UBITect: A Precise and Scalable Method to Detect Use-Before-Initialization Bugs in Linux Kernel

Yizhuo Zhai, Yu Hao, Hang Zhang, Daimeng Wang, Chengyu Song, Zhiyun Qian, Mohsen Lesani, Srikanth V. Krishnamurthy, Paul Yu
Use-Before-Initialization (UBI) Bugs

(1) Vulnerable Code

```c
static int queue_manag(void *data)
{
    /* backlog is declared without initialization */
    struct crypto_async_request *backlog;
    if (cpg->eng_st == ENGINE_IDLE) {
        backlog = crypto_get_backlog(&cpg->queue);
    }
    /* Uninitialized backlog is used*/
    if (backlog) {
        /* uninitialized pointer dereferenced! */
        backlog->complete(backlog, -EINPROGRESS);
    }
    return 0;
}
```

(2) UBI Scenario
Security Risks

- malicious->func();
- uninit->func();
- copy_to_user(dst, src, size);
- for (int i = 0; i < len; i++)
  a[i]

- Arbitrary Code Execution
- Denial of Service
- Information Leakage
- Out of Bound Memory Access
Previous Solutions and Limitations

Mitigation:
- Zeroing the allocated object:
  e.g. Unisan, SafeInit

Detection:
- Intra-procedural static analysis:
  e.g. -Wuninitialized, cppcheck
- Symbolic execution:
  e.g. Clang Static Analyzer
- Dynamic Analysis:
  e.g. MemorySanitizer, kmemcheck
Challenges

Scalability

Precise Analysis

Approach: **UBI bugs deTector**

**Flow-Sensitive Qualifier Inference**
- Bottom-up, summary-based
- Inter-procedural
- flow-/filed-/context-sensitive
- Guidance

**Path-sensitive Symbolic Execution**
- Path-sensitive

**Scalable**

**Precise**
### Approach: UBITect (Continue)

<table>
<thead>
<tr>
<th>Type Qualifier</th>
<th>Qualifier Analysis</th>
<th>Under-Constrained SE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What?</strong></td>
<td>Type annotations</td>
<td></td>
</tr>
<tr>
<td><strong>Why?</strong></td>
<td>Additional information</td>
<td>Ensure the correct use</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>const int var;</code></td>
<td></td>
</tr>
</tbody>
</table>
## Approach: UBITect (Continue)

<table>
<thead>
<tr>
<th>Type Qualifier</th>
<th>Qualifier Analysis</th>
<th>Under-Constrained SE</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>init</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>uninit</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Approach: UBITect (Continue)

<table>
<thead>
<tr>
<th>Type Qualifier</th>
<th>Qualifier Analysis</th>
<th>Under-Constrained SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uninit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

variable get declared

variable get initialized

variable get used, a warning is generated here.
### Approach: UBITect (Continue)

<table>
<thead>
<tr>
<th>Type Qualifier</th>
<th>Qualifier Analysis</th>
<th>Under-Constrained SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uninit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Init**
- Variable declared
- Variable initialized
- Variable used, a warning is generated here.

**Uninit**
- Variable declared
- Variable initialized
- Variable used, a warning is generated here.
Approach: UBITect (Continue)

Bottom-up, summary-based

* FS: Function Summary
Approach: UBITect (Continue)

