Linux Driver Verification Program

Alexey Khoroshilov
khoroshilov@linuxtesting.org
Yet another static analysis tool?

- sparse
- Coccinelle
Static Analysis

Key characteristics

• Scope of analysis (kind of bugs)
• False positives (false bugs reported)
• False negatives (real bugs missed)
• Resources required for analysis
Static Analysis: Trade-Off Triangle

- False positives
- Time of analysis
- False negatives
Static Analysis: Trade-Off Triangle

- False positives
- False negative
- Light-weight
- Heavy-weight
- Time of analysis
Coccinelle

- Intra-procedural analysis
- Limited data-flow analysis
The simplest rule

mutex

- should not be locked twice
- should not be unlocked if it is not locked
static ssize_t ep_read (struct file *fd, char __user *buf, size_t len, loff_t *ptr)
{
    struct ep_data *data = fd->private_data;
    void *kbuf;
    ssize_t value;

    if ((value = get_ready_ep (fd->f_flags, data)) < 0)
        return value;

    /* halt any endpoint by doing a "wrong direction" i/o call */
    if (usb_endpoint_dir_in(&data->desc)) {
        if (usb_endpoint_xfer_isoc(&data->desc))
            return -EINVAL;

        DBG (data->dev, "%s halt\n", data->name);
        spin_lock_irq (&data->dev->lock);
        if (likely (data->ep != NULL))
            usb_ep_set_halt (data->ep);
        spin_unlock_irq (&data->dev->lock);
        mutex_unlock(&data->lock);
        return -EBADMSG;
    }

    /* FIXME readahead for O_NONBLOCK and poll(); careful with ZLPs */

    value = -ENOMEM;
    kbuf = kmalloc (len, GFP_KERNEL);
    if (unlikely (!kbuf))
        goto freel;
static int
get_ready_ep (unsigned f_flags, struct ep_data *epdata)
{
    int     val;

    if (f_flags & O_NONBLOCK) {
        if (!mutex_trylock(&epdata->lock))
            goto nonblock;
        if (epdata->state != STATE_EP_ENABLED) {
            mutex_unlock(&epdata->lock);
        } else
            val = -EAGAIN;
    } else
        val = 0;
    return val;
}

val = mutex_lock_interruptible(&epdata->lock);
if (val < 0)
    return val;

switch (epdata->state) {
    case STATE_EP_ENABLED:
        break;
    case STATE_EP_UNBOUND: /* clean disconnect */
        val = -ENODEV;
        val = mutex_unlock(&epdata->lock);
}
return val;
drivers/usb/gadget/inode.c
drivers/usb/gadget/inode.c

/pub/scm / linux/kernel/git/torvalds/linux-2.6.git

summary | shortlog | log | commit | commitdiff | tree
(parent: d06847f) | patch

USB: usb-gadget: unlock data->lock mutex on error path in ep_read()  

author       Alexey Khoroshilov <khoroshilov@ispras.ru>
              Wed, 16 Mar 2011 19:54:05 +0000 (21:54 +0200)

committer    Greg Kroah-Hartman <gregkh@suse.de>

commit        00cc7a5faf25b3ba5cf30fcff62249bd0d122006
tree           604e54a588f74f1904a5cd7810fb922815fed37e       tree | snapshot
parent         d06847f4ec256f4f902075ce5986e10f7c55fa250   commit | diff

USB: usb-gadget: unlock data->lock mutex on error path in ep_read()

ep_read() acquires data->lock mutex in get_ready_ep() and releases it on
all paths except for one: when usb_endpoint_xfer_isoc() failed. The
patch adds mutex_unlock(&data->lock) at that path.

Found by Linux Driver Verification project (linuxtesting.org).

