Subject: [PATCH] Report that kernel is tainted if there were an OOPS before (v3)
Posted by Pavel Emelianov on Thu, 07 Jun 2007 12:47:11 GMT
View Forum Message <> Reply to Message

If the kernel OOPSed or BUGed then it probably should
be considered as tainted. Thus, all subsequent OOPSes
and SysRq dumps will report the tainted kernel. This
saves a lot of time explaining oddities in the calltraces.

The previous version was buggy and reported the kernel
to be tainted at the very first oops as was noticed by
Dave Jones in the report from Antonino Daplas.

Compilation is checked for i386, x86_64 and ia64 since
I have no all the others at hands :)

Signed-off-by: Pavel Emelianov <xemul@openvz.org>
Cc: Randy Dunlap <randy.dunlap@oracle.com>

```
diff --git a/Documentation/oops-tracing.txt b/Documentation/oops-tracing.txt
index 7d5b60d..610e234 100644
--- a/Documentation/oops-tracing.txt
+++ b/Documentation/oops-tracing.txt
@@ -237,6 +237,8 @@ characters, each representing a particul
   7: 'U' if a user or user application specifically requested that the
      Tainted flag be set, ' ' otherwise.

+  8: 'D' if the kernel has died recently, i.e. there was an OOPS or BUG.
+
 The primary reason for the 'Tainted: ' string is to tell kernel
 debuggers if this is a clean kernel or if anything unusual has
 occurred.  Tainting is permanent: even if an offending module is
diff --git a/arch/alpha/kernel/traps.c b/arch/alpha/kernel/traps.c
index d6e665d..ec0f05e 100644
--- a/arch/alpha/kernel/traps.c
+++ b/arch/alpha/kernel/traps.c
@@ -184,6 +184,7 @@ die_if_kernel(char * str, struct pt_regs
 #endif
  printk("%s(%d): %s %ld\n", current->comm, current->pid, str, err);
  dik_show_regs(regs, r9_15);
+ add_taint(TAINT_DIE);
  dik_show_trace((unsigned long *)(regs+1));
  dik_show_code((unsigned int *)regs->pc);

diff --git a/arch/arm/kernel/traps.c b/arch/arm/kernel/traps.c
index 10ff36e..8c7c5d9 100644
```

```
--- a/arch/arm/kernel/traps.c
+++ b/arch/arm/kernel/traps.c
@@ -237,6 +237,7 @@ NORET_TYPE void die(const char *str, str
  bust_spinlocks(1);
  __die(str, err, thread, regs);
  bust_spinlocks(0);
+ add_taint(TAINT_DIE);
  spin_unlock_irq(&die_lock);

  if (panic_on_oops)
diff --git a/arch/arm26/kernel/traps.c b/arch/arm26/kernel/traps.c
index d594fb5..2911e2e 100644
--- a/arch/arm26/kernel/traps.c
+++ b/arch/arm26/kernel/traps.c
@@ -185,6 +185,7 @@ NORET_TYPE void die(const char *str, str
  printk("Internal error: %s: %x\n", str, err);
  printk("CPU: %d\n", smp_processor_id());
  show_regs(regs);
+ add_taint(TAINT_DIE);
  printk("Process %s (pid: %d, stack limit = 0x%p)\n",
   current->comm, current->pid, end_of_stack(tsk));

diff --git a/arch/avr32/kernel/traps.c b/arch/avr32/kernel/traps.c
index 86d1075..2a1174b 100644
--- a/arch/avr32/kernel/traps.c
+++ b/arch/avr32/kernel/traps.c
@@ -56,6 +56,7 @@ void NORET_TYPE die(const char *str, str
  show_regs_log_lvl(regs, KERN_EMERG);
  show_stack_log_lvl(current, regs->sp, regs, KERN_EMERG);
  bust_spinlocks(0);
+ add_taint(TAINT_DIE);
  spin_unlock_irq(&die_lock);

  if (in_interrupt())
diff --git a/arch/i386/kernel/traps.c b/arch/i386/kernel/traps.c
index 90c533f..837dee5 100644
--- a/arch/i386/kernel/traps.c
+++ b/arch/i386/kernel/traps.c
@@ -506,6 +506,7 @@ void die(const char * str, struct pt_reg

  bust_spinlocks(0);
  die.lock_owner = -1;
+ add_taint(TAINT_DIE);
  spin_unlock_irqrestore(&die.lock, flags);

  if (!regs)
diff --git a/arch/ia64/kernel/traps.c b/arch/ia64/kernel/traps.c
index 15ad85d..3aeaf15 100644
```

```
--- a/arch/ia64/kernel/traps.c
+++ b/arch/ia64/kernel/traps.c
@@ -69,6 +69,7 @@ die (const char *str, struct pt_regs *re

  bust_spinlocks(0);
  die.lock_owner = -1;
+ add_taint(TAINT_DIE);
  spin_unlock_irq(&die.lock);

  if (panic_on_oops)
diff --git a/arch/m68k/kernel/traps.c b/arch/m68k/kernel/traps.c
index a27a4fa..4e2752a 100644
--- a/arch/m68k/kernel/traps.c
+++ b/arch/m68k/kernel/traps.c
@@ -1170,6 +1170,7 @@ void die_if_kernel (char *str, struct pt
  console_verbose();
  printk("%s: %08x\n",str,nr);
  show_registers(fp);
+ add_taint(TAINT_DIE);
  do_exit(SIGSEGV);
 }

diff --git a/arch/m68knommu/kernel/traps.c b/arch/m68knommu/kernel/traps.c
index bed5f47..fde04e1 100644
--- a/arch/m68knommu/kernel/traps.c
+++ b/arch/m68knommu/kernel/traps.c
@@ -83,6 +83,7 @@ void die_if_kernel(char *str, struct pt_
  printk(KERN_EMERG "Process %s (pid: %d, stackpage=%08lx)\n",
   current->comm, current->pid, PAGE_SIZE+(unsigned long)current);
  show_stack(NULL, (unsigned long *)fp);
+ add_taint(TAINT_DIE);
  do_exit(SIGSEGV);
 }

diff --git a/arch/mips/kernel/traps.c b/arch/mips/kernel/traps.c
index 200de02..e7d2311 100644
--- a/arch/mips/kernel/traps.c
+++ b/arch/mips/kernel/traps.c
@@ -325,6 +325,7 @@ NORET_TYPE void ATTRIB_NORET die(const c
