Finding variability bugs in Linux

Iago Abal Rivas
IT Universitetet i København

Joint work with Andrzej Wąsowski and Claus Brabrand

FOSD Meeting 2014

Agenda

40 variability bugs in Linux: A Qualitative Study (10m)

- Method
- Example
- Observations
- Conclusion

Next step: Towards a feature-sensitive code scanner (5m)

Contribution

- *Identification of 40 variability bugs in the Linux kernel*.
- *A database containing the results of our analysis*.
  (The current version is available at `http://VBDb.itu.dk`.)
- *Self-contained simplified C99 versions of all bugs.*
- *An aggregated reflection over the collection of bugs.*

A technical report is available online at
`http://bit.ly/ITU-TR-2014-180`

Research questions

- RQ1: Are variability bugs limited to any particular type of bugs, "error-prone" features, or specific location?
- RQ2: In what ways does variability affect software bugs?

Filter commits that *look like* variability-related

```
commit 6252547b8a7acced581b649af4ebf6d65f63a34b
Author: Russell King <rmk+kernel@arm.linux.org.uk>
Date:   Tue Feb 7 09:47:21 2012 +0000

    ARM: omap: fix broken twl-core dependencies and ifdefs

    In commit aeb5032b3f, a dependency on IRQ_DOMAIN was added, which causes
    regressions on previously working setups: a previously working non-DT
    kernel configuration now loses its PMIC support.  The lack of PMIC
    support in turn causes the loss of other functionality the kernel had.

    This dependency was added because the driver now registers its
    interrupts with the IRQ domain code, presumably to prevent a build error.

    The result is that OMAP3 oopses in the vp.c code (fixed by a previous
    commit) due to the lack of PMIC support.

    However, even with IRQ_DOMAIN enabled, the driver oopses:

    Unable to handle kernel NULL pointer dereference at virtual address 0000000
```

Filter those that *look like* variability-related

```
diff --git a/drivers/mfd/Kconfig b/drivers/mfd/Kconfig
index cd13e9f..f147395 100644
--- a/drivers/mfd/Kconfig
+++ b/drivers/mfd/Kconfig
@@ -200,7 +200,7 @@ config MENELAUS

 config TWL4030_CORE
        bool "Texas Instruments TWL4030/TWL5030/TWL6030/TPS659x0 Support"
-       depends on I2C=y && GENERIC_HARDIRQS && IRQ_DOMAIN
+       depends on I2C=y && GENERIC_HARDIRQS
        help
          Say yes here if you have TWL4030 / TWL6030 family chip on your board.
          This core driver provides register access and IRQ handling
diff --git a/drivers/mfd/twl-core.c b/drivers/mfd/twl-core.c
index e04e04d..8ce3959 100644
--- a/drivers/mfd/twl-core.c
+++ b/drivers/mfd/twl-core.c
@@ -263,7 +263,9 @@ struct twl_client {

 static struct twl_client twl_modules[TWL_NUM_SLAVES];

+#ifdef CONFIG_IRQ_DOMAIN
 static struct irq_domain domain;
+#endif
```

Filter commits that *look like* fixing a bug

```
commit 6252547b8a7acced581b649af4ebf6d65f63a34b
Author: Russell King <rmk+kernel@arm.linux.org.uk>
Date:   Tue Feb 7 09:47:21 2012 +0000

    ARM: omap: fix broken twl-core dependencies and ifdefs

    In commit aeb5032b3f, a dependency on IRQ_DOMAIN was added, which causes
    regressions on previously working setups: a previously working non-DT
    kernel configuration now loses its PMIC support.  The lack of PMIC
    support in turn causes the loss of other functionality the kernel had.

    This dependency was added because the driver now registers its
    interrupts with the IRQ domain code, presumably to prevent a build error.

    The result is that OMAP3 oopses in the vp.c code (fixed by a previous
    commit) due to the lack of PMIC support.

    However, even with IRQ_DOMAIN enabled, the driver oopses:

    Unable to handle kernel NULL pointer dereference at virtual address 000000
    pgd = c0004000
    [00000000] *pgd=00000000
    Internal error: Oops: 5 [#1] SMP
```

ARM: omap: fix broken twl-core dependencies and ifdefs

```
static int twl_probe()
{
  int *ops = NULL;

  ops = &irq_domain_ops;

  irq_domain_add(ops);
}
```

```
void irq_domain_add(int *ops)
{
  int irq = *ops;
}
```