Bottom-up, summary based

```
int F3(int a, int *pa)
{
    *pa = 4;
    if (a) {
        //do sth here
        return 0;
    } else
        return -1;
}

* FS: Function Summary

<table>
<thead>
<tr>
<th>FS3</th>
<th>requirement</th>
<th>update</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>init</td>
<td>N/A</td>
</tr>
<tr>
<td>pa</td>
<td>init</td>
<td>N/A</td>
</tr>
<tr>
<td>pa_obj</td>
<td>N/A</td>
<td>init</td>
</tr>
<tr>
<td>ret</td>
<td>N/A</td>
<td>init</td>
</tr>
</tbody>
</table>
Putting them together

Implementation:
LLVM 7.0.0
13K+ LoC
SE Engine: KLEE
Evaluations

- Detecting Known UBI bugs
- Detecting New UBI bugs
- Comparison with cppcheck and Clang Static Analyzer
Evaluation I: Detecting Known UBI bugs

Table 2: Evaluation I: UBI bugs patched since 2013. All of the uninitialized variables are located on stack. UBI\textsc{Tect} can successfully detect all of them.

<table>
<thead>
<tr>
<th>Commit or CVE No</th>
<th>Type</th>
<th>UBI\textsc{Tect}</th>
</tr>
</thead>
<tbody>
<tr>
<td>bde6f9d</td>
<td>intra-procedural</td>
<td>Yes</td>
</tr>
<tr>
<td>1a92b2b</td>
<td>intra-procedural</td>
<td>Yes</td>
</tr>
<tr>
<td>8134233</td>
<td>inter-procedural</td>
<td>Yes</td>
</tr>
<tr>
<td>c94a3d4</td>
<td>inter-procedural</td>
<td>Yes</td>
</tr>
<tr>
<td>da5e7ff</td>
<td>inter-procedural</td>
<td>Yes</td>
</tr>
<tr>
<td>CVE 2010-2963</td>
<td>inter-procedural</td>
<td>Yes</td>
</tr>
<tr>
<td>7814657</td>
<td>inter-procedural</td>
<td>Yes</td>
</tr>
<tr>
<td>6fd4b15</td>
<td>inter-procedural</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Evaluation II: Detecting New UBI bugs

- Linux 4.14, al YESконfig
- 16163 files, 616893 functions
- 1 week analysis
- SE timeout to 120s
- SE memory out as 2GB

- 138 human verified bugs
- 118 unpatched
- 52 bugs confirmed
False Positive Reasons

- Incomplete guidance
- Imprecise indirect call resolution
- LLVM optimizations
- Limitations of SE
Evaluation II: Detecting New UBI bugs

cppcheck
- Intra-procedural analysis

Clang Static Analyzer (CSA)
- Symbolic Execution in a single file

164 bugs
2 TPs
78 Files
17 bugs
Case Study

```c
/*drivers/media/usb/pvusb2/pvusb2-hdw.c*/
static unsigned int ctrl_cx2341x_getv4lflags(struct pvr2_ctrl *cptr) {
    struct v4l2_queryctrl qctrl;
    qctrl.id = cptr->info->v4l_id;
    /*drivers/media/common/cx2341x.c*/
    cx2341x_ctrl_query(&cptr->hdw->enc_ctl_state,&qctrl);
    if (qctrl.flags & V4L2_CTRL_FLAG_READ_ONLY) {
        return qctrl.flags;
    }
}
```
Conclusion

- UBI bugs cause critical security issues and zeroing the variable cannot fully mitigate them.
- UBITect: A precise and scalable tool to detect UBI bugs in Linux kernel
- 52 new bugs have been confirmed in Linux
Conclusion

UBI bugs cause critical security issues and zeroing the variable cannot fully mitigate them.

UBITect: A precise and scalable tool to detect UBI bugs in the Linux kernel

Progressive Scrutiny: Incremental Detection of UBI bugs in the Linux Kernel

Yizhuo Zhai, Yu Hao, Zheng Zhang, Weiteng Chen, Guoren Li, Zhiyun Qian, Chengyu Song, Manu Sridharan, Srikanth V. Krishnamurthy, Trent Jaeger, Paul Yu
Background: Rapid Linux Kernel Development Cycle

- v4.14
- v4.15
- v4.16
- v4.17
- v4.18

- v4.15-rcs
- v4.16-rcs
- v4.17-rcs
- v4.18-rcs

- v4.14.x
- v4.15.x
- v4.16.x
- v4.17.x
- v4.18.x

- 2 Months
- 2 Months
- 2 Months
- 2 Months

- New Feature
- Bug Fixes
- Bugs !!!

- 10 Commits/Hour
Background: Security Issues in Linux Kernel

- Critical Bugs: 3.3 Years
- High-Sensitive Bugs: 6.4 Years
Existing Effort

Fuzzing + Sanitizer

Specified bugs
Framework for specified modules.
Existing Effort - Limitations

Dynamic Analysis

Key
- Not instrumented
- Conditionalized but not executed
- Executed
- Not executed

Static Analysis