 #endif /* CONFIG_MIPS_MT_SMTC */
  printk("%s[#%d]:\n", str, ++die_counter);
  show_registers(regs);
+ add_taint(TAINT_DIE);
  spin_unlock_irq(&die_lock);

  if (in_interrupt())
diff --git a/arch/powerpc/kernel/traps.c b/arch/powerpc/kernel/traps.c
index bf6445a..f444700 100644
```

```
--- a/arch/powerpc/kernel/traps.c
+++ b/arch/powerpc/kernel/traps.c
@@ -149,6 +149,7 @@ int die(const char *str, struct pt_regs

  bust_spinlocks(0);
  die.lock_owner = -1;
+ add_taint(TAINT_DIE);
  spin_unlock_irqrestore(&die.lock, flags);

  if (kexec_should_crash(current) ||
diff --git a/arch/ppc/kernel/traps.c b/arch/ppc/kernel/traps.c
index aea100b..d32e387 100644
--- a/arch/ppc/kernel/traps.c
+++ b/arch/ppc/kernel/traps.c
@@ -92,6 +92,7 @@ int die(const char * str, struct pt_regs
  if (nl)
   printk("\n");
  show_regs(fp);
+ add_taint(TAINT_DIE);
  spin_unlock_irq(&die_lock);
  /* do_exit() should take care of panic'ing from an interrupt
   * context so we don't handle it here
diff --git a/arch/s390/kernel/traps.c b/arch/s390/kernel/traps.c
index cbfe730..87559bc 100644
--- a/arch/s390/kernel/traps.c
+++ b/arch/s390/kernel/traps.c
@@ -260,6 +260,7 @@ void die(const char * str, struct pt_reg
  printk("%s: %04lx [#%d]\n", str, err & 0xffff, ++die_counter);
        show_regs(regs);
  bust_spinlocks(0);
+ add_taint(TAINT_DIE);
        spin_unlock_irq(&die_lock);
  if (in_interrupt())
   panic("Fatal exception in interrupt");
diff --git a/arch/sh/kernel/traps.c b/arch/sh/kernel/traps.c
index 5b75cb6..f5f1aba 100644
--- a/arch/sh/kernel/traps.c
+++ b/arch/sh/kernel/traps.c
@@ -101,6 +101,7 @@ void die(const char * str, struct pt_reg
     (unsigned long)task_stack_page(current));

  bust_spinlocks(0);
+ add_taint(TAINT_DIE);
  spin_unlock_irq(&die_lock);

  if (kexec_should_crash(current))
diff --git a/arch/sparc/kernel/traps.c b/arch/sparc/kernel/traps.c
index dc9ffea..3bc3bff 100644
```

```
--- a/arch/sparc/kernel/traps.c
+++ b/arch/sparc/kernel/traps.c
@@ -101,6 +101,7 @@ void die_if_kernel(char *str, struct pt_

  printk("%s(%d): %s [#%d]\n", current->comm, current->pid, str, ++die_counter);
  show_regs(regs);
+ add_taint(TAINT_DIE);

  __SAVE; __SAVE; __SAVE; __SAVE;
  __SAVE; __SAVE; __SAVE; __SAVE;
diff --git a/arch/sparc64/kernel/traps.c b/arch/sparc64/kernel/traps.c
index 00a9e32..6ef2d29 100644
--- a/arch/sparc64/kernel/traps.c
+++ b/arch/sparc64/kernel/traps.c
@@ -2225,6 +2225,7 @@ void die_if_kernel(char *str, struct pt_
  notify_die(DIE_OOPS, str, regs, 0, 255, SIGSEGV);
  __asm__ __volatile__("flushw");
  __show_regs(regs);
+ add_taint(TAINT_DIE);
  if (regs->tstate & TSTATE_PRIV) {
   struct reg_window *rw = (struct reg_window *)
    (regs->u_regs[UREG_FP] + STACK_BIAS);
diff --git a/arch/x86_64/kernel/traps.c b/arch/x86_64/kernel/traps.c
index d9a5b7a..b6c524f 100644
--- a/arch/x86_64/kernel/traps.c
+++ b/arch/x86_64/kernel/traps.c
@@ -589,6 +589,7 @@ void __kprobes __die(const char * str, s
  printk("\n");
  notify_die(DIE_OOPS, str, regs, err, current->thread.trap_no, SIGSEGV);
  show_registers(regs);
+ add_taint(TAINT_DIE);
  /* Executive summary in case the oops scrolled away */
  printk(KERN_ALERT "RIP ");
  printk_address(regs->rip);
diff --git a/arch/xtensa/kernel/traps.c b/arch/xtensa/kernel/traps.c
index 693ab26..c5e62f9 100644
--- a/arch/xtensa/kernel/traps.c
+++ b/arch/xtensa/kernel/traps.c
@@ -482,6 +482,7 @@ void die(const char * str, struct pt_reg
  if (!user_mode(regs))
   show_stack(NULL, (unsigned long*)regs->areg[1]);

+ add_taint(TAINT_DIE);
  spin_unlock_irq(&die_lock);

  if (in_interrupt())
diff --git a/include/linux/kernel.h b/include/linux/kernel.h
index b267ec0..4300bb4 100644
```

```
--- a/include/linux/kernel.h
+++ b/include/linux/kernel.h
@@ -210,6 +210,7 @@ extern enum system_states {
 #define TAINT_MACHINE_CHECK  (1<<4)
 #define TAINT_BAD_PAGE   (1<<5)
 #define TAINT_USER   (1<<6)
+#define TAINT_DIE   (1<<7)

 extern void dump_stack(void) __cold;

diff --git a/kernel/panic.c b/kernel/panic.c
index 623d182..f64f4c1 100644
--- a/kernel/panic.c
+++ b/kernel/panic.c
@@ -159,14 +159,15 @@ const char *print_tainted(void)
 {
  static char buf[20];
  if (tainted) {
-  snprintf(buf, sizeof(buf), "Tainted: %c%c%c%c%c%c%c",
+  snprintf(buf, sizeof(buf), "Tainted: %c%c%c%c%c%c%c%c",
     tainted & TAINT_PROPRIETARY_MODULE ? 'P' : 'G',
     tainted & TAINT_FORCED_MODULE ? 'F' : ' ',
     tainted & TAINT_UNSAFE_SMP ? 'S' : ' ',
     tainted & TAINT_FORCED_RMMOD ? 'R' : ' ',
      tainted & TAINT_MACHINE_CHECK ? 'M' : ' ',
     tainted & TAINT_BAD_PAGE ? 'B' : ' ',
-    tainted & TAINT_USER ? 'U' : ' ');
+    tainted & TAINT_USER ? 'U' : ' ',
+    tainted & TAINT_DIE ? 'D' : ' ');
  }
  else
   snprintf(buf, sizeof(buf), "Not tainted");
```