Signed-off-by: Alexey Khoroshilov <khoroshilov@ispras.ru>
Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>
Coccinelle

- Intra-procedural analysis
- Limited data-flow analysis
drivers/scsi/mpt2sas/mpt2sas_ctl.c

618 /**
619 * _ctl_do_mpt_command - main handler for MPT2COMMAND opcode
620 * @state - NON_BLOCKING or BLOCKING
621 */
622 static long
623 _ctl_do_mpt_command(...) {
624 ...
625 if (state == NON_BLOCKING && !mutex_trylock(&ioc->ctl_cmds.mutex))
626     return -EAGAIN;
627 else if (mutex_lock_interruptible(&ioc->ctl_cmds.mutex))
628     return -ERESTARTSYS;
629}
From Alexey Khoroshilov <>
Subject [PATCH] [SCSI] mpt2sas: fix double mutex lock in NON_BLOCKING state
Date Mon, 18 Apr 2011 22:53:38 +0400

If mutex_trylock succeed, the control flow goes to mutex_lock_interruptible()
that is not a good thing.

Found by Linux Driver Verification project (linuxtesting.org).

Signed-off-by: Alexey Khoroshilov <khoroshi.ev@ispras.ru>

---
drivers/scsi/mpt2sas/mpt2sas_ctl.c | 24 ++++++++++++++++++++++++++++++++++
1 files changed, 16 insertions(+), 8 deletions(-)
diff --git a/drivers/scsi/mpt2sas/mpt2sas_ctl.c b/drivers/scsi/mpt2sas/mpt2sas_ctl.c
index 1c6d2b4..9bd7ffe 100644
--- a/drivers/scsi/mpt2sas/mpt2sas_ctl.c
+++ b/drivers/scsi/mpt2sas/mpt2sas_ctl.c
@@ -648,8 +648,10 @@ _ctl_do_mpt_command(struct MPT2SAS_ADAPTER *ioc,

    issue_reset = 0;

    - if (state == NON_BLOCKING && !mutex_trylock(&ioc->ctl_cmds.mutex))
    -     return -EAGAIN;
+   if (state == NON_BLOCKING) {
+     if (!mutex_trylock(&ioc->ctl_cmds.mutex))
+         return -EAGAIN;
+   }
   else if (mutex_lock_interruptible(&ioc->ctl_cmds.mutex))
     return -ERESTARTSYS;

@@ -1587,8 +1589,10 @@ _ctl_diag_register(void user *arg, enum block state state)
Heavy-Weight Analysis

Based on picture from http://engineer.org.in
How it works?

- CEGAR - Counter-Example Guided Abstraction Refinement
CEGAR

1. Abstraction

2. Checking of boolean program

3. Error trace analysis

4. Model refinement

program in C

new predicates

There is a path to error state

The path is unfeasible

The path is feasible

trace

boolean program

SAFE

UNSAFE
CEGAR-based Heavy-Weight Tools

Commercial:
• Microsoft SDV

Academic:
• BLAST
• CPAChecker (U. Passau)
• SATABS (U. Oxford)
• ARMC (U. Munich)
Microsoft Static Driver Verifier

We've created a number of things to do rich static analysis. We actually went out and bought for a little over $30 million a company that was in the business of building those kinds of tools, and we said now we want you to focus on applying these tools to large-scale software systems, the kind of system we have in the source code of Windows or Office, and see how far we can get on this.

...We call the system that does this kind of proof, it's a model-checking system. You describe the constraints, including things as simple as nobody should acquire the lock if they've already acquired it, nobody should release it if they haven't acquired it, certain things about the multi-threading aspect of the code that you want to make sure work very well. And you describe those things literally, in this case in the C code itself, and then the analyzer goes through and reduces the program, takes away anything that doesn't affect the path analysis that it's trying to go through to determine is there some path through the program that violates the constraints.

The initial domain we applied this in was in device drivers.

