A collection of variability-induced bugs in the Linux kernel

Iago Abal Rivas
IT University of Copenhagen

VARIETE project workshop
November 18, 2013

Motivation

Let's say you want to start research on verification of software product lines (SPL) **today**:

- What kind of SPL bugs can be found?
- What kind of feature interactions induce them?
- How can you benchmark your prototype ...
  - without running it on the real system?

Motivation

Let's say that I give you a handful of bugs:

- It takes time to understand the underlying cause.
- You don't have the required technical background.
- You lack a simplified view of the problem.

The Linux kernel as an SPL

- Prime example of software product line.
  - Open source, freely available.
- Roughly 6 million lines of C code.
- More than 6 thousand features.

Agenda

How do we find these bugs?

How does an SPL bug look like?

A few facts about our data

Methodology

1. Look at commits made to the stable branch.
2. Filter those that *look like* fixing a bug.
3. Filter those that *look like* variability-induced.
4. Analyze each potential candidate.

Look at commits made to the stable branch

```
9948133 pktcdvd: convert printk to pr_<level>
5323fb7 pktcdvd: convert ZONE macro to static function get_zone()
6723734 panic: call panic handlers before kmsg_dump
6325932 affs: use loff_t in affs_truncate()
5173b41 aoe: remove do-nothing NAME="%k" term from example udev rules
fea1b13 aoe: do not BUG if memory pressure prevented debugfs file creation
e0ec360 aoe: suppress compiler warnings
a88c1f0 aoe: remove custom implementation of kbasename()
896dcd9 aoe: update internal version number to 85
fc85799 [SCSI] fnic: fnic Driver Tuneables Exposed through CLI
ec34512 aoe: update copyright date
2256c1c aoe: fill in per-AoE-target information for debugfs file
1cf9479 aoe: provide file operations for debugfs files
e8866cf aoe: add AoE-target files to debugfs
190519c aoe: create and destroy debugfs directory for aoe
0bd4213 mm/zswap: use postorder iteration when destroying rbtree
7c993e1 rbtree: allow tests to run as builtin
a791a62 rbtree_test: add test for postorder iteration
2b52908 rbtree: add rbtree_postorder_for_each_entry_safe() helper
9dee5c5 rbtree: add postorder iteration functions
b4bc4a1 block/partitions/efi.c: consistently use pr_foo()
70f637e partitions/efi: some style cleanups
08009b3 partitions/efi: delete annoying emacs style comments
aa054bc partitions/efi: compare first and last usable LBAs
27a7c64 partitions/efi: account for pmbr size in lba
b05ebbb partitions/efi: detect hybrid MBRs
3e69ac3 partitions/efi: do not require gpt partition to begin at sector 1
33afd7a partitions/efi: check pmbr record's starting lba
c2ebdc2 partitions/efi: use lba-aware partition records
6f79d33 s390/vmcore: use vmcore for zfcpdump
11e376a vmcore: enable /proc/vmcore mmap for s390
23df79d s390/vmcore: implement remap_oldmem_pfn_range for s390
9cb2181 vmcore: introduce remap_oldmem_pfn_range()
97b0f6f s390/vmcore: use ELF header in new memory feature
```

Filter those 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 00000
    pgd = c0004000
    [00000000] *pgd=00000000
    Internal error: Oops: 5 [#1] SMP
```

Filter those that *look like* variability-induced

```
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-induced

```
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
```

Analyze each potential candidate

Agenda

How do we find these bugs?

How does an SPL bug look like?

A few facts about our data

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
```

Bug 6252547

type: NULL pointer dereference

description: NULL pointer on !OF_IRQ gets dereferenced on IRQ_DOMAIN

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: I2C && TWL4030_CORE && IRQ_DOMAIN && !OF_IRQ

commit:

branch: linux-stable

hash: 6252547b8a7acced581b649af4ebf6d65f63a34b

source: git log -p --grep="end trace" | grep "CONFIG_"

fix-in: model, mapping

trace:

```
init/main.c:789:kernel_init()
...
drivers/base/dd.c:203:driver_probe_device()
    215: ret = really_probe(dev, drv);
drivers/base/dd.c:108:really_probe()
```

Agenda

How do we find these bugs?

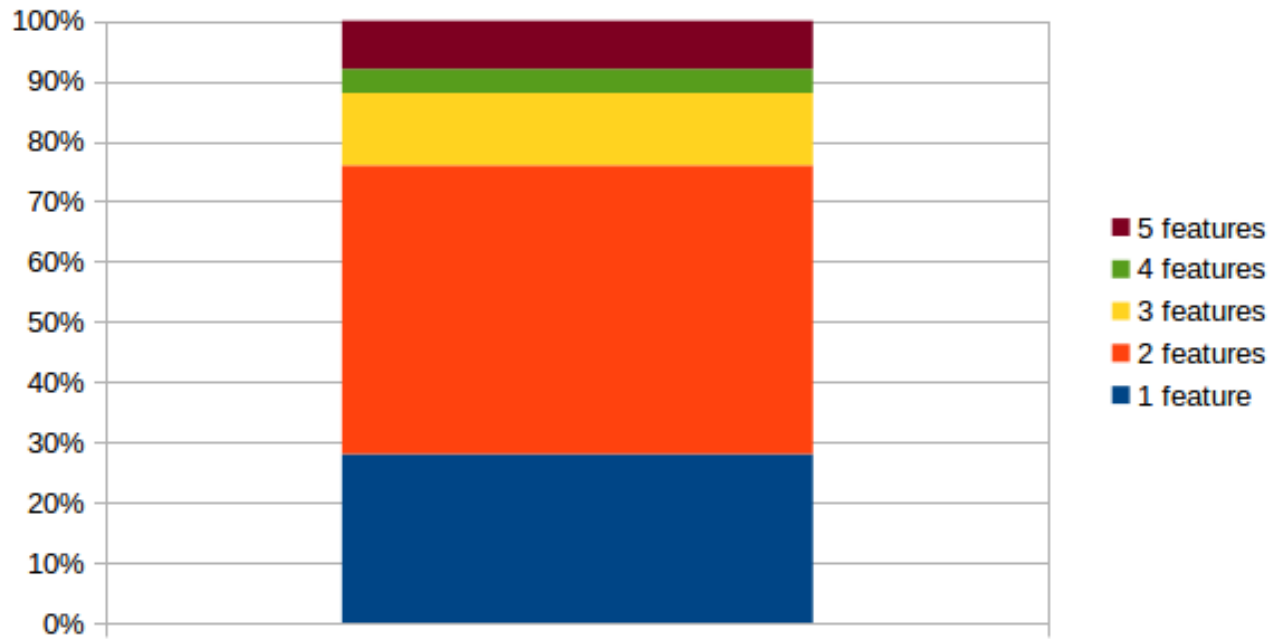
How does an SPL bug look like?

A few facts about our data

Our sample: features involved

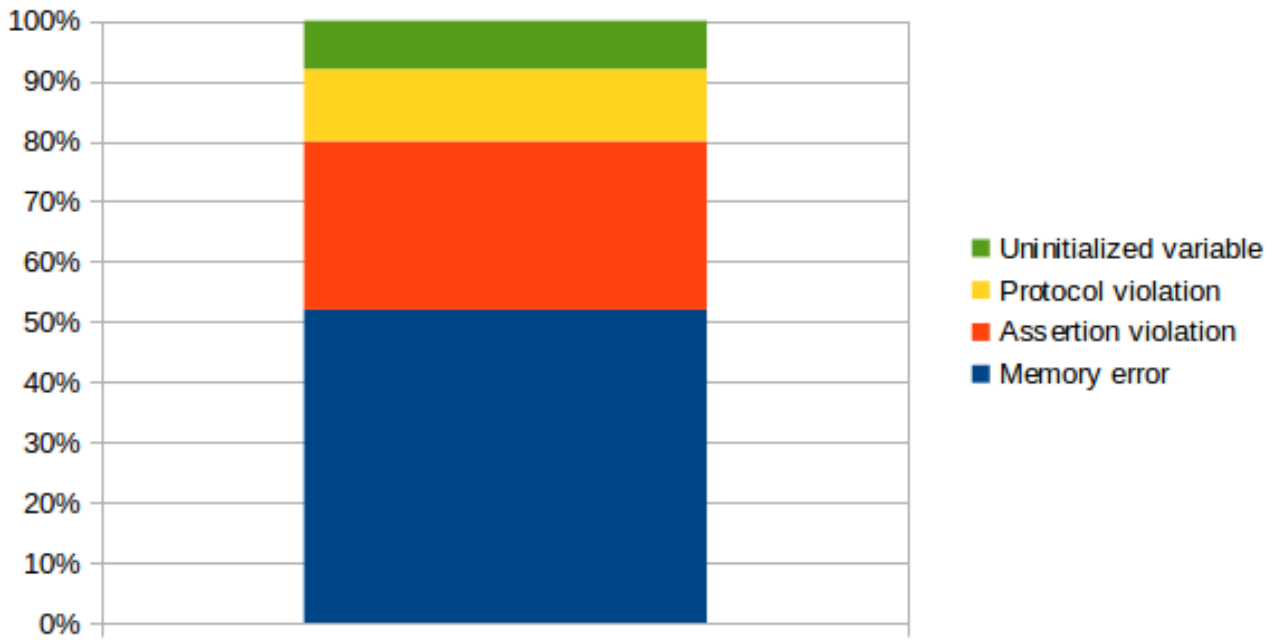
- ACPI_WMI
- ANDROID
- ARCH_OPAM3
- BDI_SWITCH
- DISCONTIGMEM
- DRM_I915
- EXTCON
- HIGHMEM
- HOTPLUG
- I2C
- IPV6
- IRQ_DOMAIN
- JFFS2_FS_WBUF_VERIFY
- KGDB
- KPROBES
- LOCKDEP
- MODULE_UNLOAD
- NETPOLL
- NUMA
- OF_IRQ
- PARISC
- PCI
- PM
- PPC64
- PREEMPT
- PROC_PAGE_MONITOR
- PROVE_LOCKING
- RCU_FAST_NO_HZ
- S390
- S390_PRNG
- SCTP_DBG_MSG
- SHMEM
- SLAB
- SMP
- SYSFS
- TCP_MD5SIG
- TMPFS
- TRACE_IRQFLAGS
- TREE_RCU
- TWL4030_CORE
- VLAN_8021Q
- X86
- X86_32
- XMON

Our sample: features per bug



Our sample: disabled features

Our sample: main types of bugs



Conclusion

Let's say you want to start research on verification of software product lines **in a few months**:

- You know what kind of SPL bugs can be found . . .
  - what kind of feature interactions induce them, and
  - understand the cause in a matter of *minutes*.
- You can benchmark your prototype . . .
  - without running it on the real system.

Conclusion

Let's say you want to start research on verification of software product lines **in a few months**:

- You know what kind of SPL bugs can be found . . .
  - what kind of feature interactions induce them, and
  - understand the cause in a matter of *minutes*.
- You can benchmark your prototype . . .
  - without running it on the real system.

I will do it as well :-)

Thank you