How to Avoid #ifdef Bugs in The Linux Kernel

Valentin Rothberg Daniel Lohmann

valentin.rothberg@lip6.fr dl@cs.fau.de

Inria / LIP6 Paris
and
System Software Group
Friedrich-Alexander University Erlangen-Nürnberg (FAU)

`https://www4.cs.fau.de/Research/CADOS/`

LPC '14

supported by DFG

What could possibly go wrong?

```
--- a/kernel/smp.c
+++ b/kernel/smp.c
@@ -34,8 +39,45 @@
[...]
+#ifdef CONFIG_CPU_HOTPLUG
+       case CPU_UP_CANCELED:
+       case CPU_UP_CANCELED_FROZEN:
+
+       case CPU_DEAD:
+       case CPU_DEAD_FROZEN:
+               free_cpumask_var(cfd->cpumask);
+               break;
+#endif
[...]
```

What could possibly go wrong?

```
--- a/kernel/smp.c
+++ b/kernel/smp.c
@@ -34,8 +39,45 @@
[...]
+#ifdef CONFIG_CPU_HOTPLUG
+       case CPU_UP_CANCELED:
+       case CPU_UP_CANCELED_FROZEN:
+
+       case CPU_DEAD:
+       case CPU_DEAD_FROZEN:
+               free_cpumask_var(cfd->cpumask);
+               break;
+#endif
[...]
```

- The kernel leaks memory!

What could possibly go wrong?

```
--- a/kernel/smp.c
+++ b/kernel/smp.c
@@ -34,8 +39,45 @@
[...]
+#ifdef CONFIG_CPU_HOTPLUG
+       case CPU_UP_CANCELED:
+       case CPU_UP_CANCELED_FROZEN:
+
+       case CPU_DEAD:
+       case CPU_DEAD_FROZEN:
+               free_cpumask_var(cfd->cpumask);
+               break;
+#endif
[...]
```

- The kernel leaks memory!
- CONFIG_CPU_HOTPLUG does not exist
- CONFIG_HOTPLUG_CPU is the right option

Undefined CPP Identifiers

- Undefined CPP identifiers evaluate to **false**

Undefined CPP Identifiers

- Undefined CPP identifiers evaluate to **false**
- This can lead to **dead** `#ifdef` blocks ...

```
#ifdef CONFIG_UNDEFINED

    /* I will never see the compiler :( */

#endif
```

Undefined CPP Identifiers

- Undefined CPP identifiers evaluate to **false**
- This can lead to **dead** `#ifdef` blocks ...

```
#ifdef CONFIG_UNDEFINED

    /* I will never see the compiler :( */

#endif
```

- ... and **undead** `#ifdef` blocks

```
#ifdef !CONFIG_UNDEFINED

    /* I will always see the compiler :( */

#endif
```

Undefined Kconfig Identifiers

- Undefined Kconfig identifiers evaluate to **false** ('n')

Undefined Kconfig Identifiers

- Undefined Kconfig identifiers evaluate to **false** ('n')
- This is a problem for Kconfig statements and expressions

Undefined Kconfig Identifiers

- Undefined Kconfig identifiers evaluate to **false** ('n')

- This is a problem for Kconfig statements and expressions

```
config HOTPLUG_CPU
    bool
    depends on UNDEFINED
```

Undefined Kconfig Identifiers

- Undefined Kconfig identifiers evaluate to **false** ('n')
- This is a problem for Kconfig statements and expressions

```
config HOTPLUG_CPU
    bool
    depends on UNDEFINED


if UNDEFINED
    config HOTPLUG_CPU
        bool
```

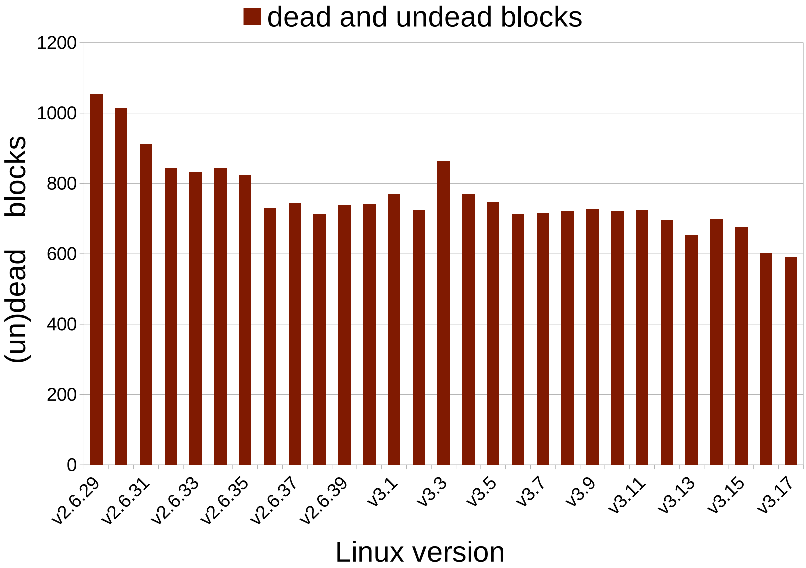
Undefined Kconfig Identifiers

- Undefined Kconfig identifiers evaluate to **false** ('n')
- This is a problem for Kconfig statements and expressions