```
(declaim (optimize sb-cover:store-cover)
(defun test (n)
  (when (zerop n)
    (if (eq n 0)
      (print 'zero)
      (if (eq (floor n 0.0)
        (print 'single-fp-zero)
        (print 'double-fp-zero))))
  (when (minusp n)
    (print 'negative))
  (when (plusp n)
    (tagbody
      (print 'positive)
      (go end)
      (print 'dummy)
      end))
```
Conclusion

UBI bugs cause critical security issues and zeroing the variable cannot fully mitigate them.

UBITect: A precise and scalable tool to detect UBI bugs in Linux kernel.

52 new bugs have been confirmed in Linux 6.

Observation

- Quicker Turnaround time
- Test proposed patches
- Exhaustive coverage
Background: UBITect (FSE’20)

Qualifier Analysis

Variable declared

Variable get initialized

Variable get used, a warning is generated here.

Var =

Var =

*Var

Uninit

Init

Init

Uninit

Under-Constraint SE

Var =

Var =

*Var

Uninit

Ruben

Uninit
Conclusion

UBI bugs cause critical security issues and zeroing the variable cannot fully mitigate them.

UBITect: A precise and scalable tool to detect UBI bugs in Linux kernel

52 new bugs have been confirmed in Linux.

IncreLux Workflow

* SCC: Strongly Connected Component
Evaluations

- V4.14 as the baseline
- Stable versions: v4.15-rcs-v4.19, v5.4, v5.9
  v4.15.1-v4.15.18
- Per patch analysis
- Equivalence Analysis

- Speed Improvement
- Time Breakdown
- Bug Finding Results
- Patch Identification Results.
Conclusion

UBI bugs cause critical security issues and zeroing the variable cannot fully mitigate them. **UBITect** : A precise and scalable tool to detect UBI bugs in Linux kernel. 52 new bugs have been confirmed in Linux.

### Evaluation – Time Speedup

**TABLE II**: Incremental analysis results for mainline versions v4.14 to v4.16. Please refer to the Appendix for the full table from v4.14 to v5.9. **T(h)**: Total analysis time in hours. **SU**: Speedup compared to exhaustive analysis of v4.14. **FM**: Number of functions modified compared to the immediate predecessor. **FR**: Number of functions (re-)analyzed in this version. **Warn**: Number of Warnings reported in the current version. **Disappearing**: Number of warnings that disappear in the current version (compared with the last analyzed version). **Equal**: Number of warnings that remain in the current version compared to the immediate previous analyzed version. **New**: Number of warnings newly introduced in the current version compared with the last analyzed version. **SE-New**: New bugs confirmed by SE that are introduced in the new version.

<table>
<thead>
<tr>
<th>versions</th>
<th>T(h)</th>
<th>SU</th>
<th>FM</th>
<th>FR</th>
<th>Warn</th>
<th>Disappearing</th>
<th>Remaining</th>
<th>New</th>
<th>SE-New</th>
</tr>
</thead>
<tbody>
<tr>
<td>v4.14</td>
<td>106.75</td>
<td>N/A</td>
<td>N/A</td>
<td>629862</td>
<td>103616</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>622</td>
</tr>
<tr>
<td>v4.15-rc1</td>
<td>28.26</td>
<td>3.78</td>
<td>23941</td>
<td>42548</td>
<td>21190</td>
<td>4325</td>
<td>99291</td>
<td>4979</td>
<td>69</td>
</tr>
<tr>
<td>v4.15-rc2</td>
<td>2.91</td>
<td>36.74</td>
<td>719</td>
<td>2617</td>
<td>2190</td>
<td>422</td>
<td>103848</td>
