

## [patch] espfix cleanup take 3

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-09/msg09120.html>

---

- *From:* Stas Sergeev <stsp@xxxxxxxx>
  - *Date:* Sat, 30 Sep 2006 22:18:03 +0400
- 

Hello.

Andrew Morton wrote:

Ho hum, this conflicts moderately with the hypervisor preparatory patches which Jeremy sent.  
So could I ask that you redo this patch in a couple of weeks time against the current -mm lineup?

Done.

The attached patch cleans up the espfix code:

- Introduced PER\_CPU() macro to be used from asm
- Introduced GET\_DESC\_BASE() macro to be used from asm
- Rewrote the fixup code in asm, as calling a C code with the altered %ss appeared to be unsafe
- No longer altering the stack from a .fixup section
- 16bit per-cpu stack is no longer used, instead the stack segment base is patched the way so that the high word of the kernel and user %esp are the same.
- Added the limit-patching for the espfix segment. (Chuck Ebbert)

Signed-off-by: Stas Sergeev <stsp@xxxxxxxx>

Acked-by: Zachary Amsden <zach@xxxxxxxx>

Acked-by: Chuck Ebbert <76306.1226@xxxxxxxxxxxxxxxx>

Acked-by: Jan Beulich <jbeulich@xxxxxxxx>

```
diff -ur linux-2.6.18-mm2/arch/i386/kernel/asm-offsets.c
```

```
linux-2.6.18-mm2-stk/arch/i386/kernel/asm-offsets.c
```

```
--- linux-2.6.18-mm2/arch/i386/kernel/asm-offsets.c 2006-09-30 14:28:27.000000000 +0400
```

```
+++ linux-2.6.18-mm2-stk/arch/i386/kernel/asm-offsets.c 2006-09-30 20:18:41.000000000 +0400
```

```
@@ -58,6 +58,11 @@
```

```
OFFSET(TI_sysenter_return, thread_info, sysenter_return);
```

```
BLANK();
```

```
+ OFFSET(GDS_size, Xgt_desc_struct, size);
```

```
+ OFFSET(GDS_address, Xgt_desc_struct, address);
```

```
+ OFFSET(GDS_pad, Xgt_desc_struct, pad);
```

```
+ BLANK();
```

[patch] espfix cleanup take 3

```
+
OFFSET(EXEC_DOMAIN_handler, exec_domain, handler);
OFFSET(RT_SIGFRAME_sigcontext, rt_sigframe, uc.uc_mcontext);
BLANK();
diff -ur linux-2.6.18-mm2/arch/i386/kernel/cpu/common.c
linux-2.6.18-mm2-stk/arch/i386/kernel/cpu/common.c
--- linux-2.6.18-mm2/arch/i386/kernel/cpu/common.c 2006-09-30 16:27:56.000000000 +0400
+++ linux-2.6.18-mm2-stk/arch/i386/kernel/cpu/common.c 2006-09-30 20:18:39.000000000 +0400
@@ -24,9 +24,6 @@
DEFINE_PER_CPU(struct Xgt_desc_struct, cpu_gdt_descr);
EXPORT_PER_CPU_SYMBOL(cpu_gdt_descr);

-DEFINE_PER_CPU(unsigned char, cpu_16bit_stack[CPU_16BIT_STACK_SIZE]);
-EXPORT_PER_CPU_SYMBOL(cpu_16bit_stack);
-
static int cachesize_override __cpuinitdata = -1;
static int disable_x86_fxr __cpuinitdata;
static int disable_x86_serial_nr __cpuinitdata = 1;
@@ -594,7 +591,6 @@
struct tss_struct * t = &per_cpu(init_tss, cpu);
struct thread_struct *thread = &current->thread;
struct desc_struct *gdt;
- __u32 stk16_off = (__u32)&per_cpu(cpu_16bit_stack, cpu);
struct Xgt_desc_struct *cpu_gdt_descr = &per_cpu(cpu_gdt_descr, cpu);

if (cpu_test_and_set(cpu, cpu_initialized)) {
@@ -642,13 +638,6 @@
* and set up the GDT descriptor:
*/
memcpy(gdt, cpu_gdt_table, GDT_SIZE);
-
- /* Set up GDT entry for 16bit stack */
- *(__u64 *)&gdt[GDT_ENTRY_ESPFIX_SS] |=
- (((__u64)stk16_off) << 16) & 0x000000fffff0000ULL) |
- (((__u64)stk16_off) << 32) & 0xff00000000000000ULL) |
- (CPU_16BIT_STACK_SIZE - 1);
-
cpu_gdt_descr->size = GDT_SIZE - 1;
cpu_gdt_descr->address = (unsigned long)gdt;

diff -ur linux-2.6.18-mm2/arch/i386/kernel/entry.S linux-2.6.18-mm2-stk/arch/i386/kernel/entry.S
--- linux-2.6.18-mm2/arch/i386/kernel/entry.S 2006-09-30 16:27:56.000000000 +0400
+++ linux-2.6.18-mm2-stk/arch/i386/kernel/entry.S 2006-09-30 20:18:42.000000000 +0400
@@ -48,6 +48,7 @@
#include <asm/smp.h>
#include <asm/page.h>
#include <asm/desc.h>
+#include <asm/percpu.h>
#include <asm/dwarf2.h>
#include "irq_vectors.h"
```

