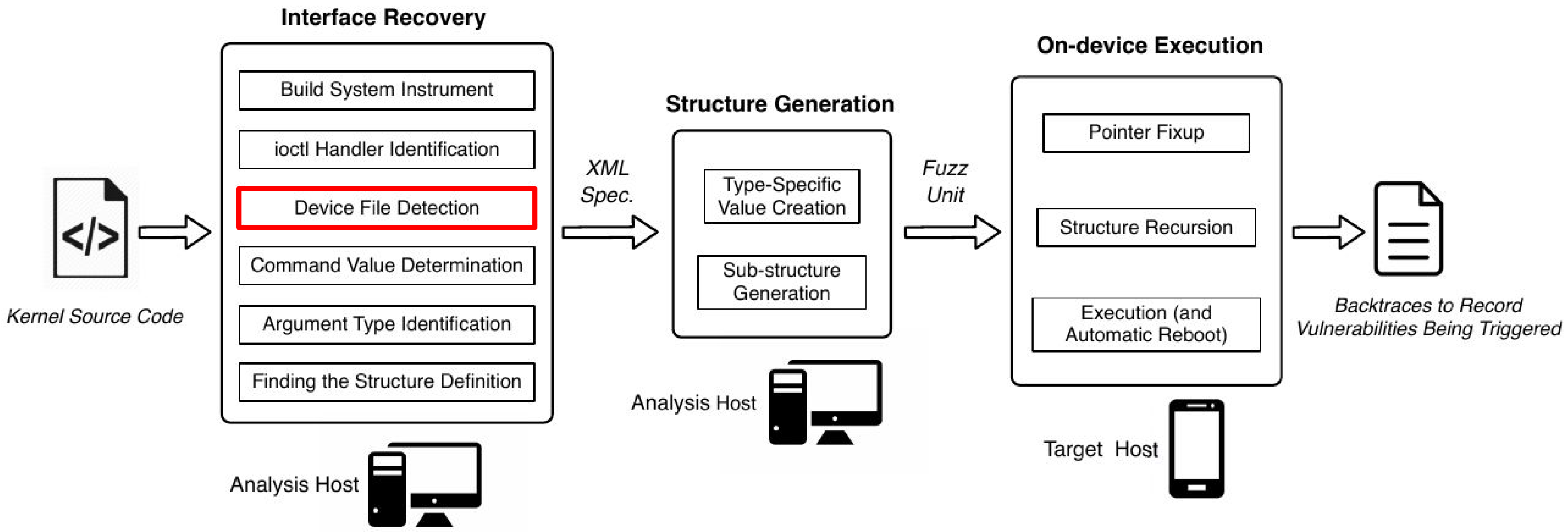
```
static const struct file_operations IspFileOper = {
            .owner = THIS_MODULE,
            .open = ISP_open,
            .release = ISP_release,
            .mmap = ISP_mmap,
            .unlocked_ioctl = ISP_ioctl,
};
```

Ioctl handler identification



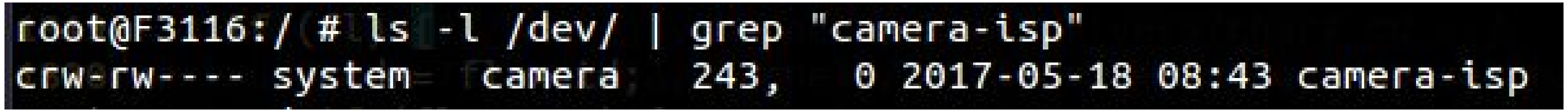
```
static const struct file_operations IspFileOper = {
            .owner = THIS_MODULE,
            .open = ISP_open,
            .release = ISP_release,
            .mmap = ISP_mmap,
            .unlocked_ioctl = ISP_ioctl,
};
```

DIFUZE



Kernel Source Code
Interface Recovery
Build System Instrument
ioctl Handler Identification
Device File Detection
Command Value Determination
Argument Type Identification
Finding the Structure Definition
Analysis Host
XML Spec.
Structure Generation
Type-Specific Value Creation
Sub-structure Generation
Analysis Host
Fuzz Unit
On-device Execution
Pointer Fixup
Structure Recursion
Execution (and Automatic Reboot)
Target Host
Backtraces to Record Vulnerabilities Being Triggered

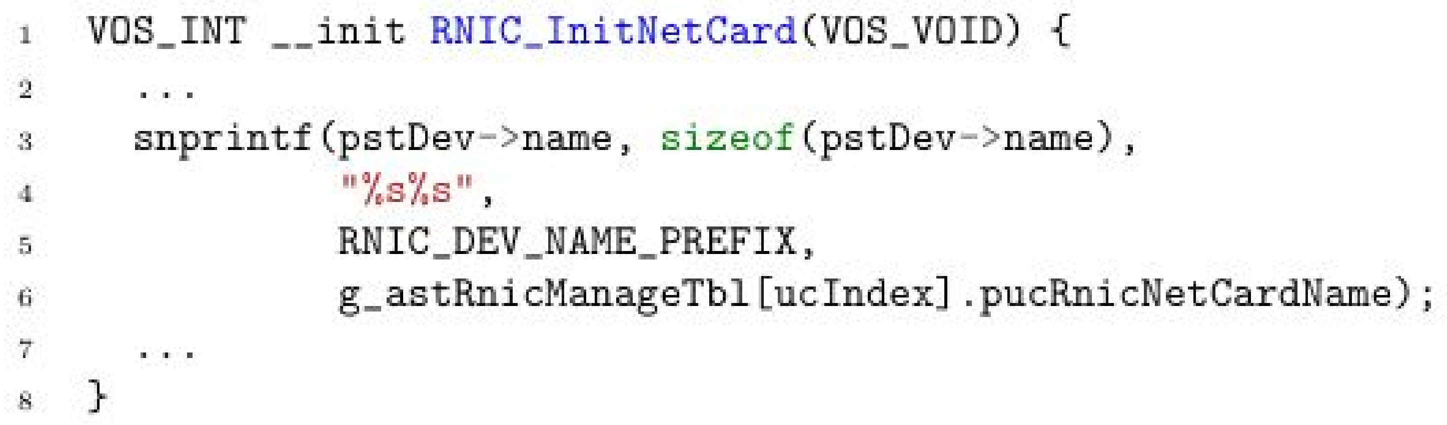
Device File Detection



```
root@F3116:/ # ls -l /dev/ | grep "camera-isp"
crw-rw---- system   camera   243,   0 2017-05-18 08:43 camera-isp
```

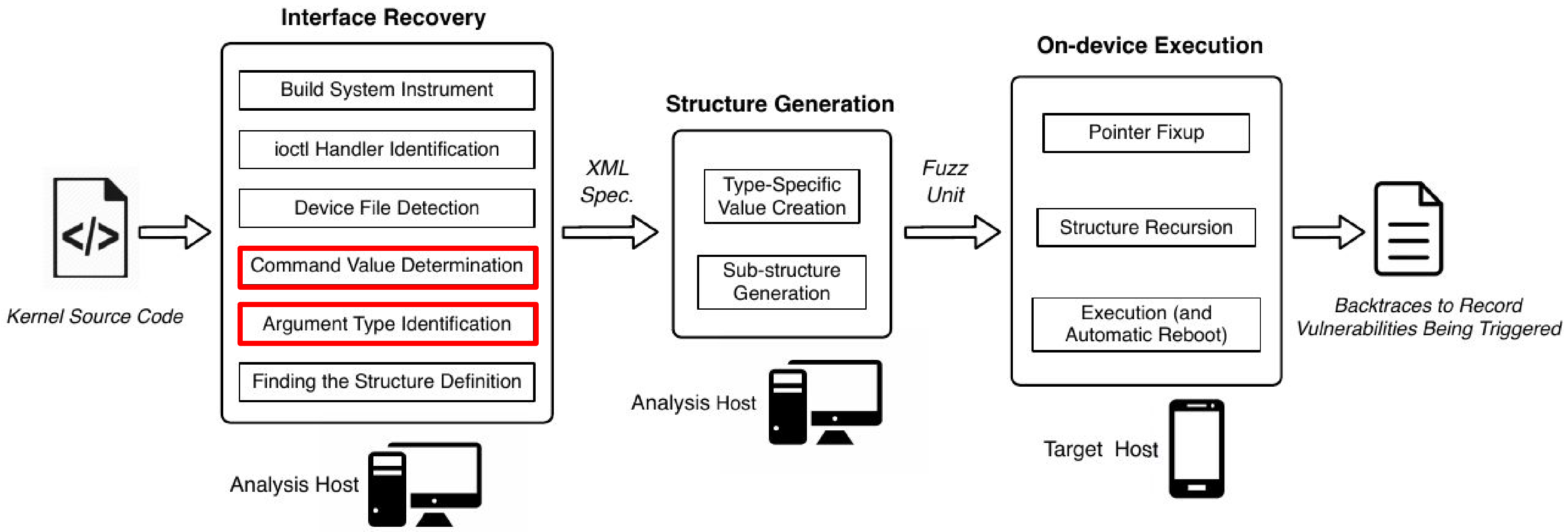
- Link file_operations to device name.
- Techniques differ across device types.
- Cannot handle dynamic names.