ARM: omap: fix broken twl-core dependencies and ifdefs

```
static int twl_probe()
{
  int *ops = NULL;
#ifdef CONFIG_OF_IRQ
  ops = &irq_domain_ops;
#endif
  irq_domain_add(ops);
}
```

```
#ifdef IRQ_DOMAIN
void irq_domain_add(int *ops)
{
  int irq = *ops;
}
#endif
```

ARM: omap: fix broken twl-core dependencies and ifdefs

```
static int twl_probe()
{
  int *ops = NULL;
#ifdef CONFIG_OF_IRQ
  ops = &irq_domain_ops;
#endif
  irq_domain_add(ops);
}
```

```
#ifdef IRQ_DOMAIN
void irq_domain_add(int *ops)
{
  int irq = *ops;
}
#endif
```

type: Null pointer dereference

descr: Null pointer on *!OF_IRQ* gets dereferenced if *IRQ_DOMAIN*.

In *TWL4030* driver, attempt to register an IRQ domain with a NULL ops structure: ops is de-referenced when registering an IRQ domain, but this field is only set when *OF_IRQ*.

config: *TWL4030_CORE && !OF_IRQ*

bugfix:

repo: git://git.kernel.org/pub/.../linux-stable.git

hash: 6252547b8a7acced581b649af4ebf6d65f63a34b

fix: model, mapping

trace:

```
!!trace |
.   dyn-call drivers/mfd/twl-core.c:1190:twl_probe()
.   1235:  irq_domain_add(&domain);
..   call kernel/irq/irqdomain.c:20:irq_domain_add()
...   call include/linux/irqdomain.h:74:irq_domain_to_irq()
...   ERROR 77:  if (d->ops->to_irq)
```

links:

```
!!md |
*   [I2C](http://cateee.net/lkddb/web-lkddb/I2C.html)
*   [TWL4030](http://www.ti.com/general/docs/...)
*   [IRQ domain](http://lxr.gwbnsh.net.cn/.../IRQ-domain.txt)
```

Observation (1)

Variability bugs are not limited to any particular type of bugs.

15	**memory errors**	CWE ID
4	null pointer dereference	476
3	buffer overflow	120
3	read out of bounds	125
2	insufficient memory	-
1	memory leak	401
1	use after free	416
1	write on read only	-
8	**compiler warnings**	CWE ID
5	uninitialized variable	457
1	unused function (dead code)	561
1	unused variable	563
1	void pointer dereference	-
7	**type errors**	CWE ID
5	undefined symbol	-
1	undeclared identifier	-
1	wrong number of args to function	-
7	**assertion violations**	CWE ID
5	fatal assertion violation	617
2	non-fatal assertion violation	617
2	**API violations**	CWE ID
1	Linux API contract violation	-
1	double lock	764
1	**arithmetic errors**	CWE ID
1	numeric truncation	197

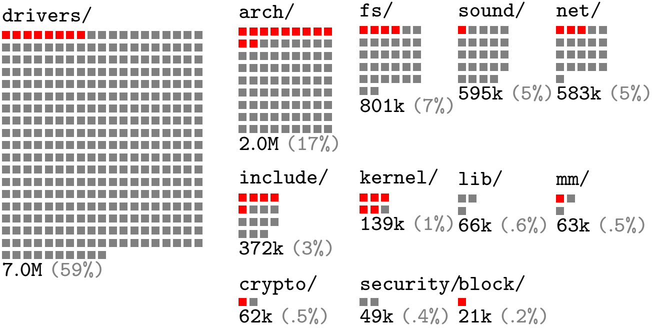
Observation (2)

Variability bugs appear to not be restricted to specific "error prone" features.

64BIT
ACPI_VIDEO
ACPI_WMI
ANDROID
ARCH_OMAP2420
ARCH_OPAM3
ARM_LPAE
BACKLIGHT_CLASS_DEVICE
BCM47XX
BDI_SWITCH
BF60x
BLK_CGROUP
CRYPTO_BLKCIPHER
CRYPTO_TEST
DEVPTS_MULTIPLE_INSTANCES
DISCONTIGMEM
DRM_I915
EP93XX_ETH
EXTCON
FORCE_MAX_ZONEORDER=11
HIGHMEM
HOTPLUG
I2C
IP_SCTP
JFFS2_FS_WBUF_VERIFY
KGDB
KPROBES
KTIME_SCALAR
LOCKDEP
MACH_OMAP_H4
MODULE_UNLOAD
NETPOLL
NUMA
OF_IRQ
PARISC
PCI
PM
PPC64
PPC_256K_PAGES
PREEMPT
PROC_PAGE_MONITOR
PROVE_LOCKING
QUOTA_DEBUG
RCU_CPU_STALL_INFO
RCU_FAST_NO_HZ
S390
SECURITY
SHMEM
SLAB
SLOB
SMP
SND_FSI_AK4642
SND_FSI_DA7210
SSB_DRIVER_EXTIF
STUB_POULSBO
SYSFS
TCP_MD5SIG
TMPFS
TRACE_IRQFLAGS
TRACING
TREE_RCU
TWL4030_CORE
UNIX98_PTYS
VLAN_8021Q
VORTEX
X86
X86_32
XMON
ZONE_DMA