Subject: Re: [PATCH] Report that kernel is tainted if there were an OOPS before (v3)
Posted by Randy Dunlap on Thu, 07 Jun 2007 22:06:17 GMT
View Forum Message <> Reply to Message

On Thu, 07 Jun 2007 16:51:50 +0400 Pavel Emelianov wrote:

> If the kernel OOPSed or BUGed then it probably should
> be considered as tainted. Thus, all subsequent OOPSes
> and SysRq dumps will report the tainted kernel. This
> saves a lot of time explaining oddities in the calltraces.
>
> The previous version was buggy and reported the kernel
> to be tainted at the very first oops as was noticed by

> Dave Jones in the report from Antonino Daplas.
>
> Compilation is checked for i386, x86_64 and ia64 since
> I have no all the others at hands :)
>
> Signed-off-by: Pavel Emelianov <xemul@openvz.org>
> Cc: Randy Dunlap <randy.dunlap@oracle.com>

Acked-by: Randy Dunlap <randy.dunlap@oracle.com>

I would change the doc. comment below:

> ---
>
> diff --git a/Documentation/oops-tracing.txt b/Documentation/oops-tracing.txt
> index 7d5b60d..610e234 100644
> --- a/Documentation/oops-tracing.txt
> +++ b/Documentation/oops-tracing.txt
> @@ -237,6 +237,8 @@ characters, each representing a particul
>   7: 'U' if a user or user application specifically requested that the
>   Tainted flag be set, ' ' otherwise.
>
> + 8: 'D' if the kernel has died recently, i.e. there was an OOPS or BUG.

something like:

+ 8: 'D' if the kernel has already died, i.e. there was a prior OOPS or BUG.

> +
> The primary reason for the 'Tainted: ' string is to tell kernel
> debuggers if this is a clean kernel or if anything unusual has
> occurred. Tainting is permanent: even if an offending module is

---
~Randy
*** Remember to use Documentation/SubmitChecklist when testing your code ***