Device File Detection Limitation: Dynamic Names



```
VOS_INT __init RNIC_InitNetCard(VOS_VOID) {
  ...
  snprintf(pstDev->name, sizeof(pstDev->name),
           "%s%s",
           RNIC_DEV_NAME_PREFIX,
           g_astRnicManageTbl[ucIndex].pucRnicNetCardName);
  ...
}
```

DIFUZE


Interface Recovery
Build System Instrument
ioctl Handler Identification
Device File Detection
Command Value Determination
Argument Type Identification
Finding the Structure Definition
Analysis Host
Kernel Source Code
XML Spec.
Structure Generation
Type-Specific Value Creation
Sub-structure Generation
Analysis Host
Fuzz Unit
On-device Execution
Pointer Fixup
Structure Recursion
Execution (and Automatic Reboot)
Target Host
Backtraces to Record Vulnerabilities Being Triggered

Command Value + Param Type Recovery

- Follow all paths from the start of ioctl function to all invocations of copy_from_user:
  - Collect constraints on the value of command argument.
  - Collect type information of the destination argument of copy_from_user where source argument is param.
  - Inter-procedural

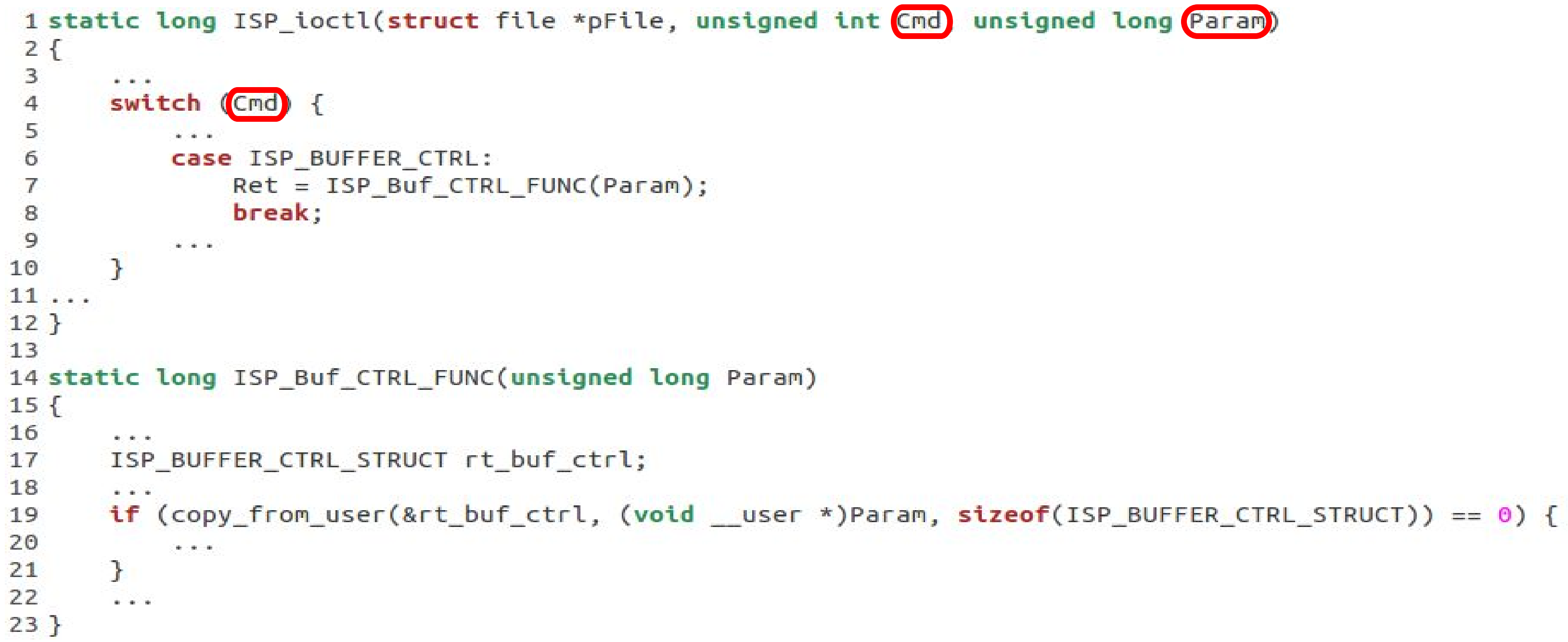
- Possible values and types:
  - (All command constrains) X (All types at copy_from_user).

Command Value + Param Type Recovery

```c
static long ISP_ioctl(struct file *pFile, unsigned int Cmd, unsigned long Param)
{
    ...
    switch (Cmd) {
        ...
        case ISP_BUFFER_CTRL:
            Ret = ISP_Buf_CTRL_FUNC(Param);
            break;
        ...
    }
...
}

static long ISP_Buf_CTRL_FUNC(unsigned long Param)
{
    ...
    ISP_BUFFER_CTRL_STRUCT rt_buf_ctrl;
    ...
    if (copy_from_user(&rt_buf_ctrl, (void __user *)Param, sizeof(ISP_BUFFER_CTRL_STRUCT)) == 0) {
        ...
    }
    ...
}
```

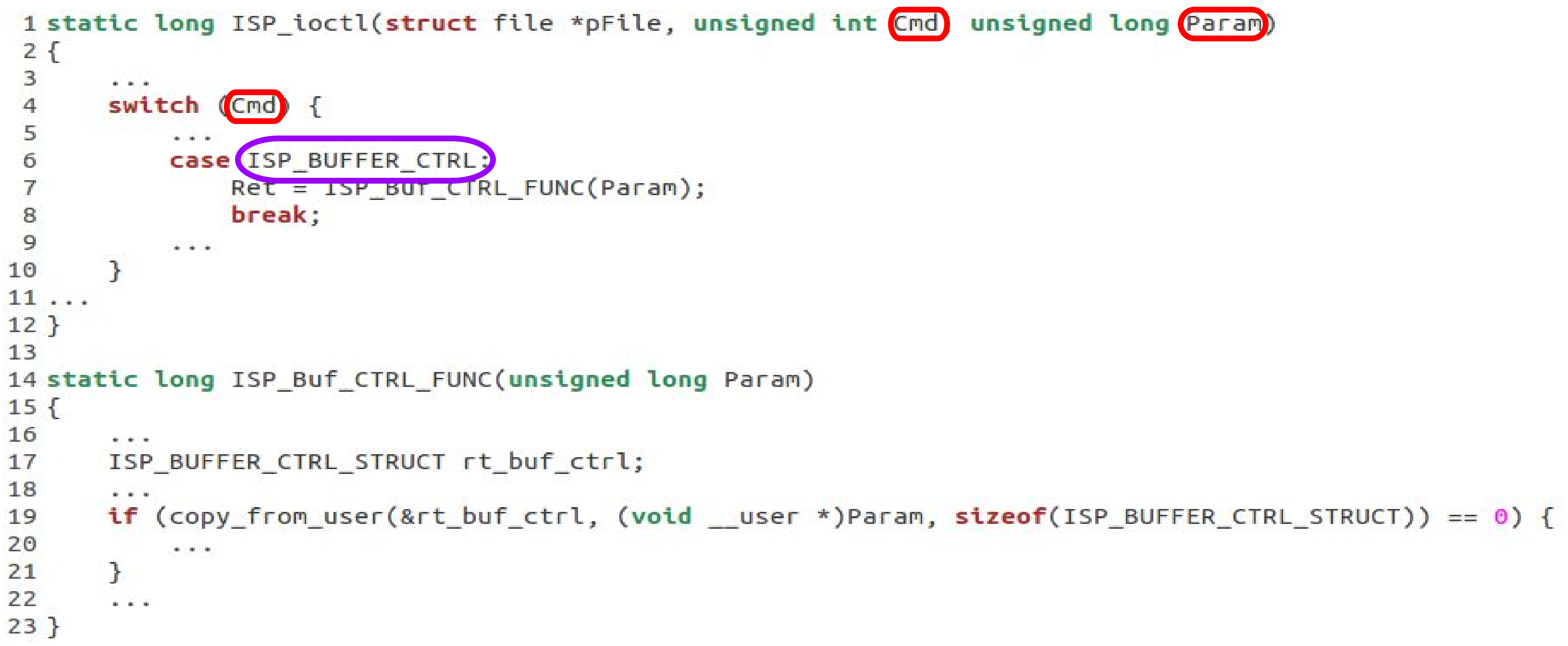
Command Value + Param Type Recovery



```c
static long ISP_ioctl(struct file *pFile, unsigned int Cmd, unsigned long Param)
{
    ...
    switch (Cmd) {
        ...
        case ISP_BUFFER_CTRL:
            Ret = ISP_Buf_CTRL_FUNC(Param);
            break;
        ...
    }
...
}

static long ISP_Buf_CTRL_FUNC(unsigned long Param)
{
    ...
    ISP_BUFFER_CTRL_STRUCT rt_buf_ctrl;
    ...
    if (copy_from_user(&rt_buf_ctrl, (void __user *)Param, sizeof(ISP_BUFFER_CTRL_STRUCT)) == 0) {
        ...
    }
    ...
}
```

