Mitigating Security Risks in Linux with KLAUS

A Method for Evaluating Patch Correctness

Yuhang Wu, Zhenpeng Lin, Yueqi Chen, Dang K Le,
Dongliang Mu, Xinyu Xing
Linux Patching in the Fuzzing Era: Navigating a Bug Surge

Syzbot has verified ~5600 valid bug reports in 5 years

~4600 patches were developed to fix these bugs
Linux Patching in the Fuzzing Era: Navigating a Bug Surge

Noticeable increase in both bug reports and bug fixes

Bug lifetimes of Linux kernel (https://syzkaller.appspot.com/upstream/graph/lifetimes)
Linux Patching in the Fuzzing Era: Navigating a Bug Surge

More bugs
More experts
More patches

... but

~ 6% of the Linux kernel patches are incorrect
The Pitfalls of Incorrect Patching

Root Causes of Incorrect Patches

❖ Lack of understanding of the code
❖ Misdiagnosis of the root cause of the bug

Common Patching Mistakes

❖ Not considering all potential branches or pathways that lead to the patched site
❖ Adding insufficient sanitize checks
A Real-World Example

**Initial UAF:** Dangling pointer in timer queue after `sys_disconnect`

**Incorrect patch:** line 8, 9 are deleted in the patch

**New UAF:** Dangling pointer left after `sk` has been cloned and `sys_close`

**Reason:** `sk->uaf` is not set to `NULL`
The Birth of AWRP (Altered Write-Read Pairs)

- Manual analysis of 182 incorrect patches in Linux kernel

**Observation 1:** Old and new vulnerabilities share similar contexts

- **Init objA**
- **Crash A**
- **Path A**
- **Incorrect patch introduced**
- **Path B**
- **Init objB**
- **Path B**
- **Crash B**
The Birth of AWRP (Altered Write-Read Pairs)

- Manual analysis of 182 incorrect patches in Linux kernel

Observation 2: New vulnerability results from Altered Write-Read Pairs (AWRP)

Incorrect patches

- Add read/write
  - Obj A
  - write/read from another site
  - Bug triggered

- Remove read/write
  - Obj A
  - write/read from another site
  - Bug triggered

- Path condition be changed
  - Obj A
  - PoC that could bypass the condition
KLAUS: A Framework to Identify and Utilize AWRP

- The AWRP mechanism can provide a method for analyzing patches

**AWRP Identification**
- Intra-procedural/Inter-procedural analysis
- AWRP construction

**AWRP Application**
- AWRP-Driven Fuzzing
AWRP Identification: The Abstract State

**Variables in Kernel:** \( V = \{v_1, \cdots, v_n\} \)

**AWRP Identity:** \( \text{type}(v) \)

- **Local Variables:** \( \text{type}(v) = \text{function\_name} + \text{stack\_offset} \)
- **Global or Static Variables:** \( \text{type}(v) = \text{module\_name} + \text{variable\_name} \)
- **Heap Objects:**
  - **Individual Object:** \( \text{type}(v) = \text{object\_type\_name} \)
  - **Field of an Object:** \( \text{type}(v) = \text{object\_type\_name} + \text{field\_offset} \)

```
struct nfc_llcp_local {
    struct list_head list;
    struct nfc_dev *dev;
    struct kref ref;
    ...
}
```

```
llcp_sock->local->ref = xxx  // write
if(llcp_sock->local->ref == xxx)  // read
```

**Write/Read Pair**

\( \text{type}(v) \): nfc_llcp_local+0x18
AWRP Identification: The Abstract State

Variables in Kernel: $V = \{v_1, \cdots, v_n\}$

AWRP Info: value(v)
  - value(v) = {⟨cond, content⟩} : under the condition con, the value of v is equal to content

```
if (llcp_sock->ssap == LLCP_SAP_MAX) {
    llcp_sock->sock = NULL;
}
```

value(v): {⟨llcp_sock->ssap == LLCP_SAP_MAX, 'NULL'⟩}

Symbolic Strings

The Abstract State: $S = \{\text{cond,} \langle \text{type}(v_1), \text{value}(v_1) \rangle, \cdots, \langle \text{type}(v_n), \text{value}(v_n) \rangle\}$
AWRP Identification: The Transfer Function

The Abstract State: \( S = \{ \text{cond}, \langle \text{type}(v_1), \text{value}(v_1) \rangle, \cdots, \langle \text{type}(v_n), \text{value}(v_n) \rangle \} \)

Transfer(\( S, \text{inst} \)): The impact of executing \( \text{inst} \) in the state \( S \)
- The \( \text{inst} \) writes to a variable \( v \)
  - replace \( \text{value}(v) \) by a new \( \langle \text{cond}, \text{content} \rangle \)
- The \( \text{inst} \) casts variable \( v \) from one type to another type
  - update \( \text{type}(v) \) to a new one
- The \( \text{inst} \) is a conditional jump
  - \( \text{cond} \) in \( S \) is conjuncted with the jump condition
AWRP Identification: Intra/Inter-procedural Analysis

Patch

Modified Functions

Original Version

State S

Patched Version

State S’

Diff State => AWRP
The Application of AWRP: AWRP-driven Fuzzer

- Developed based on Syzkaller.
- Prefer to cover more locations where AWRP is used
- Instrument the basic blocks on the essential route leading to AWRP
Evaluation

- Used 23 ground-truth cases from syzkaller community
- Same initial seed & time (3 days) & rounds (5) & environment
- Compared with Syzkaller

**KLAUS** found 23/23 incorrect patches

**Syzkaller** found 13/23 incorrect patches

**KLAUS** triggers crashes caused by incorrect patches faster than **Syzkaller** in 12/13 cases

**KLAUS** found 30 new incorrect patches in the wild! The community has confirmed and fixed 25 of these patches
Takeaways

❖ The AWRP method provides a framework for patch analysis
❖ KLAUS, utilizing the AWRP, can better detect incorrect patches
❖ We look forward to more research on AWRP

Source Code: [https://github.com/wupco/KLAUS](https://github.com/wupco/KLAUS)

yuhang.wu@northwestern.edu

@wupco1996