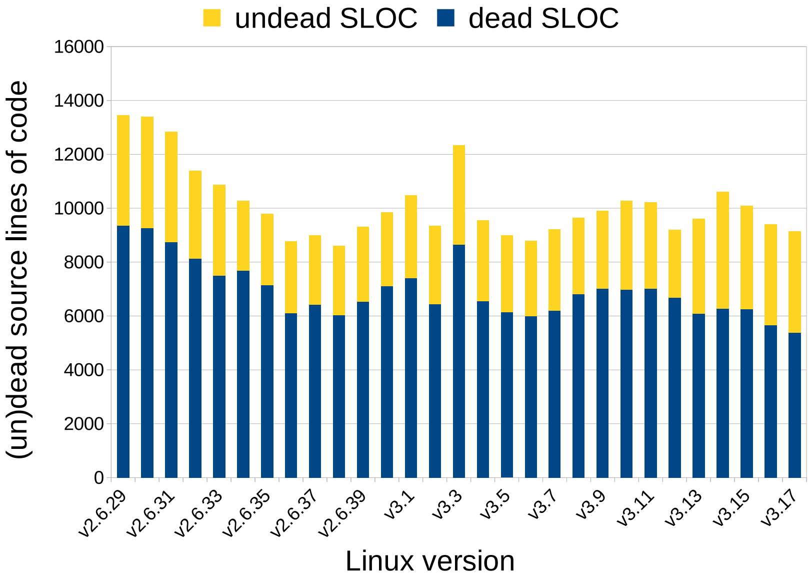
```
config HOTPLUG_CPU
    bool
    depends on UNDEFINED


if UNDEFINED
    config HOTPLUG_CPU
        bool
```

- Such issues manifest in **dead** and **undead** `#ifdef` blocks

(Un)Dead `#ifdef` **blocks** per Linux version



(Un)Dead **SLOC** per Linux version



Potential impacts of (un)dead code

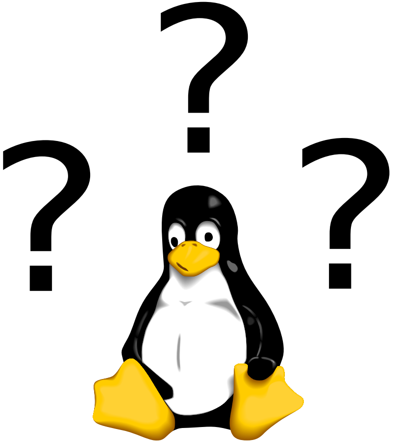
- `#ifdef` blocks are intentionally conditional
- dead and undead blocks violate this intention

Potential impacts of (un)dead code

- #ifdef blocks are intentionally conditional
- dead and undead blocks violate this intention

```
    [...]
#ifdef CONFIG_HIGHMEM_START_BOOL
    ioremap_base = CONFIG_HIGHMEM_START;
#else
    ioremap_base = 0xfe000000UL;   /* ... */
#endif /* CONFIG_HIGHMEM_START_BOOL */
    ioremap_bot = ioremap_base;

    /* Initialize the context management stuff */
    mmu_context_init();
}
```

How can we avoid these defects?



By using my tool, **undertaker-checkpatch**!

By using my tool, **undertaker-checkpatch**!

- It checks Git commits for `#ifdef` bugs

By using my tool, **undertaker-checkpatch**!

- It checks Git commits for `#ifdef` bugs
- It can be used like checkpatch.pl

By using my tool, **undertaker-checkpatch**!

- It checks Git commits for `#ifdef` bugs
- It can be used like checkpatch.pl
- It reports if such bugs are added or repaired

By using my tool, **undertaker-checkpatch**!

- It checks Git commits for `#ifdef` bugs
- It can be used like checkpatch.pl
- It reports if such bugs are added or repaired
- It analyzes the bugs and displays the bug-causing identifiers

By using my tool, **undertaker-checkpatch**!

- It checks Git commits for `#ifdef` bugs
- It can be used like checkpatch.pl
- It reports if such bugs are added or repaired
- It analyzes the bugs and displays the bug-causing identifiers
- It prevents (un)dead blocks by checking Kconfig changes

Example: Is this patch okay?

```
--- a/arch/arm/mach-omap2/board-h4.c
+++ b/arch/arm/mach-omap2/board-h4.c
@@ -379,6 +379,39 @@ ...
         .ctrl_name      = "internal",
 };

+static struct omap_usb_config h4_usb_config ....
+#ifdef CONFIG_MACH_OMAP2_H4_USB1
+       /* NOTE:  usb1 could also be used with 3 ...
+       .pins[1]        = 4,
+#endif
+
+#ifdef CONFIG_MACH_OMAP_H4_OTG
+       /* S1.10 ON -- USB OTG port
...
```