Command Value + Param Type Recovery

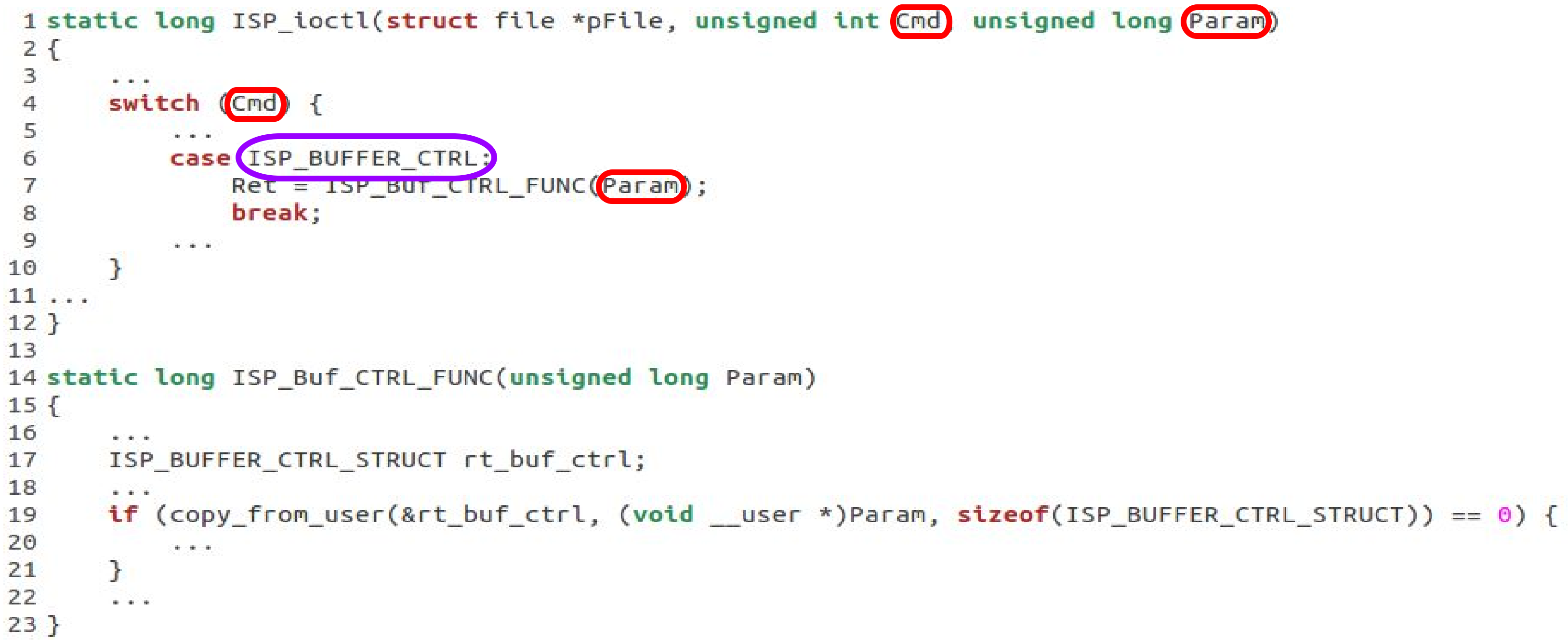


```c
static long ISP_ioctl(struct file *pFile, unsigned int Cmd, unsigned long Param)
{
    ...
    switch (Cmd) {
        ...
        case ISP_BUFFER_CTRL:
            Ret = ISP_Buf_CTRL_FUNC(Param);
            break;
        ...
    }
...
}

static long ISP_Buf_CTRL_FUNC(unsigned long Param)
{
    ...
    ISP_BUFFER_CTRL_STRUCT rt_buf_ctrl;
    ...
    if (copy_from_user(&rt_buf_ctrl, (void __user *)Param, sizeof(ISP_BUFFER_CTRL_STRUCT)) == 0) {
        ...
    }
    ...
}
```

Command Value: ISP_BUFFER_CTRL

Command Value + Param Type Recovery

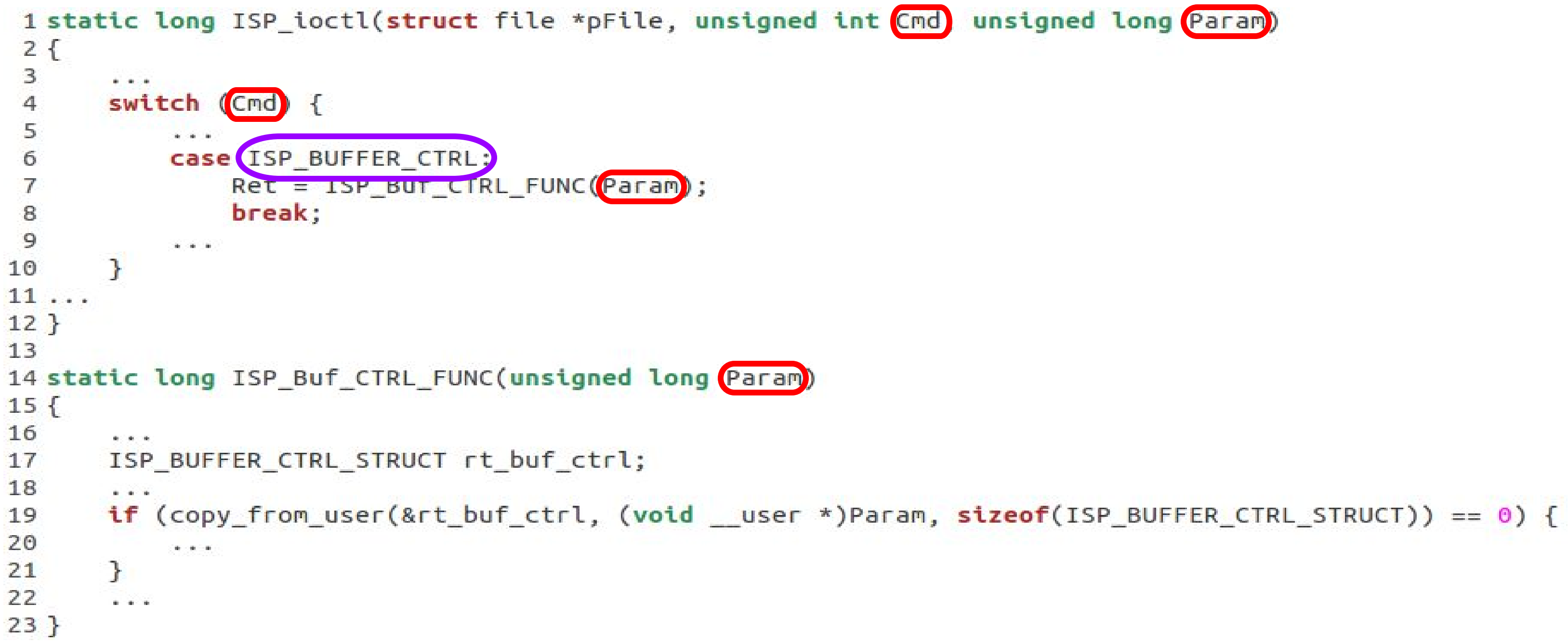


```c
static long ISP_ioctl(struct file *pFile, unsigned int Cmd, unsigned long Param)
{
    ...
    switch (Cmd) {
        ...
        case ISP_BUFFER_CTRL:
            Ret = ISP_Buf_CTRL_FUNC(Param);
            break;
        ...
    }
...
}

static long ISP_Buf_CTRL_FUNC(unsigned long Param)
{
    ...
    ISP_BUFFER_CTRL_STRUCT rt_buf_ctrl;
    ...
    if (copy_from_user(&rt_buf_ctrl, (void __user *)Param, sizeof(ISP_BUFFER_CTRL_STRUCT)) == 0) {
        ...
    }
    ...
}
```