Bill Gates at
17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages and Application, 2002
Microsoft Static Driver Verifier

- Included into Microsoft Windows Driver Developer Kit (DDK) in 2006
- Continuous improvements:
  - Kinds of interfaces:
  - Number of rules:
  - Time required to analyze one driver:
    ??? → 2-3 hours (2010)
Microsoft Static Driver Verifier

Results

- 33 critical bugs in the WDK sample drivers
- 53 critical bugs in kernel-mode drivers
CEGAR-based Heavy-Weight Tools

**Commercial:**
- Microsoft SDV

**Academic:**
- BLAST
- CPAChecker (U. Passau)
- SATABS (U. Oxford)
- ARMC (U. Munich)
14-Oct-2011: BLAST 2.7 released

Linux Verification Center announces the release of BLAST 2.7 - a new version of an open source model checker for C programs. The tool automatically checks if a C program satisfies behavioral properties of the interfaces it uses. BLAST is based on counterexample-driven automatic abstraction refinement to construct an abstract model which is model checked for safety properties.

The first version of BLAST was developed at UC Berkeley by Ranjit Jhala, Rupak Majumdar, and Grégoire Sutre and was supported by the US National Science Foundation. The BLAST 2.0 Team includes Thomas A. Henzinger, Dirk Beyer, Rupak Majumdar, and Ranjit Jhala. The latest release of the team is BLAST 2.5 of 2008.

BLAST 2.7 is a result of improvements made in BLAST 2.6 by Linux Verification Center team within Linux Driver Verification program and for the purpose to take part in Competition on Software Verification at TACAS'12.

The main improvements are as follows.

16-Sep-2011: BLAST 2.6 released

Linux Verification Center announces the release of BLAST 2.6 - a new version of an open source model checker for C programs. The tool automatically checks if a C program satisfies behavioral properties of the interfaces it uses. BLAST is based on counterexample-driven automatic abstraction refinement to construct an abstract model which is model checked for safety properties.
Yet another static analysis tool?
Linux Driver Verification Program

• Yes, our idea is to promote heavy-weight verification tools
• But our idea is **NOT** to push a particular verification technique
LDV Goals

- Provide infrastructure for application of verification tools to Linux device drivers
- Research new verification approaches in the industrial settings
- Improve quality of the Linux device drivers
- Provide a basis for education of young researches
Where we are

- Static analysis infrastructure
int main(int argc, char* argv[]) {
    ...
    other_func(var);  
    void other_func(int v) {
        ...
        assert(x != NULL);
    }
}
Device Driver World

```c
static struct pci_driver DAC960_pci_driver = {
   .name = "DAC960",
   .id_table = DAC960_id_table,
   .probe = DAC960_Probe,
   .remove = DAC960_Remove,
};

static int DAC960_init_module(void)
{
    int ret;
    ret = pci_register_driver(&DAC960_pci_driver);
    #ifdef DAC960_GAM_MINOR
    if (!ret)
        DAC960_gam_init();
    #endif
    return ret;
}

module_init(DAC960_init_module);
module_exit(DAC960_cleanup_module);
```

Callback interface procedures registration

No explicit calls to linking-level init procedures
Rule Instrumentor

mutex x;
int f(int y)
{
    lock(x);
    ...
    unlock(x);
    return y;
}

int x_locked = 0;
int f(int y)
{
    assert(x_locked == 0);
    x_locked = 1;
    ...
    assert(x_locked == 1);
    x_locked = 0;
    return y;
}
Where we are

- Static analysis infrastructure
- Cluster framework
- Front-ends
  - ldv-manager
  - ldv-git
  - ldv-online
ldv-online

Online Linux Driver Verification Service (alpha)

Start Verification  Verification History  Rules

Start Verification
on x86_64 architecture

1. Ensure that drivers satisfy the following requirements:

- The driver is archived using gzip or bzip2 and has one of the following extensions: .tar.bz2, tar.gz, .tgz
- Archive should contain:
  - Makefile (written to be compiled with the kernel)
    - obj-m is mandatory
  - Sources needed by Makefile
- Archive should not contain generated files left from builds

2. Upload driver.

3. Wait for results.

Browse...
Verification Report

Driver: test-0032-wl12xx-unsafe.tar.bz2
Timestamp: 2011-01-19 20:51:12
Verification architecture: x86_64

You can see verification verdict for each rule and linux kernel. Verdict may be:

- **Safe** - there is no mistakes for the given linux kernel and rule.
- **Unsafe** - driver may contain an error. You can see the error trace by clicking on the "Unsafe" link for the corresponding linux kernel and rule.
- **Build failed** - your driver is not compatible with the given linux kernel. In this case you may see the compile error trace by clicking on the "more details" link.
- **Unknown** - tools can not determine whether your driver Safe or Unsafe.
- **Queued** - the driver waits for the turn to verification.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Verdict</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutex lock/unlock</td>
<td>Unsafe</td>
</tr>
<tr>
<td>NOIO allocation under usb_lock</td>
<td>Safe</td>
</tr>
<tr>
<td>Module get/out</td>
<td></td>
</tr>
<tr>
<td>PCI pool create/destroy, alloc/tree</td>
<td></td>
</tr>
<tr>
<td>Delay in probe, irq, on/off</td>
<td>Queued</td>
</tr>
<tr>
<td>Memory allocation inside spinlocks</td>
<td>Queued</td>
</tr>
<tr>
<td>Linked list double add</td>
<td>Queued</td>
</tr>
<tr>
<td>USB alloc/free urb</td>
<td>Queued</td>
</tr>
<tr>
<td>Spinlocks lock/unlock</td>
<td>Queued</td>
</tr>
</tbody>
</table>
Where we are

- Static analysis infrastructure
- Cluster framework
- Front-ends
  - ldv-manager
  - ldv-git
  - ldv-online
- Results database
  - Error trace visualizer
  - Knowledge base
  - Comparison framework
Error Trace Visualizer

Rule: Mutex lock/unlock

Error trace

Source code

```
3182  LDV_IN_INTERRUPT = 1;
3191  +ldv_initialize_FOREACH();
3195  tmp_8 = nondet_int(); /* The function body
3198  assert(tmp_8 != 0);
3200  tmp_7 = nondet_int(); /* The function body
3202  assert(tmp_7 != 0);
3204  assert(tmp_7 != 1);
3206  assert(tmp_7 != 2);
3208  assert(tmp_7 != 3);
3210  assert(tmp_7 != 4);
3212  assert(tmp_7 != 5);
3214  assert(tmp_7 != 6);
3216  assert(tmp_7 != 7);
3218  assert(tmp_7 != 8);
3220  assert(tmp_7 != 9);
3222  assert(tmp_7 != 10);
3224  assert(tmp_7 != 11);
3226  carl9170_op_set_key(var_group1 /* hw */
3228  {
3230  -ar = *(hw).priv;
3232  err = 0;
3234  assert(*(ar).disable_offload == 0);
3236  assert(vif != 0);
3238  +tmp_7 = is_main_vif(ar /* ar */, vif;
3240  assert(tmp_7 == 0);
3242  assert(*(ar).rx_software_decryption
3244  +mutex_unlock_mutex(&ar->mutex /*
1031  static int carl9170_op_set_key(struct ieee80211_hw *hw, enum set_key_c
1027  struct ieee80211_vif *vif,
1028  struct ieee80211_sta *sta,
1029  struct ieee80211_key_conf *key)
1030  {
1031  struct ar9170 *ar = hw->priv;
1032  int err = 0, 1;
1033  u8 ktype;
1034  if (ar->disable_offload || !vif)
1036  return -EOPNOTSUPP;
1037  /*
1038  * We have to fall back to software encryption, whenever
1039  * the user choose to participates in an IBSS or is connected
1040  * to more than one network.
1041  */
1042  /*
1043  * This is very unfortunate, because some machines cannot handle
1044  * the high throughpt speed in 802.11n networks.
1045  */
1046  if (!is_main_wif(ar, vif))
1048  goto err_softw;
1049  /*
1050  * While the hardware supports *catch-all* key, for offloading
1051  * group-key en-/de-cryption. The way how the hardware
1052  * decides which keyId maps to which key, remains a mystery...
```
# Knowledge Base