<td>211</td>
<td>0</td>
</tr>
<tr>
<td>v4.15-rc3</td>
<td>2.27</td>
<td>47.04</td>
<td>656</td>
<td>3084</td>
<td>1937</td>
<td>301</td>
<td>103758</td>
<td>238</td>
<td>0</td>
</tr>
<tr>
<td>v4.15-rc4</td>
<td>1.13</td>
<td>94.52</td>
<td>332</td>
<td>1268</td>
<td>718</td>
<td>116</td>
<td>103880</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>v4.15-rc5</td>
<td>1.28</td>
<td>83.31</td>
<td>329</td>
<td>1793</td>
<td>1339</td>
<td>207</td>
<td>103743</td>
<td>331</td>
<td>0</td>
</tr>
<tr>
<td>v4.15-rc6</td>
<td>1.15</td>
<td>92.6</td>
<td>273</td>
<td>1761</td>
<td>1282</td>
<td>96</td>
<td>103978</td>
<td>96</td>
<td>0</td>
</tr>
<tr>
<td>v4.15-rc7</td>
<td>0.43</td>
<td>248.74</td>
<td>101</td>
<td>403</td>
<td>263</td>
<td>21</td>
<td>104053</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>v4.15-rc8</td>
<td>1.33</td>
<td>80.15</td>
<td>243</td>
<td>1707</td>
<td>1305</td>
<td>114</td>
<td>103966</td>
<td>151</td>
<td>0</td>
</tr>
<tr>
<td>v4.15-rc9</td>
<td>0.63</td>
<td>169.82</td>
<td>217</td>
<td>1031</td>
<td>696</td>
<td>49</td>
<td>104068</td>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>v4.15</td>
<td>0.13</td>
<td>800.63</td>
<td>122</td>
<td>262</td>
<td>215</td>
<td>36</td>
<td>104080</td>
<td>48</td>
<td>2</td>
</tr>
<tr>
<td>v4.16-rc1</td>
<td>26.77</td>
<td>3.99</td>
<td>21251</td>
<td>52151</td>
<td>26922</td>
<td>4240</td>
<td>99888</td>
<td>4742</td>
<td>55</td>
</tr>
<tr>
<td>v4.16-rc2</td>
<td>0.24</td>
<td>453.72</td>
<td>220</td>
<td>521</td>
<td>342</td>
<td>10</td>
<td>104620</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>v4.16-rc3</td>
<td>0.7</td>
<td>151.6</td>
<td>434</td>
<td>1012</td>
<td>713</td>
<td>136</td>
<td>104509</td>
<td>77</td>
<td>1</td>
</tr>
<tr>
<td>v4.16-rc4</td>
<td>3.39</td>
<td>31.48</td>
<td>278</td>
<td>7398</td>
<td>3446</td>
<td>502</td>
<td>104084</td>
<td>509</td>
<td>4</td>
</tr>
<tr>
<td>v4.16-rc5</td>
<td>0.76</td>
<td>141.23</td>
<td>422</td>
<td>979</td>
<td>598</td>
<td>80</td>
<td>104513</td>
<td>76</td>
<td>0</td>
</tr>
<tr>
<td>v4.16-rc6</td>
<td>0.26</td>
<td>415.01</td>
<td>144</td>
<td>401</td>
<td>279</td>
<td>15</td>
<td>104574</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>v4.16-rc7</td>
<td>0.93</td>
<td>115.2</td>
<td>498</td>
<td>1543</td>
<td>819</td>
<td>107</td>
<td>104494</td>
<td>190</td>
<td>2</td>
</tr>
<tr>
<td>v4.16</td>
<td>0.33</td>
<td>318.92</td>
<td>183</td>
<td>535</td>
<td>278</td>
<td>18</td>
<td>104666</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>v4.17rc1-v4.19</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>158018</td>
<td>43762</td>
<td>69652</td>
<td>83836</td>
<td>N/A</td>
</tr>
<tr>
<td>v5.4</td>
<td>97.9</td>
<td>1.09</td>
<td>99370</td>
<td>205327</td>
<td>152746</td>
<td>40720</td>
<td>85425</td>
<td>67321</td>
<td>N/A</td>
</tr>
<tr>
<td>v5.9</td>
<td>99.65</td>
<td>1.07</td>
<td>91741</td>
<td>197413</td>
<td>152746</td>
<td>40720</td>
<td>85425</td>
<td>67321</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Evaluations - New Bugs

▶ 44 bug report sampled
▶ 22 TP (FP: 50%)
▶ 17 can be triggered

<table>
<thead>
<tr>
<th>Sub-System</th>
<th>Module</th>
<th>Variable</th>
<th>Line No.</th>