Command Value: ISP_BUFFER_CTRL

Command Value + Param Type Recovery

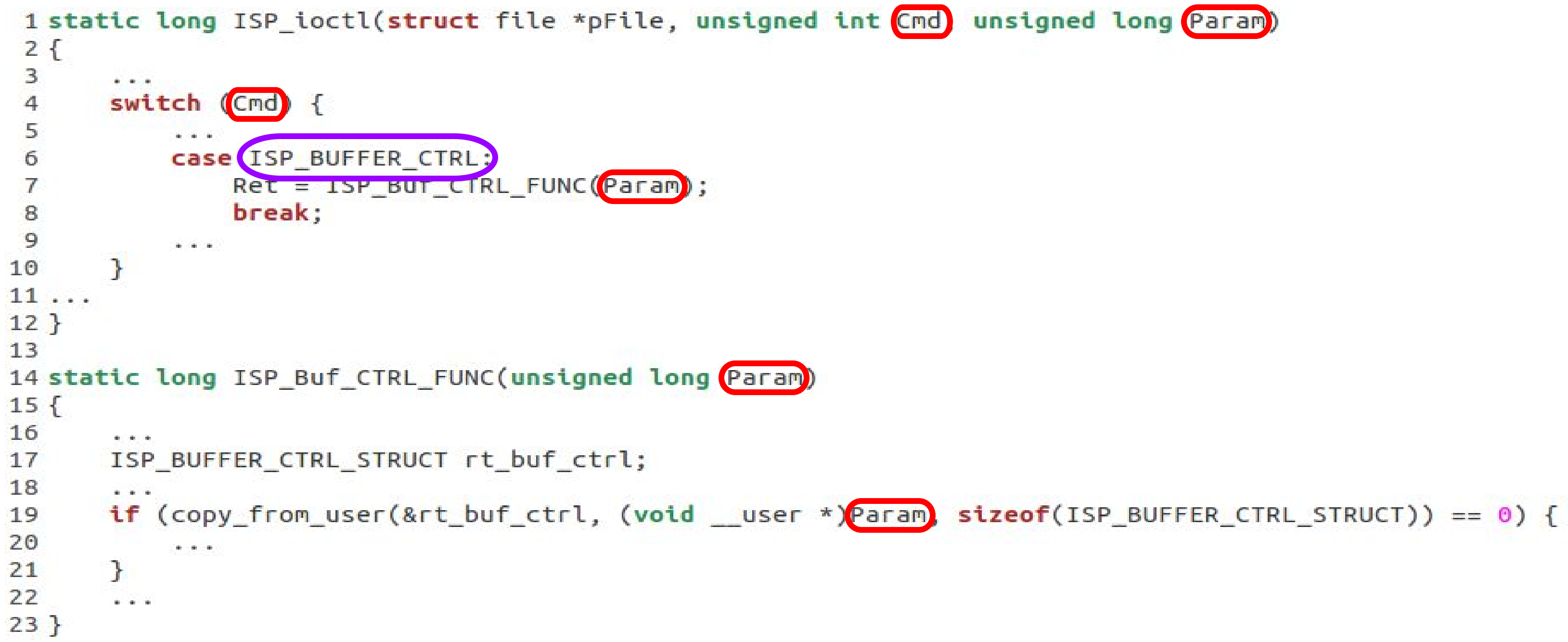


```c
static long ISP_ioctl(struct file *pFile, unsigned int Cmd, unsigned long Param)
{
    ...
    switch (Cmd) {
        ...
        case ISP_BUFFER_CTRL:
            Ret = ISP_Buf_CTRL_FUNC(Param);
            break;
        ...
    }
...
}

static long ISP_Buf_CTRL_FUNC(unsigned long Param)
{
    ...
    ISP_BUFFER_CTRL_STRUCT rt_buf_ctrl;
    ...
    if (copy_from_user(&rt_buf_ctrl, (void __user *)Param, sizeof(ISP_BUFFER_CTRL_STRUCT)) == 0) {
        ...
    }
    ...
}
```

Command Value: ISP_BUFFER_CTRL

Command Value + Param Type Recovery

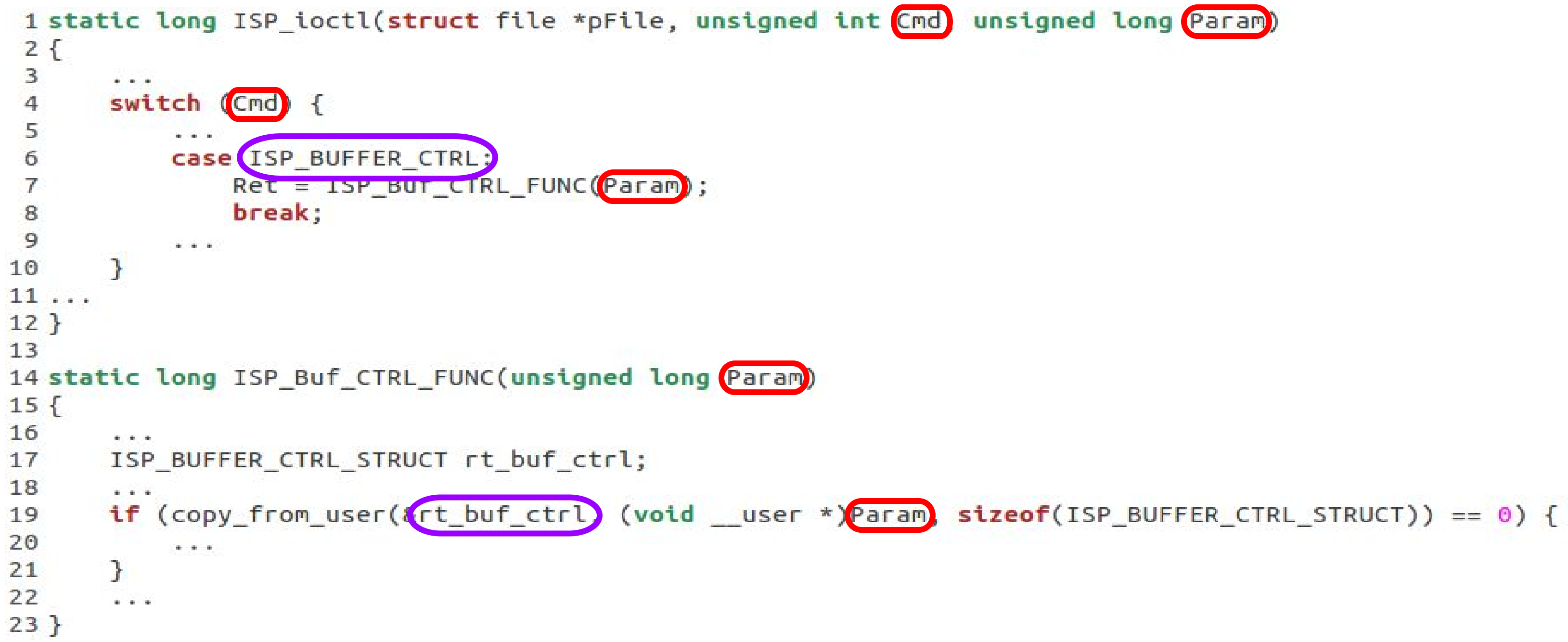


```c
static long ISP_ioctl(struct file *pFile, unsigned int Cmd, unsigned long Param)
{
    ...
    switch (Cmd) {
        ...
        case ISP_BUFFER_CTRL:
            Ret = ISP_Buf_CTRL_FUNC(Param);
            break;
        ...
    }
...
}

static long ISP_Buf_CTRL_FUNC(unsigned long Param)
{
    ...
    ISP_BUFFER_CTRL_STRUCT rt_buf_ctrl;
    ...
    if (copy_from_user(&rt_buf_ctrl, (void __user *)Param, sizeof(ISP_BUFFER_CTRL_STRUCT)) == 0) {
        ...
    }
    ...
}
```

Command Value: ISP_BUFFER_CTRL

Command Value + Param Type Recovery

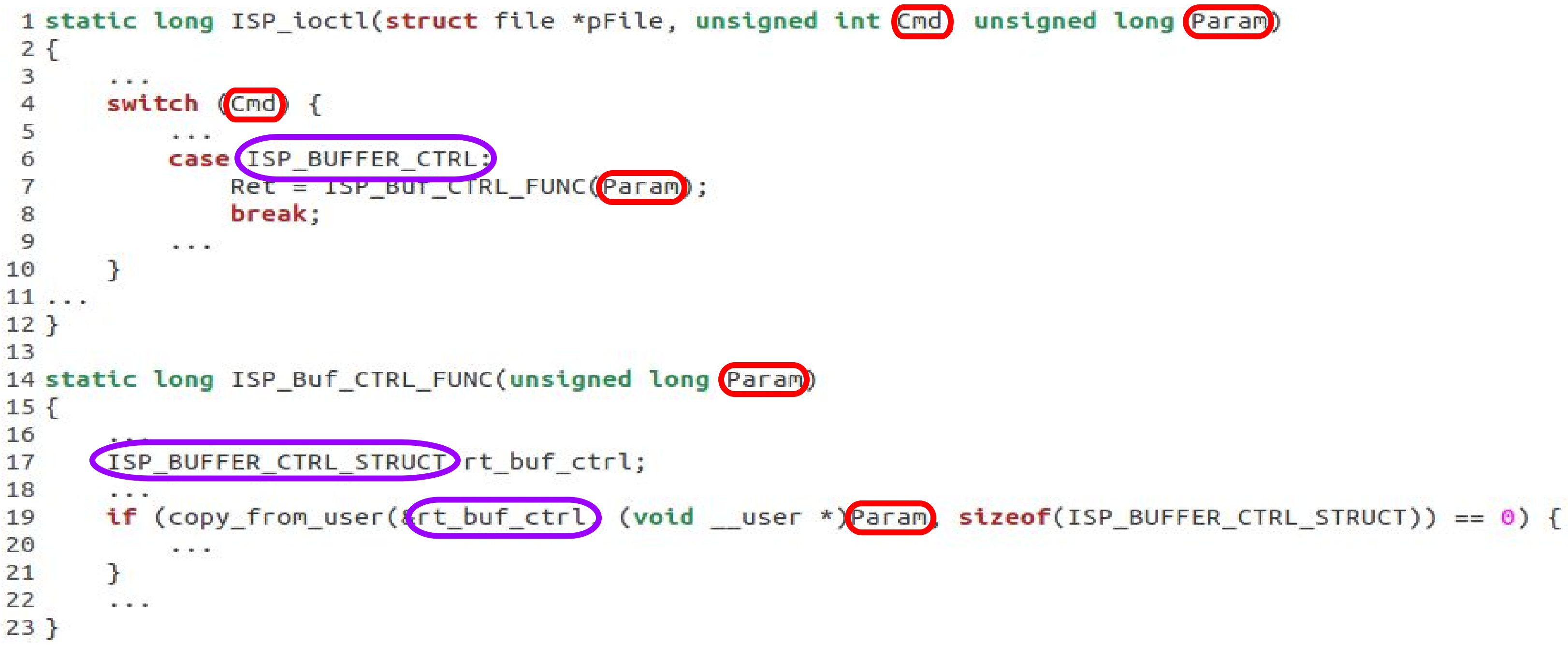


```
static long ISP_ioctl(struct file *pFile, unsigned int Cmd, unsigned long Param)
{
    ...
    switch (Cmd) {
        ...
        case ISP_BUFFER_CTRL:
            Ret = ISP_Buf_CTRL_FUNC(Param);
            break;
        ...
    }
...
}

static long ISP_Buf_CTRL_FUNC(unsigned long Param)
{
    ...
    ISP_BUFFER_CTRL_STRUCT rt_buf_ctrl;
    ...
    if (copy_from_user(&rt_buf_ctrl, (void __user *)Param, sizeof(ISP_BUFFER_CTRL_STRUCT)) == 0) {
        ...
    }
    ...
}
```

