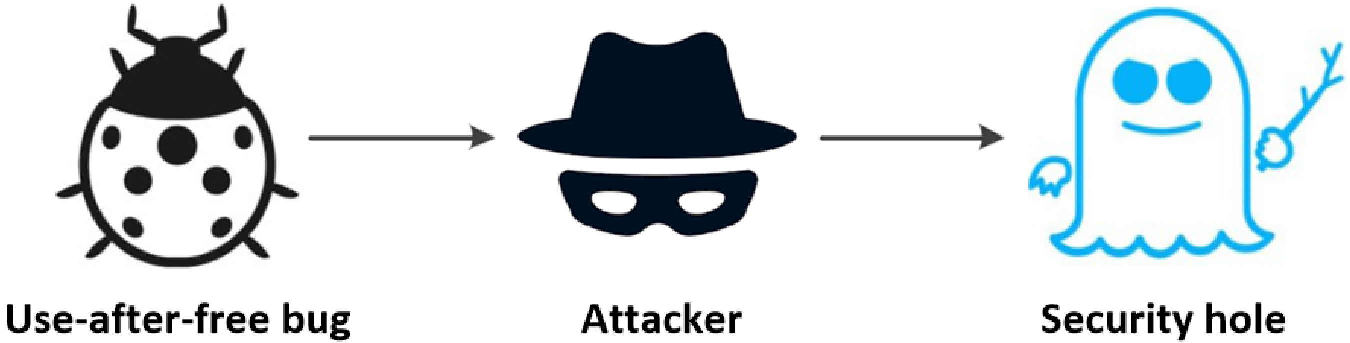
Effective Static Analysis of Concurrency Use-After-Free Bugs in Linux Device Drivers

Jia-Ju Bai[1], Julia Lawall[2], Qiu-Liang Chen[1], Shi-Min Hu[1]

[1]Tsinghua University, [2]Sorbonne University/Inria/LIP6

Background

- Use-after-free bugs in device drivers
  - Reliability: may cause system crashes
  - Security: can be exploited to attack the operating system



Background

- Sequential use-after-free bug

```
void DriverExit(struct device *pdev) {
    kfree(pdev->buf);
    pdev->num = 0;
    pdev->buf->last = NULL;
}
```

Thread 1

- Concurrency use-after-free bug

```
void DriverFunc1(struct device *pdev) {
    kfree(pdev->buf);
    pdev->buf = kmalloc(...)
    pdev->buf->last = NULL;
}
```

Thread 1

```
void DriverFunc2(struct device *pdev) {
    spin_lock(...);
    pdev->buf->first = NULL;
    spin_unlock(...);
}
```

Thread 2

Background

- Concurrency use-after-free bugs in device drivers
  - Caused by driver concurrency
  - Hard to trigger and reproduce at runtime
  - Lead to system crashes or security problems

Study of Linux kernel commits

- Use-after-free commits (Jan.2016~Dec.2018 (3 years))

Commits	Drivers	Concurrency
949	461	195 (42%)

- Mentioned tools in use-after-free commits

Tool type	Commits	Concurrency
Runtime analysis	120	56
Static analysis	7	0

It is important to explore static analysis to detect concurrency use-after-free bugs in device drivers!

Challenges for static analysis

- Identify driver functions that can be concurrently executed
  - Poor documentation about driver concurrency
  - Many functions defined in the driver code

- Accuracy and efficiency of code analysis
  - Large size of the Linux driver code base
  - Many function calls across different source files

Approach

- DCUAF
  - ***Local-global strategy:*** extract driver functions that may be concurrently executed
  - ***Summary-based lockset analysis:*** detect concurrency use-after-free bugs

Effective Static Analysis of Concurrency Use-After-Free Bugs in Linux Device Drivers

Wednesday, July 10, 4:10pm

Track II: Security #1: Kernel

Hope to see you at our presentation!