[patch] espfix cleanup take 3

```
@@ -418,23 +419,18 @@
* This is an "official" bug of all the x86-compatible
* CPUs, which we can try to work around to make
* dosemu and wine happy. */
- subl $8, %esp # reserve space for switch16 pointer
- CFI_ADJUST_CFA_OFFSET 8
+ movl OLDESP(%esp), %eax
+ movl %esp, %edx
+ call patch_espfix_desc
+ pushl $__ESPFIX_SS
+ CFI_ADJUST_CFA_OFFSET 4
+ pushl %eax
+ CFI_ADJUST_CFA_OFFSET 4
DISABLE_INTERRUPTS
TRACE_IRQS_OFF
- movl %esp, %eax
- /* Set up the 16bit stack frame with switch32 pointer on top,
- * and a switch16 pointer on top of the current frame. */
- call setup_x86_bogus_stack
- CFI_ADJUST_CFA_OFFSET -8 # frame has moved
- TRACE_IRQS_IRET
- RESTORE_REGS
- lss 20+4(%esp), %esp # switch to 16bit stack
-1: INTERRUPT_RETURN
-.section __ex_table,"a"
- .align 4
- .long 1b,iret_exc
-.previous
+ lss (%esp), %esp
+ CFI_ADJUST_CFA_OFFSET -8
+ jmp restore_nocheck
CFI_ENDPROC

# perform work that needs to be done immediately before resumption
@@ -524,30 +520,30 @@
CFI_ENDPROC

#define FIXUP_ESPFIX_STACK \
- movl %esp, %eax; \
- /* switch to 32bit stack using the pointer on top of 16bit stack */ \
- lss %ss:CPU_16BIT_STACK_SIZE-8, %esp; \
- /* copy data from 16bit stack to 32bit stack */ \
- call fixup_x86_bogus_stack; \
- /* put ESP to the proper location */ \
- movl %eax, %esp;
-#define UNWIND_ESPFIX_STACK \
+ /* since we are on a wrong stack, we cant make it a C code :( */ \
+ GET_THREAD_INFO(%ebp); \
+ movl TI_cpu(%ebp), %ebx; \
+ PER_CPU(cpu_gdt_descr, %ebx); \
+ movl GDS_address(%ebx), %ebx; \
```

[patch] espfix cleanup take 3

```
+ GET_DESC_BASE(GDT_ENTRY_ESPFI_X_SS, %ebx, %eax, %ax, %al, %ah); \
+ addl %esp, %eax; \
+ pushl $__KERNEL_DS; \
+ CFI_ADJUST_CFA_OFFSET 4; \
pushl %eax; \
CFI_ADJUST_CFA_OFFSET 4; \
+ lss (%esp), %esp; \
+ CFI_ADJUST_CFA_OFFSET -8;
+#define UNWIND_ESPFI_X_STACK \
movl %ss, %eax; \
- /* see if on 16bit stack */ \
+ /* see if on espfix stack */ \
cmpw $__ESPFI_X_SS, %ax; \
- je 28f; \
-27: popl %eax; \
- CFI_ADJUST_CFA_OFFSET -4; \
-.section .fixup,"ax"; \
-28: movl $__KERNEL_DS, %eax; \
+ jne 27f; \
+ movl $__KERNEL_DS, %eax; \
movl %eax, %ds; \
movl %eax, %es; \
- /* switch to 32bit stack */ \
+ /* switch to normal stack */ \
FIXUP_ESPFI_X_STACK; \
- jmp 27b; \
-.previous
+27::

/*
* Build the entry stubs and pointer table with
@@ -614,7 +610,6 @@
pushl %eax
CFI_ADJUST_CFA_OFFSET 4
CFI_REL_OFFSET eax, 0
- xorl %eax, %eax
pushl %ebp
CFI_ADJUST_CFA_OFFSET 4
CFI_REL_OFFSET ebp, 0
@@ -627,7 +622,6 @@
pushl %edx
CFI_ADJUST_CFA_OFFSET 4
CFI_REL_OFFSET edx, 0
- decl %eax # eax = -1
pushl %ecx
CFI_ADJUST_CFA_OFFSET 4
CFI_REL_OFFSET ecx, 0
@@ -644,7 +638,7 @@
/*CFI_REGISTER es, ecx*/
movl ES(%esp), %edi # get the function address
movl ORIG_EAX(%esp), %edx # get the error code
```