Command Value: ISP_BUFFER_CTRL

Command Value + Param Type Recovery

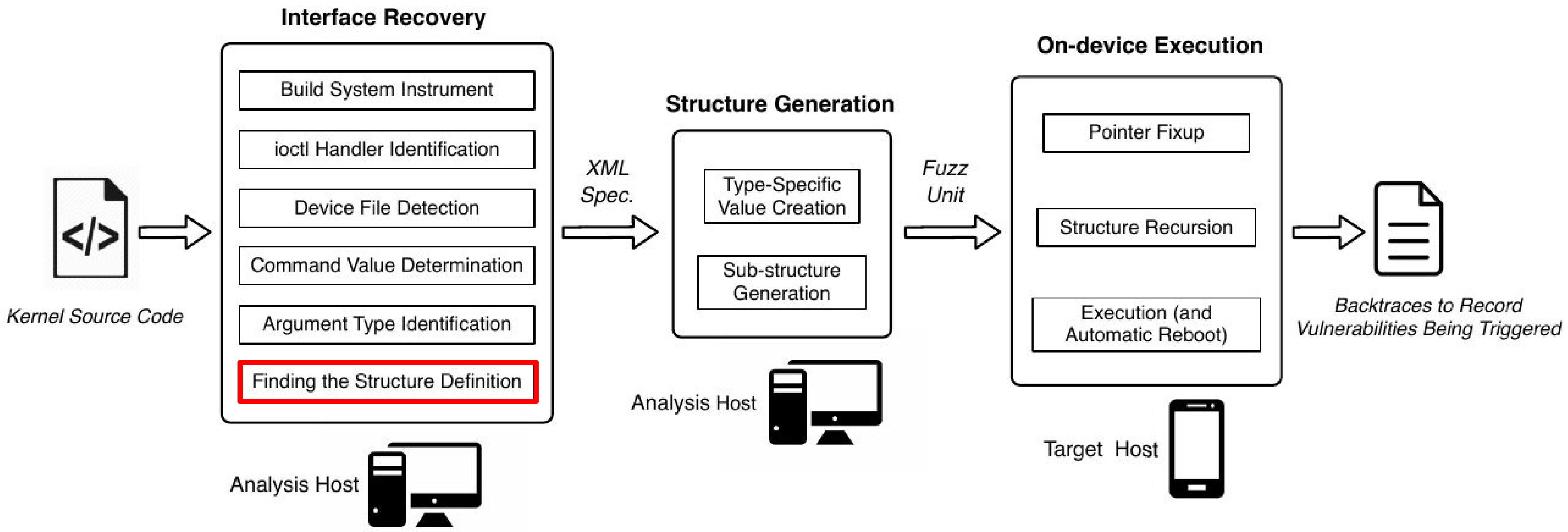


```
static long ISP_ioctl(struct file *pFile, unsigned int Cmd, unsigned long Param)
{
    ...
    switch (Cmd) {
        ...
        case ISP_BUFFER_CTRL:
            Ret = ISP_Buf_CTRL_FUNC(Param);
            break;
        ...
    }
...
}

static long ISP_Buf_CTRL_FUNC(unsigned long Param)
{
    ...
    ISP_BUFFER_CTRL_STRUCT rt_buf_ctrl;
    ...
    if (copy_from_user(&rt_buf_ctrl, (void __user *)Param, sizeof(ISP_BUFFER_CTRL_STRUCT)) == 0) {
        ...
    }
    ...
}
```

Command Value: ISP_BUFFER_CTRL
Type: ISP_BUFFER_CTRL_STRUCT

DIFUZE


Interface Recovery
Build System Instrument
ioctl Handler Identification
Device File Detection
Command Value Determination
Argument Type Identification
Finding the Structure Definition
Analysis Host
Kernel Source Code
XML Spec.
Structure Generation
Type-Specific Value Creation
Sub-structure Generation
Analysis Host
Fuzz Unit
On-device Execution
Pointer Fixup
Structure Recursion
Execution (and Automatic Reboot)
Target Host
Backtraces to Record Vulnerabilities Being Triggered

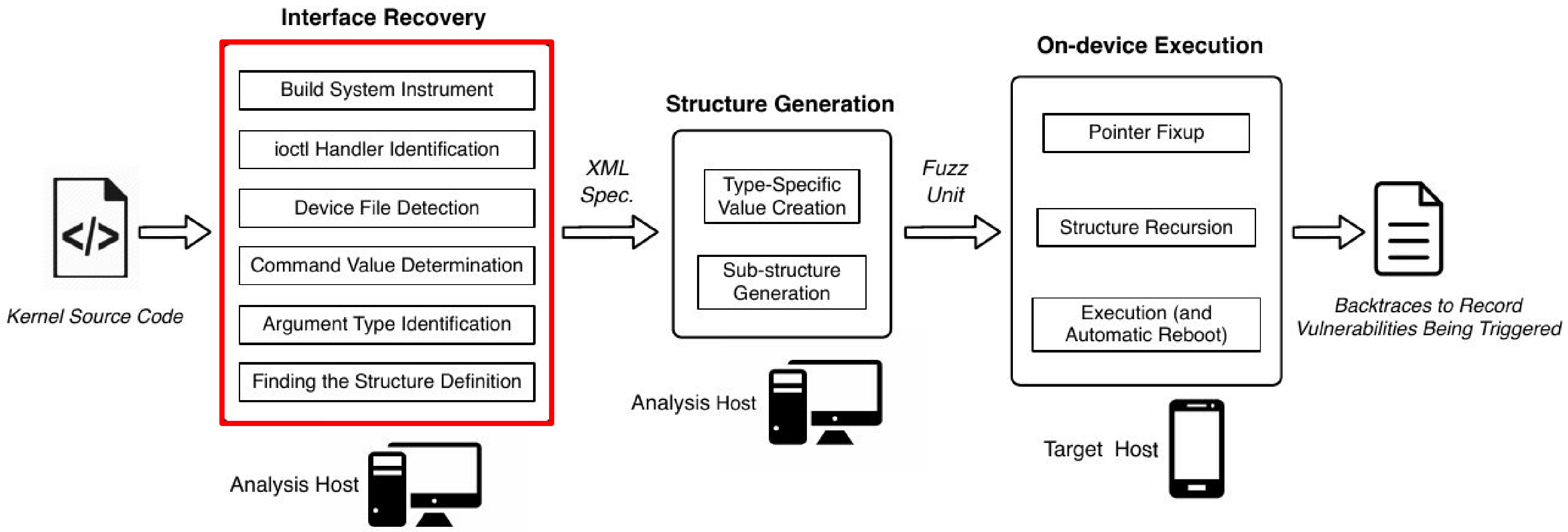
Structure Definition Recovery

- Identify the source file(s) for the driver.