No, it’s broken!

```
user@abc:∼linux$ undertaker-checkpatch patch
```

No, it's broken!

```
user@abc:∼linux$ undertaker-checkpatch patch

New defect: arch/arm/mach-omap2/board-h4.c:
    B0:383:386:missing.globally.dead:
    CONFIG_MACH_OMAP2_H4_USB1 referenced but not defined

New defect: arch/arm/mach-omap2/board-h4.c:
    B1:388:403:missing.globally.dead:
    CONFIG_MACH_OMAP_H4_OTG referenced but not defined
```

Kconfig changes are critical

- Renaming / removing a feature without propagating the change

Kconfig changes are critical

- Renaming / removing a feature without propagating the change

```
--- a/arch/arm/mach-ixp23xx/Kconfig
+++ /dev/null
@@ -1,25 +0,0 @@
...
-config MACH_IXDP2351
-       bool "Support Intel IXDP2351 platform"
-       help
```

Kconfig changes are critical

- Renaming / removing a feature without propagating the change

```
--- a/arch/arm/mach-ixp23xx/Kconfig
+++ /dev/null
@@ -1,25 +0,0 @@
...
-config MACH_IXDP2351
-       bool "Support Intel IXDP2351 platform"
-       help
```

- **undertaker-checkpatch** displays leftover references

```
Feature CONFIG_MACH_IXDP2351 is removed
    but still referenced in:
drivers/net/ethernet/cirrus/cs89x0.c:176:
#if defined(CONFIG_MACH_IXDP2351)
```

Example: Logical Constraints

```
#ifdef CONFIG_X86_X2APIC /* depends on INTR_REMAP */

#ifdef CONFIG_INTR_REMAP
    /* I am undead */
#else
    /* I am dead  */
#endif

#endif
```

[1] http://undertaker.cs.fau.de

Example: Logical Constraints

```
#ifdef CONFIG_X86_X2APIC /* depends on INTR_REMAP */

#ifdef CONFIG_INTR_REMAP
    /* I am undead */
#else
    /* I am dead  */
#endif

#endif
```

- **25%** of (un)dead blocks are caused on a logic level
- I use the **Undertaker** [1] toolsuite to detect such logic issues

[1] http://undertaker.cs.fau.de

Conclusion

- **undertaker-checkpatch** detects, and further analyzes `#ifdef` bugs

[2] https://www4.cs.fau.de/Publications/2014/tartler_14_usenix.pdf

Conclusion

- **undertaker-checkpatch** detects, and further analyzes `#ifdef` bugs
  - It integrates into a developer's work flow (like checkpatch.pl)

[2] https://www4.cs.fau.de/Publications/2014/tartler_14_usenix.pdf

Conclusion

- **undertaker-checkpatch** detects, and further analyzes `#ifdef` bugs
  - It integrates into a developer's work flow (like checkpatch.pl)
  - It can be used on automated testing systems as well

[2] `https://www4.cs.fau.de/Publications/2014/tartler_14_usenix.pdf`

Conclusion

- **undertaker-checkpatch** detects, and further analyzes `#ifdef` bugs
  - It integrates into a developer's work flow (like checkpatch.pl)
  - It can be used on automated testing systems as well
  - Helps to detect and analyze **symbolic** and **logic defects**

[2] `https://www4.cs.fau.de/Publications/2014/tartler_14_usenix.pdf`

Conclusion

- **undertaker-checkpatch** detects, and further analyzes #ifdef bugs
  - It integrates into a developer's work flow (like checkpatch.pl)
  - It can be used on automated testing systems as well
  - Helps to detect and analyze **symbolic** and **logic defects**
- **Future work**: configurability aware compile-testing of patches

[2] https://www4.cs.fau.de/Publications/2014/tartler_14_usenix.pdf

Conclusion

- **undertaker-checkpatch** detects, and further analyzes `#ifdef` bugs
  - It integrates into a developer's work flow (like checkpatch.pl)
  - It can be used on automated testing systems as well
  - Helps to detect and analyze **symbolic** and **logic defects**
- **Future work**: configurability aware compile-testing of patches

"Frankly, most of the sw configuration ones tend to be annoyances rather than anything hugely fundamental. Compile warnings or failures that developers don't notice because it's not the configuration they use." [Linus Torvalds]

[2] `https://www4.cs.fau.de/Publications/2014/tartler_14_usenix.pdf`

Conclusion

- **undertaker-checkpatch** detects, and further analyzes `#ifdef` bugs
  - It integrates into a developer's work flow (like checkpatch.pl)
  - It can be used on automated testing systems as well
  - Helps to detect and analyze **symbolic** and **logic defects**
- **Future work**: configurability aware compile-testing of patches

"Frankly, most of the sw configuration ones tend to be annoyances rather than anything hugely fundamental. Compile warnings or failures that developers don't notice because it's not the configuration they use." [Linus Torvalds]

⇒ We have a tool to do that, the **Vampyr** [2]

[2] `https://www4.cs.fau.de/Publications/2014/tartler_14_usenix.pdf`

Interested?

- Download and try the tool:

  `http://undertaker.cs.fau.de`

- More information and papers on the project's website:

  `https://cados.cs.fau.de`

- Questions? Contact me directly ...

  valentin.rothberg@lip6.fr

- ... or write to our mailing list!

  cados-dev@lists.cs.fau.de