Observation (3)

Variability bugs are not confined to any specific location (file or kernel subsystem)



Observation (4)

We have identified 29 bugs that involve non-locally defined features; i.e., features that are “remotely” defined in another subsystem than where the bug occurred.

E.g.

- 6252547b8a7 occurs in `drivers/` but one of the interacting features, *IRQ_DOMAIN*, is defined in `kernel/`
- 0dc77b6dabe, which occurs also in `drivers/`, is caused by an improper use of the *sysfs* virtual filesystem API—feature *SYSFS* in `fs/`.

Observation (4)

We have identified 29 bugs that involve non-locally defined features; i.e., features that are "remotely" defined in another subsystem than where the bug occurred.

E.g.

- `6252547b8a7` occurs in `drivers/` but one of the interacting features, *IRQ_DOMAIN*, is defined in `kernel/`
- `0dc77b6dabe`, which occurs also in `drivers/`, is caused by an improper use of the *sysfs* virtual filesystem API—feature *SYSFS* in `fs/`.

Observation (5)

Variability can be implicit and even hidden in (alternative) configuration-dependent macro, function, or type definitions specified in (potentially different) header files.

E.g.

- In `0988c4c7fb5`, function `vlan_hwaccel_do_receive` just `BUG()`s when *VLAN_8021Q* is not present.
- In `0f8f8094d28`, `kmalloc_caches` length is configuration-dependent, resulting in a read out of bounds in PowerPC architectures.

Observation (5)

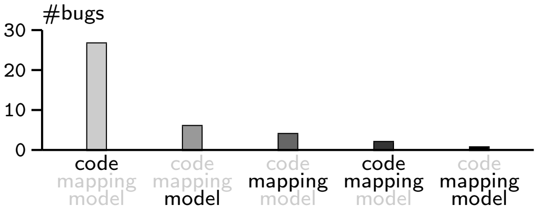
Variability can be implicit and even hidden in (alternative) configuration-dependent macro, function, or type definitions specified in (potentially different) header files.

E.g.

- In `0988c4c7fb5`, function `vlan_hwaccel_do_receive` just `BUG()`s when *`VLAN_8021Q`* is not present.
- In `0f8f8094d28`, `kmalloc_caches` length is configuration-dependent, resulting in a read out of bounds in PowerPC architectures.

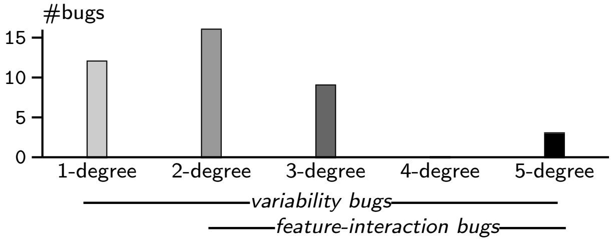
Observation (6)

Variability bugs are fixed not only in the code; some are fixed in the mapping, some are fixed in the model, and some are fixed in a combination of these.



Observation (7)

We have identified as many as 28 feature-interaction bugs in the Linux kernel.

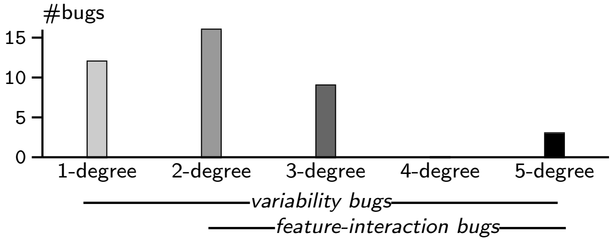


E.g.

- `51fd36f3fad` fixes a bug in the Linux high-resolution timers mechanism due to a numeric truncation error, that only happens in 32-bit architectures not supporting the *KTIME_SCALAR* feature.

Observation (7)

We have identified as many as 28 feature-interaction bugs in the Linux kernel.