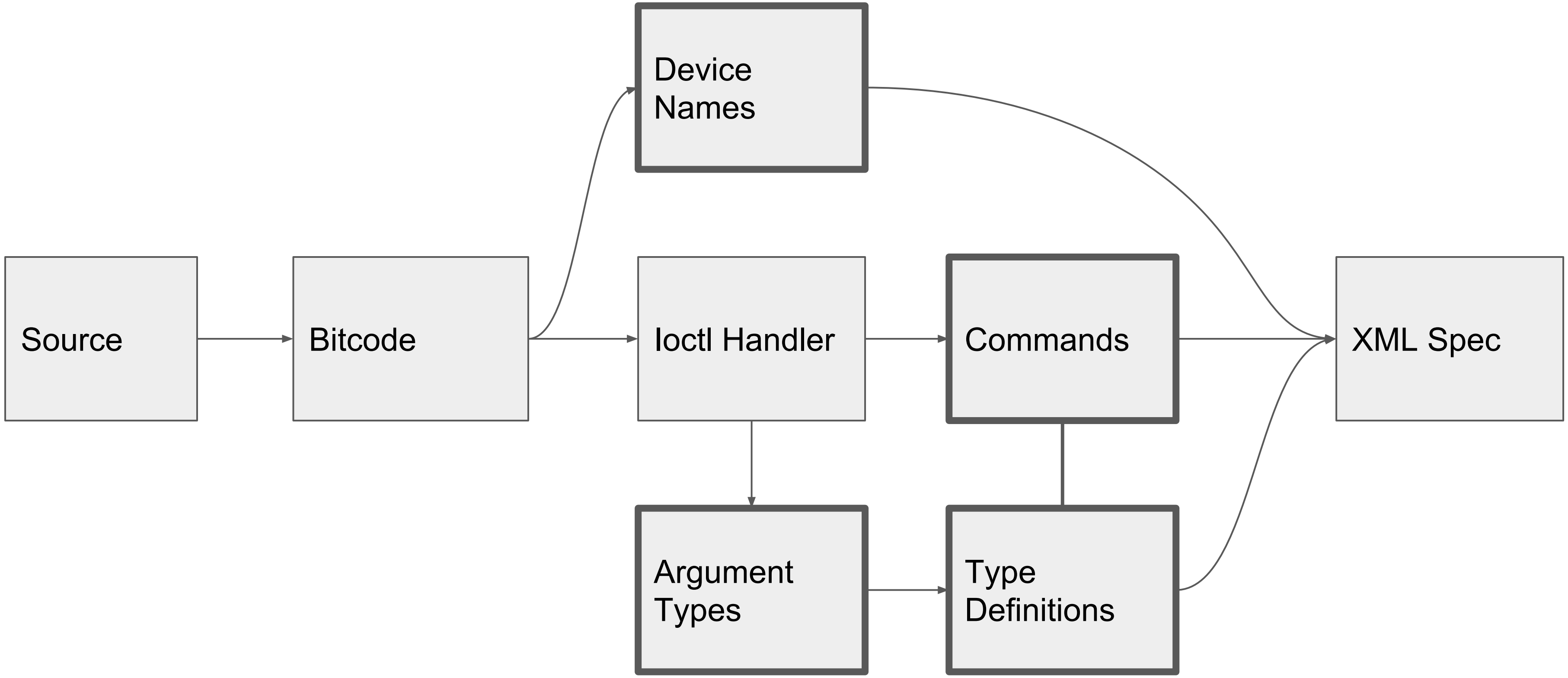
- Generate pre-processed files.

- Use c2xml to convert into XML and extract the structures:
    - Need to consider padding, Recursive structures, etc.

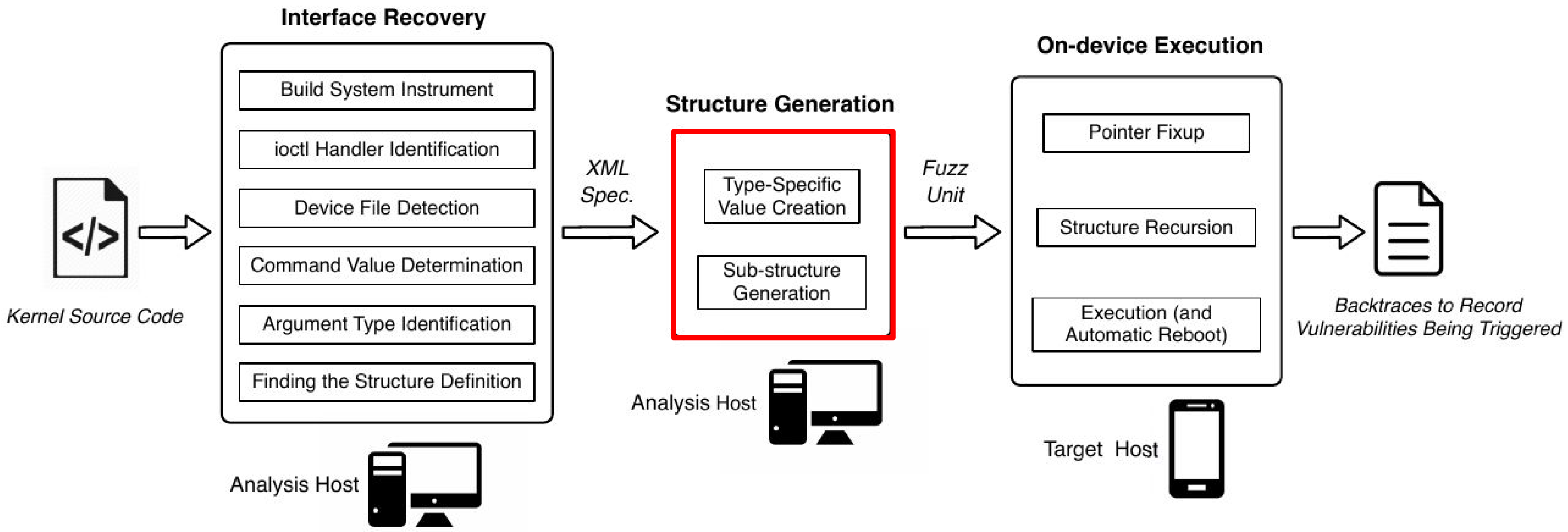
DIFUZE


Interface Recovery
Build System Instrument
ioctl Handler Identification
Device File Detection
Command Value Determination
Argument Type Identification
Finding the Structure Definition
Analysis Host
Kernel Source Code
XML Spec.
Structure Generation
Type-Specific Value Creation
Sub-structure Generation
Analysis Host
Fuzz Unit
On-device Execution
Pointer Fixup
Structure Recursion
Execution (and Automatic Reboot)
Target Host
Backtraces to Record Vulnerabilities Being Triggered

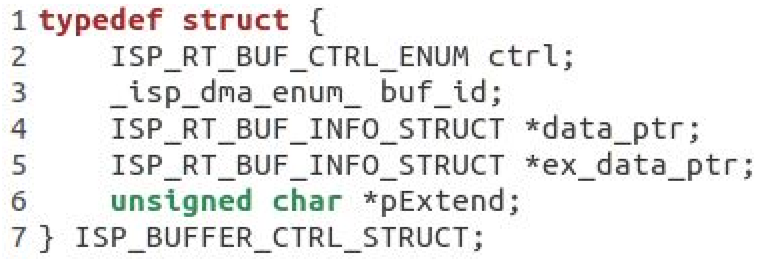
Interface Recovery



DIFUZE

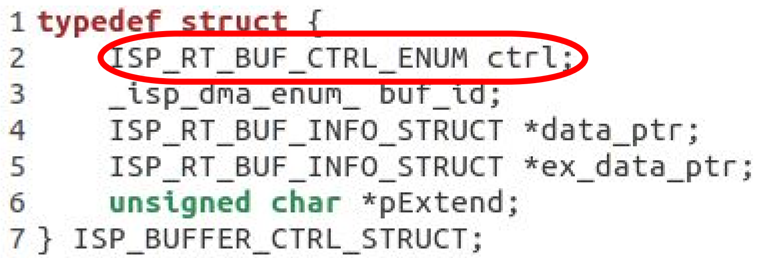

Interface Recovery
Build System Instrument
ioctl Handler Identification
Device File Detection
Command Value Determination
Argument Type Identification
Finding the Structure Definition
Kernel Source Code
Analysis Host
XML Spec.
Structure Generation
Type-Specific Value Creation
Sub-structure Generation
Analysis Host
Fuzz Unit
On-device Execution
Pointer Fixup
Structure Recursion
Execution (and Automatic Reboot)
Target Host
Backtraces to Record Vulnerabilities Being Triggered

Structure Generation

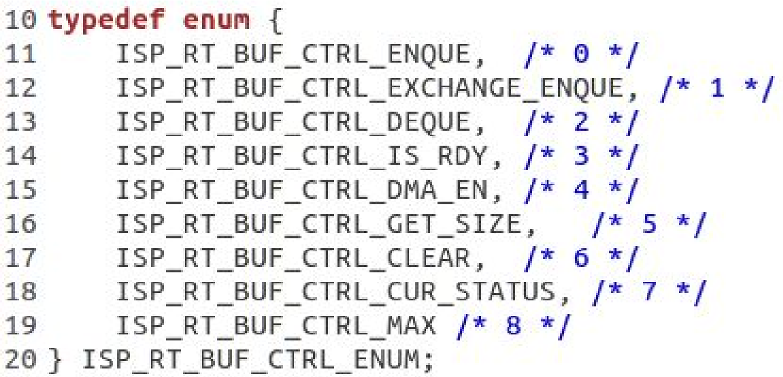


```c
typedef struct {
    ISP_RT_BUF_CTRL_ENUM ctrl;
    _isp_dma_enum_ buf_id;
    ISP_RT_BUF_INFO_STRUCT *data_ptr;
    ISP_RT_BUF_INFO_STRUCT *ex_data_ptr;
    unsigned char *pExtend;
} ISP_BUFFER_CTRL_STRUCT;
```

Structure Generation

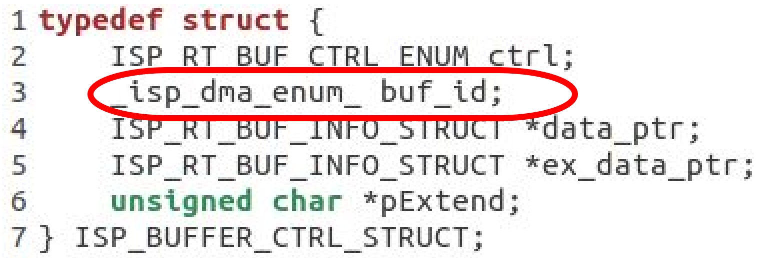


```c
typedef struct {
    ISP_RT_BUF_CTRL_ENUM ctrl;
    _isp_dma_enum_ buf_id;
    ISP_RT_BUF_INFO_STRUCT *data_ptr;
    ISP_RT_BUF_INFO_STRUCT *ex_data_ptr;
    unsigned char *pExtend;
} ISP_BUFFER_CTRL_STRUCT;
```