<table>
<thead>
<tr>
<th>#</th>
<th>Task</th>
<th>Kernel</th>
<th>Rule</th>
<th>Total</th>
<th>Safe</th>
<th>Unsafe</th>
<th>Unknown</th>
<th>Verdicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Task description May, 2011</td>
<td>linux-2.6.38.2</td>
<td>Fail before RI</td>
<td>75</td>
<td>-</td>
<td>-</td>
<td>75</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>32_1</td>
<td>2747</td>
<td>2077</td>
<td>66</td>
<td>604</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>32_7</td>
<td>2747</td>
<td>2227</td>
<td>20</td>
<td>500</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>39_7</td>
<td>2747</td>
<td>2244</td>
<td>15</td>
<td>488</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>68_1</td>
<td>2747</td>
<td>2129</td>
<td>68</td>
<td>550</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>linux-2.6.39</td>
<td>Fail before RI</td>
<td>81</td>
<td>-</td>
<td>-</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>32_7</td>
<td>2826</td>
<td>2278</td>
<td>21</td>
<td>527</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>Task description June, 2011</td>
<td>linux-2.6.39</td>
<td>Fail before RI</td>
<td>81</td>
<td>-</td>
<td>-</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>08_1</td>
<td>2826</td>
<td>2124</td>
<td>50</td>
<td>652</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>32_7</td>
<td>2826</td>
<td>2292</td>
<td>29</td>
<td>505</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>39_7</td>
<td>2826</td>
<td>2319</td>
<td>19</td>
<td>488</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td>43_1a</td>
<td>2826</td>
<td>1861</td>
<td>7</td>
<td>958</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td>68_1</td>
<td>2826</td>
<td>2186</td>
<td>76</td>
<td>564</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>Task description August, 2011</td>
<td>linux-3.0.1</td>
<td>Fail before RI</td>
<td>102</td>
<td>-</td>
<td>-</td>
<td>102</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td>08_1</td>
<td>3203</td>
<td>2550</td>
<td>66</td>
<td>587</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td>32_7</td>
<td>3203</td>
<td>2631</td>
<td>43</td>
<td>529</td>
<td>11</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td>39_7</td>
<td>3203</td>
<td>2659</td>
<td>24</td>
<td>520</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td>43_1a</td>
<td>3203</td>
<td>2623</td>
<td>8</td>
<td>572</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td>68_1</td>
<td>3203</td>
<td>2524</td>
<td>90</td>
<td>589</td>
<td>3</td>
</tr>
</tbody>
</table>
### Bugs Found

http://linuxtesting.org/results/ldv

- 42 patches already applied

#### Problems in Linux Kernel

This section contains information about problems in Linux kernel found within Linux Driver Verification program.

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Brief</th>
<th>Added on</th>
<th>Accepted</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>K0009</td>
<td>Leak</td>
<td>(ath5k) sc-&gt;ah is allocated in ath5k_init_softc() but is not freed</td>
<td>2011-08-08</td>
<td>[Kernel Bug Tracker, bug #37592](<a href="https://lkml.org/lkml/2011/8/23/380">https://lkml.org/lkml/2011/8/23/380</a> commit)</td>
<td>Fixed in the kernel 3.1-rc1</td>
</tr>
</tbody>
</table>
Where we are

but there is no magic

- Verification tools
  - issues with pointer analysis, container_of, functional_pointers, complex data structures
- Environment generator
  - issues with inaccurate environment model in some cases
- RuleDB
  - only 5 rules formalized and debugged
Where we are going

- Improve verification tools
- Formalize new rules
- Continuous application of the tools to Linux device drivers
- Integrate new verification tools
What we are looking for

- Prioritization of rules
- Identification of new rules
- Industrial partners
- Computational power
Conclusions

- Heavy-weight verification is useful in practice
- LDV infrastructure is ready for research and industrial usage
- Number of supported rules must be increased
- Help on rules prioritization and identification are welcome
Thank you!

Alexey Khoroshilov
khoroshilov@linuxtesting.org
http://linuxtesting.org/project/ldv