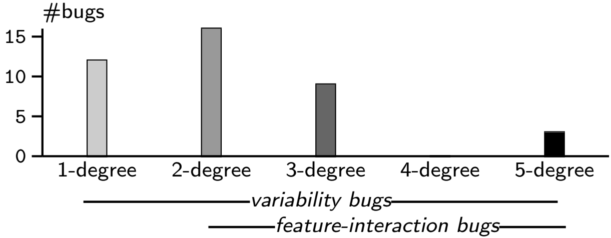


E.g.

- `51fd36f3fad` fixes a bug in the Linux high-resolution timers mechanism due to a numeric truncation error, that only happens in 32-bit architectures not supporting the *KTIME_SCALAR* feature.

Observation (8)

We have identified 12 bugs involving three or more features.

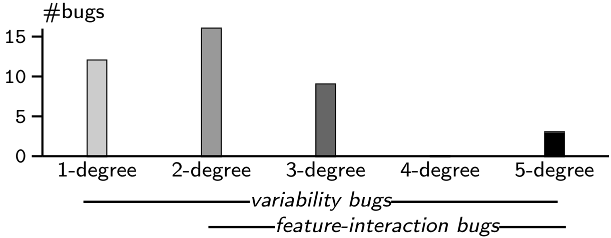


E.g.

- 221ac329e93 is a 5-degree bug due to 32-bit PowerPC architectures not disabling kernel memory write-protection when *KPROBES* is enabled.

Observation (8)

We have identified 12 bugs involving three or more features.



E.g.

- 221ac329e93 is a 5-degree bug due to 32-bit PowerPC architectures not disabling kernel memory write-protection when *KPROBES* is enabled.

Observation (9)

Presence conditions for variability bugs also involve disabled features.

19	***some-enabled***	
6	a	
7	$a \wedge b$	
5	$a \wedge b \wedge c$	
0	$a \wedge b \wedge c \wedge d$	
1	$a \wedge b \wedge c \wedge d \wedge e$	
19	***some-enabled-one-disabled***	
4	$\neg a$	
11	$a \wedge \neg b$	*one of which is:* $(a \vee a') \wedge \neg b$
3	$a \wedge b \wedge \neg c$	
0	$a \wedge b \wedge c \wedge \neg d$	
1	$a \wedge b \wedge c \wedge d \wedge \neg e$	
2	**other configurations**	
1	$\neg a \wedge \neg b$	
1	$a \wedge \neg b \wedge \neg c \wedge \neg d \wedge \neg e$	

Observation (9)

Presence conditions for variability bugs also involve disabled features.

19	***some-enabled***
19	***some-enabled-one-disabled***
2	**other configurations**

- E.g. In `60e233a5660` the implementation of a function `add_uevent_var`, when feature *HOTPLUG* is disabled, fails to preserve an invariant causing a buffer overflow.
- If negated features occur in practice as often as in our sample, then testing maximal configurations only, will miss a significant amount of bugs.

Observation (10)

Effective testing strategies exist for the observed bug presence conditions.

test formula(s)	cost	benefit
$\bigwedge_{f \in F} f$	$O(1)$	48% (19/40)
$\forall g \in F$: $(\bigwedge_{f \in F \setminus \{g\}} f) \wedge \neg g$	$O(\lvert F \rvert)$	95% (38/40)
ψ	$O(2^{\lvert F \rvert})$	100% (40/40)

Should we consider *one-disabled configuration testing* ?

Observation (10)

Effective testing strategies exist for the observed bug presence conditions.

test formula(s)	cost	benefit
$\bigwedge_{f \in F} f$	$O(1)$	48% (19/40)
$\forall g \in F$: $(\bigwedge_{f \in F \setminus \{g\}} f) \wedge \neg g$	$O(\|F\|)$	95% (38/40)
ψ	$O(2^{\|F\|})$	100% (40/40)

Should we consider *one-disabled configuration testing* ?

Conclusion

- *Variability bugs are diverse.*

  (i.e. not confined to particular types of errors, features, locations, . . .)
- *Variability significantly increases the complexity of software bugs.*

Agenda

40 variability bugs in Linux: A Qualitative Study (10m)

- Method
- Example
- Observations
- Conclusion

Next step: Towards a feature-sensitive code scanner (5m)

Goals

- **Real-World Verification**® of C with cpp.
- Primary goal is to find bugs, not verifying their absence.
- Primary subject of study is Linux.
- Simple problems, yet obscured by variability.
- Any technique that scales and works: type-checking, data-flow analysis, model checking, . . . ?

Goals

- **Real-World Verification**® of C with cpp.
- Primary goal is to find bugs, not verifying their absence.
- Primary subject of study is Linux.
- Simple problems, yet obscured by variability.
- Any technique that scales and works: type-checking, data-flow analysis, model checking, . . . ?

and the name of the tool will be . . .