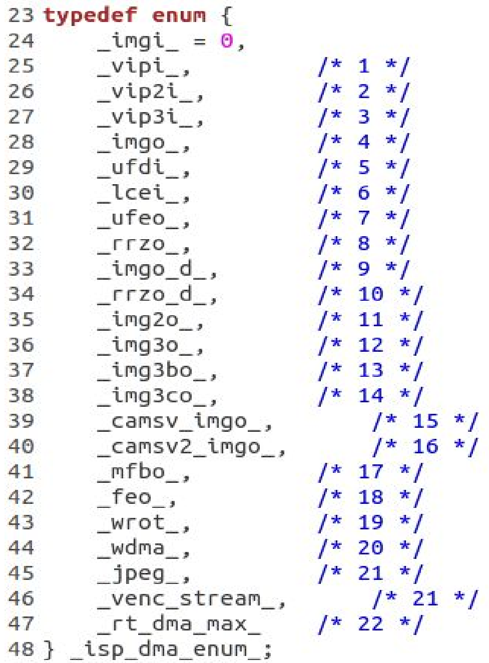


```c
typedef enum {
    ISP_RT_BUF_CTRL_ENQUE,  /* 0 */
    ISP_RT_BUF_CTRL_EXCHANGE_ENQUE, /* 1 */
    ISP_RT_BUF_CTRL_DEQUE,  /* 2 */
    ISP_RT_BUF_CTRL_IS_RDY, /* 3 */
    ISP_RT_BUF_CTRL_DMA_EN, /* 4 */
    ISP_RT_BUF_CTRL_GET_SIZE,   /* 5 */
    ISP_RT_BUF_CTRL_CLEAR,  /* 6 */
    ISP_RT_BUF_CTRL_CUR_STATUS, /* 7 */
    ISP_RT_BUF_CTRL_MAX /* 8 */
} ISP_RT_BUF_CTRL_ENUM;
```

Structure Generation

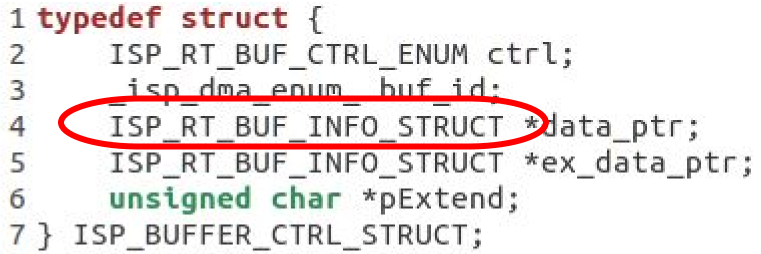


```c
typedef struct {
    ISP_RT_BUF_CTRL_ENUM ctrl;
    _isp_dma_enum_ buf_id;
    ISP_RT_BUF_INFO_STRUCT *data_ptr;
    ISP_RT_BUF_INFO_STRUCT *ex_data_ptr;
    unsigned char *pExtend;
} ISP_BUFFER_CTRL_STRUCT;
```

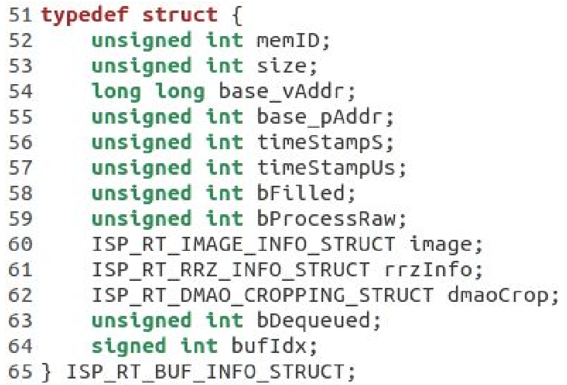


```c
typedef enum {
    _imgi_ = 0,
    _vipi_,             /* 1 */
    _vip2i_,            /* 2 */
    _vip3i_,            /* 3 */
    _imgo_,             /* 4 */
    _ufdi_,             /* 5 */
    _lcei_,             /* 6 */
    _ufeo_,             /* 7 */
    _rrzo_,             /* 8 */
    _imgo_d_,           /* 9 */
    _rrzo_d_,           /* 10 */
    _img2o_,            /* 11 */
    _img3o_,            /* 12 */
    _img3bo_,           /* 13 */
    _img3co_,           /* 14 */
    _camsv_imgo_,           /* 15 */
    _camsv2_imgo_,          /* 16 */
    _mfbo_,             /* 17 */
    _feo_,              /* 18 */
    _wrot_,             /* 19 */
    _wdma_,             /* 20 */
    _jpeg_,             /* 21 */
    _venc_stream_,          /* 21 */
    _rt_dma_max_        /* 22 */
} _isp_dma_enum_;
```

Structure Generation

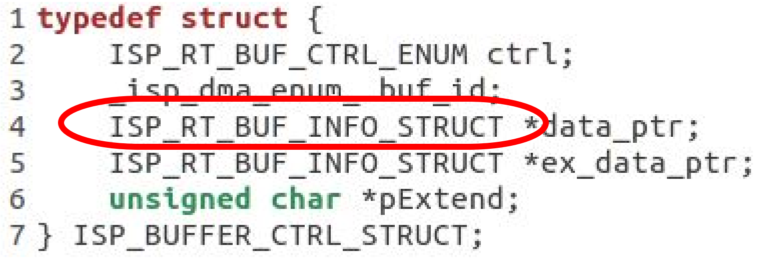


```c
typedef struct {
    ISP_RT_BUF_CTRL_ENUM ctrl;
    _isp_dma_enum_ buf_id;
    ISP_RT_BUF_INFO_STRUCT *data_ptr;
    ISP_RT_BUF_INFO_STRUCT *ex_data_ptr;
    unsigned char *pExtend;
} ISP_BUFFER_CTRL_STRUCT;
```

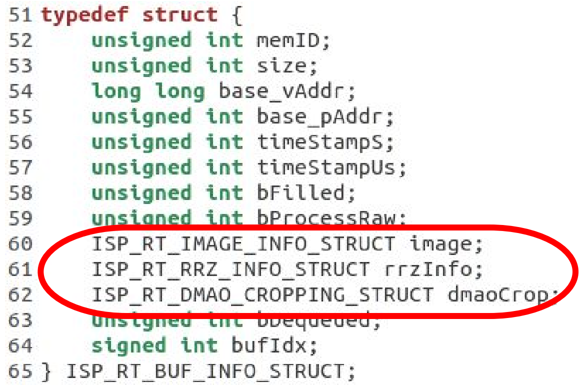


```c
typedef struct {
    unsigned int memID;
    unsigned int size;
    long long base_vAddr;
    unsigned int base_pAddr;
    unsigned int timeStampS;
    unsigned int timeStampUs;
    unsigned int bFilled;
    unsigned int bProcessRaw;
    ISP_RT_IMAGE_INFO_STRUCT image;
    ISP_RT_RRZ_INFO_STRUCT rrzInfo;
    ISP_RT_DMAO_CROPPING_STRUCT dmaoCrop;
    unsigned int bDequeued;
    signed int bufIdx;
} ISP_RT_BUF_INFO_STRUCT;
```

Structure Generation

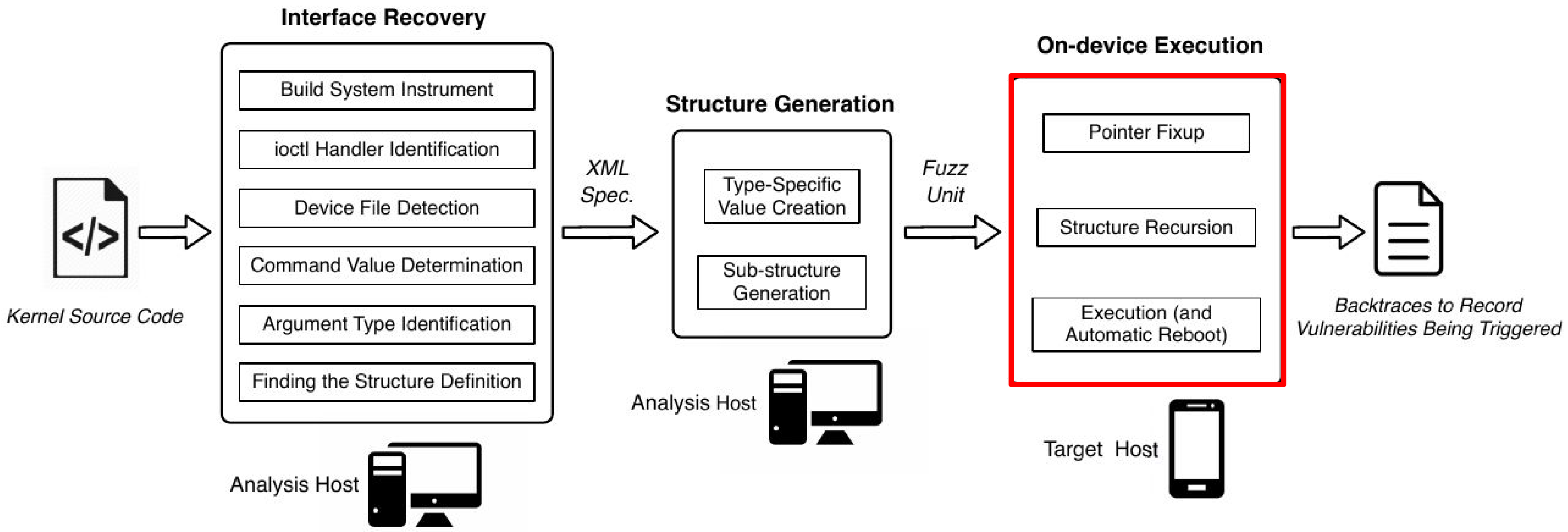


```c
typedef struct {
    ISP_RT_BUF_CTRL_ENUM ctrl;
    _isp_dma_enum_ buf_id;
    ISP_RT_BUF_INFO_STRUCT *data_ptr;
    ISP_RT_BUF_INFO_STRUCT *ex_data_ptr;
    unsigned char *pExtend;
} ISP_BUFFER_CTRL_STRUCT;
```



```c
typedef struct {
    unsigned int memID;
    unsigned int size;
    long long base_vAddr;
    unsigned int base_pAddr;
    unsigned int timeStampS;
    unsigned int timeStampUs;
    unsigned int bFilled;
    unsigned int bProcessRaw;
    ISP_RT_IMAGE_INFO_STRUCT image;
    ISP_RT_RRZ_INFO_STRUCT rrzInfo;
    ISP_RT_DMAO_CROPPING_STRUCT dmaoCrop;
    unsigned int bDequeued;
    signed int bufIdx;
} ISP_RT_BUF_INFO_STRUCT;
```