<th>Intro.</th>
<th>PatchImpact</th>
</tr>
</thead>
<tbody>
<tr>
<td>input/ideep</td>
<td>hidem.c</td>
<td>unmask_code</td>
<td>380</td>
<td>v4.15-rc1</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>atomisp-mt9m14.c</td>
<td>revalue</td>
<td>1552</td>
<td>v4.15-rc1</td>
<td>A</td>
</tr>
<tr>
<td>drm/mouveau media/mx274</td>
<td>ioctl.c</td>
<td>type</td>
<td>269</td>
<td>v4.15-rc1</td>
<td>S</td>
</tr>
<tr>
<td>net/mx25</td>
<td>imx274.c</td>
<td>err</td>
<td>659</td>
<td>v4.15-rc1</td>
<td>F</td>
</tr>
<tr>
<td>net/mx5e</td>
<td>en_common.c</td>
<td>min_inline_mode</td>
<td>180</td>
<td>v4.15-rc1</td>
<td>F</td>
</tr>
<tr>
<td>xfs</td>
<td>xfs_bmap.c</td>
<td>params</td>
<td>989</td>
<td>v4.15-rc1</td>
<td>F</td>
</tr>
<tr>
<td>xfs</td>
<td>xfs_bmap.c</td>
<td>got_br_startoff</td>
<td>4868</td>
<td>v4.15-rc1</td>
<td>E</td>
</tr>
<tr>
<td>iio/adc</td>
<td>stm332-dfsm-adc.c</td>
<td>s</td>
<td>1521</td>
<td>v4.15-rc1</td>
<td>E</td>
</tr>
<tr>
<td>iio/adc</td>
<td>stm332-dfsm-adc.c</td>
<td>status</td>
<td>866</td>
<td>v4.16-rc1</td>
<td>C</td>
</tr>
<tr>
<td>iio/adc</td>
<td>qcom-pm08xx-xx-xoadc.c</td>
<td>int_en</td>
<td>873</td>
<td>v4.16-rc1</td>
<td>C</td>
</tr>
<tr>
<td>iio/adc</td>
<td>ch</td>
<td>599</td>
<td>v4.16-rc1</td>
<td>F</td>
<td>MC</td>
</tr>
<tr>
<td>net: msc</td>
<td>ocelot.c</td>
<td>val</td>
<td>365</td>
<td>4.18-rc1</td>
<td>F</td>
</tr>
<tr>
<td>media: davinci_vpe</td>
<td>dm365_isif.bc</td>
<td>format, pixelformat</td>
<td>234</td>
<td>4.18-rc1</td>
<td>F</td>
</tr>
<tr>
<td>display</td>
<td>dc_link.c</td>
<td>old_downspread.raw</td>
<td>34</td>
<td>4.18-rc1</td>
<td>E</td>
</tr>
<tr>
<td>display</td>
<td>dc_link_dp.c</td>
<td>training_rd_interval</td>
<td>34</td>
<td>4.18-rc1</td>
<td>E</td>
</tr>
<tr>
<td>net:mscc</td>
<td>ocelot.c</td>
<td>val</td>
<td>34</td>
<td>4.18-rc1</td>
<td>E</td>
</tr>
<tr>
<td>scci: sd</td>
<td>sd.c</td>
<td>sshdr.asc</td>
<td>2390</td>
<td>v4.19-rc1</td>
<td>E</td>
</tr>
</tbody>
</table>
Bug Lifetime – Case Study

```c
/* drivers/media/i2c/imx274.c */
/* uninteresting code lines are omitted */
static int imx274_regmap_util_write_table_8 ()
{
    int err;
    if (range_count == 1)
        err = regmap_write(regmap,
                           range_start, range_vals[0]);
    else if (range_count > 1)
        err = regmap_bulk_write(regmap, range_start,
                           &range_vals[0],
                           range_count);
    else
        err = 0;
    if (err) {
        return err;
    }
}
```

Fig. 6: The patch that fixed the previous bug; this bug was introduced in v4.15-rc1 and the patch was applied in v4.16-rc1. By continuously tracking the bug, INCRELUX could find both the bug upon introduction, and the time of the bug disappearance. If we use this patch as the input for the incremental analysis, the disappearance of the bug indicates that this commit was related to a bug fix.
Conclusion

• IncreLux : A framework for principled incremental analysis of the Linux kernel.
• Dramatic speed ups compared to today’s expensive whole-kernel analysis.
• Fit into the kernel development cycle.
• Effectively identify bugs and bug fixes.
• https://github.com/seclab-ucr/IncreLux.git
Q & A