Goals

- **Real-World Verification**® of C with cpp.
- Primary goal is to find bugs, not verifying their absence.
- Primary subject of study is Linux.
- Simple problems, yet obscured by variability.
- Any technique that scales and works: type-checking, data-flow analysis, model checking, . . . ?

and the name of the tool will be . . . `#check`

:-)

Practical considerations

- **Handle *all* C?** Instead take partially preprocessed files.
- **Assembly code?** Support common functions built-in, and ignore the rest (yes, this is unsound).
- **False positives?** No, thanks.
- **Pointer analysis?** Of course, starting with Steensgaard unification-based algorithm (plus tweaks).
- **Data-flow analysis?** Use with care (and with pointer analysis!).
- **Build on existing infrastructure?** I would love to, there is no perfect match though.

Practical considerations

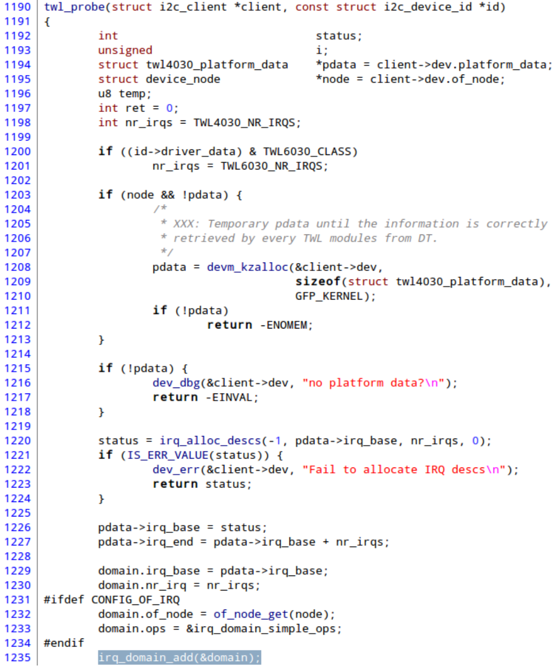
- **Handle *all* C?** Instead take partially preprocessed files.
- **Assembly code?** Support common functions built-in, and ignore the rest (yes, this is unsound).
- **False positives?** No, thanks.
- **Pointer analysis?** Of course, starting with Steensgaard unification-based algorithm (plus tweaks).
- **Data-flow analysis?** Use with care (and with pointer analysis!).
- **Build on existing infrastructure?**
  I would love to, there is no perfect match though.

More practical considerations

- **Infeasible paths:** Beyond the usual difficulties, some paths are determined unfeasible due to hardware specifications.
- **Interprocedural analysis:** Would interprocedural techniques scale? How many of these bugs can we found by intraprocedural analysis?
- **Aliasing:** Everywhere. Yet, Linux seems to satisfy the observations made by Manuvir Das[1].
- **Function pointers:** Linux uses (nested) structs of function pointers to represent interfaces for objects like drivers.

[1] *Unification-based pointer analysis with directional assignments.* PLDI'00

Thank you



```
twl_probe(struct i2c_client *client, const struct i2c_device_id *id)
{
        int                             status;
        unsigned                        i;
        struct twl4030_platform_data    *pdata = client->dev.platform_data;
        struct device_node              *node = client->dev.of_node;
        u8 temp;
        int ret = 0;
        int nr_irqs = TWL4030_NR_IRQS;

        if ((id->driver_data) & TWL6030_CLASS)
                nr_irqs = TWL6030_NR_IRQS;

        if (node && !pdata) {
                /*
                 * XXX: Temporary pdata until the information is correctly
                 * retrieved by every TWL modules from DT.
                 */
                pdata = devm_kzalloc(&client->dev,
                                     sizeof(struct twl4030_platform_data),
                                     GFP_KERNEL);
                if (!pdata)
                        return -ENOMEM;
        }

        if (!pdata) {
                dev_dbg(&client->dev, "no platform data?\n");
                return -EINVAL;
        }

        status = irq_alloc_descs(-1, pdata->irq_base, nr_irqs, 0);
        if (IS_ERR_VALUE(status)) {
                dev_err(&client->dev, "Fail to allocate IRQ descs\n");
                return status;
        }

        pdata->irq_base = status;
        pdata->irq_end = pdata->irq_base + nr_irqs;

        domain.irq_base = pdata->irq_base;
        domain.nr_irq = nr_irqs;
#ifdef CONFIG_OF_IRQ
        domain.of_node = of_node_get(node);
        domain.ops = &irq_domain_simple_ops;
#endif
        irq_domain_add(&domain);
```