DIFUZE


Kernel Source Code
Interface Recovery
Build System Instrument
ioctl Handler Identification
Device File Detection
Command Value Determination
Argument Type Identification
Finding the Structure Definition
Analysis Host
XML Spec.
Structure Generation
Type-Specific Value Creation
Sub-structure Generation
Analysis Host
Fuzz Unit
On-device Execution
Pointer Fixup
Structure Recursion
Execution (and Automatic Reboot)
Target Host
Backtraces to Record Vulnerabilities Being Triggered

On Device Execution

- Run on the phone connected to host device via ADB (Android Debug Bridge).
- Map the binary data, do pointer fix ups.
- Open device and perform the ioctl.

Evaluation

Manufacturer	**Device**	**Chipset**
Google	Pixel	Qualcomm
HTC	E9 Plus	Mediatek
HTC	One M9	Qualcomm
Huawei	P9 Lite	Huawei
Huawei	Honor 8	Huawei
Samsung	Galaxy S6	Samsung
Sony	Xperia XA	Mediatek

Device Name Recovery

Ioctl Handlers	**Device Names Automatically Identified**
789	469

- ~ 60% effective
- 40% missed mostly because of dynamic device names (mainline kernel drivers)

Type + Command ID Recovery

- 53% of the commands expect the param to be a pointer to some structure.

- 90% accuracy: Random sampling.

Fuzzing

- **syzkaller**.
- **syzkaller + Device Path**.
- **DIFUZE^{i}** : Syzkaller + Device Path + Command IDs.
- **DIFUZE^{s}** : Syzkaller + All interface information.
- **DIFUZE^{m}** : Standalone fuzzer + All interface information.

Fuzzing Results

	syzkaller	syzkaller + path	DIFUZE^i	DIFUZE^s	DIFUZE^m	Total Unique
E9 Plus	0	0	4	6	6	6
Galaxy S6	-	-	-	-	0	0
Honor 8	0	0	1	2	2	2
One M9	0	0	3	3	2	3
P9 Lite	0	0	2	5	5	6
Pixel	0	1	2	5	3	5
Xperia XA	0	2	10	13	12	14
Total	**0**	**3**	**22**	**34**	**30**	**36**

Fuzzing Results

	syzkaller	syzkaller + path	DIFUZE^i	DIFUZE^s	DIFUZE^m	Total Unique
E9 Plus	0	0	4	6	6	6
Galaxy S6	-	-	-	-	0	0
Honor 8	0	0	1	2	2	2
One M9	0	0	3	3	2	3
P9 Lite	0	0	2	5	5	6
Pixel	0	1	2	5	3	5
Xperia XA	0	2	10	13	12	14
Total	**0**	**3**	**22**	**34**	**30**	**36**

Bug Types

Crash Type	Count
Arbitrary Read	4
Arbitrary Write	4
Assert Failure	6
Buffer Overflow	2
Null Dereference	9
Out of Bound Index	5
Uncategorized	5
Writing to non-volatile memory	1

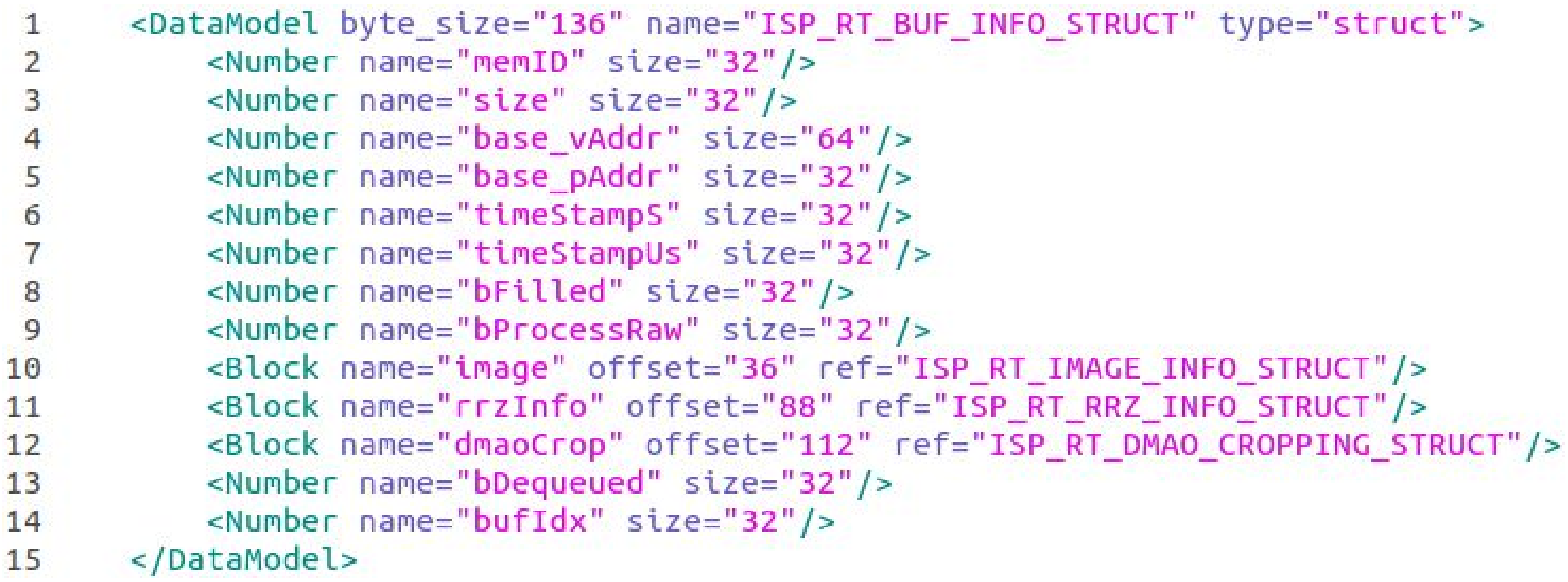
Conclusions

✓ Method to extract interface information from driver source code.

✓ Interface information can improve the effective of kernel driver fuzzing.

✓ https://github.com/ucsb-seclab/difuze

Structure Definition Recovery



```xml
<DataModel byte_size="136" name="ISP_RT_BUF_INFO_STRUCT" type="struct">
    <Number name="memID" size="32"/>
    <Number name="size" size="32"/>
    <Number name="base_vAddr" size="64"/>
    <Number name="base_pAddr" size="32"/>
    <Number name="timeStampS" size="32"/>
    <Number name="timeStampUs" size="32"/>
    <Number name="bFilled" size="32"/>
    <Number name="bProcessRaw" size="32"/>
    <Block name="image" offset="36" ref="ISP_RT_IMAGE_INFO_STRUCT"/>
    <Block name="rrzInfo" offset="88" ref="ISP_RT_RRZ_INFO_STRUCT"/>
    <Block name="dmaoCrop" offset="112" ref="ISP_RT_DMAO_CROPPING_STRUCT"/>
    <Number name="bDequeued" size="32"/>
    <Number name="bufIdx" size="32"/>
</DataModel>
```

XML Spec (jpit)

```xml
<Mango author="jay` bot" description="autogenerated jpit" version="1.0">

    <Config>
        <devname value="/dev/camera-isp"/>
        <ioctl_id value="2148559647"/>
        <target_struct value="ISP_REGISTER_USERKEY_STRUCT"/>
    </Config>

    <DataModel byte_size="16" name="ISP_REGISTER_USERKEY_STRUCT" type="struct">
        <Number name="userKey" size="32"/>
        <String length="4" name="padding256"/>
        <Pointer base="char" elem_size="1" length="8" name="userName" offset="8" ptr_depth="1" ptr_to="String"/>
    </DataModel>

</Mango>
```