[patch] espfix cleanup take 3

```
- movl %eax, ORIG_EAX(%esp)
+ movl $-1, ORIG_EAX(%esp)
movl %ecx, ES(%esp)
/*CFI_REL_OFFSET es, ES*/
movl $(__USER_DS), %ecx
@@ -754,7 +748,7 @@
cmpw $__ESPFIX_SS, %ax
popl %eax
CFI_ADJUST_CFA_OFFSET -4
- je nmi_16bit_stack
+ je nmi_espfix_stack
cmpl $sysenter_entry,(%esp)
je nmi_stack_fixup
pushl %eax
@@ -797,7 +791,7 @@
FIX_STACK(24,nmi_stack_correct, 1)
jmp nmi_stack_correct

-nmi_16bit_stack:
+nmi_espfix_stack:
/* We have a RING0_INT_FRAME here.
*
* create the pointer to lss back
@@ -806,7 +800,6 @@
CFI_ADJUST_CFA_OFFSET 4
pushl %esp
CFI_ADJUST_CFA_OFFSET 4
- movzwl %sp, %esp
addw $4, (%esp)
/* copy the iret frame of 12 bytes */
.rept 3
@@ -817,11 +810,11 @@
CFI_ADJUST_CFA_OFFSET 4
SAVE_ALL
FIXUP_ESPFIX_STACK # %eax == %esp
- CFI_ADJUST_CFA_OFFSET -20 # the frame has now moved
xorl %edx,%edx # zero error code
call do_nmi
RESTORE_REGS
- lss 12+4(%esp), %esp # back to 16bit stack
+ lss 12+4(%esp), %esp # back to espfix stack
+ CFI_ADJUST_CFA_OFFSET -24
1: INTERRUPT_RETURN
CFI_ENDPROC
.section __ex_table,"a"
diff -ur linux-2.6.18-mm2/arch/i386/kernel/head.S linux-2.6.18-mm2-stk/arch/i386/kernel/head.S
--- linux-2.6.18-mm2/arch/i386/kernel/head.S 2006-09-30 16:27:56.000000000 +0400
+++ linux-2.6.18-mm2-stk/arch/i386/kernel/head.S 2006-09-30 20:18:41.000000000 +0400
@@ -584,7 +584,7 @@
.quad 0x00009a000000ffff /* 0xc0 APM CS 16 code (16 bit) */
.quad 0x004092000000ffff /* 0xc8 APM DS data */
```



### [patch] espfix cleanup take 3

```
- /* copy the data from 16bit stack to 32bit stack */
- len = CPU_16BIT_STACK_SIZE - 8 - sp;
- stack16 = (unsigned char *)(stack_bot + sp);
- stack32 = (unsigned char *)
- (switch32_ptr[0] + CPU_16BIT_STACK_SIZE - 8 - len);
- memcpy(stack32, stack16, len);
- return stack32;
+ struct Xgt_desc_struct *cpu_gdt_descr = &per_cpu(cpu_gdt_descr, cpu);
+ struct desc_struct *gdt = (struct desc_struct *)cpu_gdt_descr->address;
+ unsigned long base = (kesp - uesp) & -THREAD_SIZE;
+ unsigned long new_ksesp = kesp - base;
+ unsigned long lim_pages = (new_ksesp | (THREAD_SIZE - 1)) >> PAGE_SHIFT;
+ __u64 desc = *((__u64 *)&gdt[GDT_ENTRY_ESPFIX_SS]);
+ /* Set up base for espfix segment */
+ desc &= 0x00f0ff0000000000ULL;
+ desc |= (((__u64)base) << 16) & 0x000000ffffff0000ULL |
+ (((__u64)base) << 32) & 0xff00000000000000ULL |
+ (((__u64)lim_pages) << 32) & 0x000f000000000000ULL |
+ (lim_pages & 0xffff);
+ *((__u64 *)&gdt[GDT_ENTRY_ESPFIX_SS]) = desc;
+ return new_ksesp;
}

/*
diff -ur linux-2.6.18-mm2/include/asm-i386/desc.h linux-2.6.18-mm2-stk/include/asm-i386/desc.h
--- linux-2.6.18-mm2/include/asm-i386/desc.h 2006-09-30 16:28:08.000000000 +0400
+++ linux-2.6.18-mm2-stk/include/asm-i386/desc.h 2006-09-30 20:20:49.000000000 +0400
@@ -4,8 +4,6 @@
#include <asm/ldt.h>
#include <asm/segment.h>

-#define CPU_16BIT_STACK_SIZE 1024
-
#ifdef __ASSEMBLY__

#include <linux/preempt.h>
@@ -16,8 +14,6 @@

extern struct desc_struct cpu_gdt_table[GDT_ENTRIES];

-DECLARE_PER_CPU(unsigned char, cpu_16bit_stack[CPU_16BIT_STACK_SIZE]);
-
struct Xgt_desc_struct {
unsigned short size;
unsigned long address __attribute__((packed));
@@ -181,6 +177,29 @@
return base;
}

+#else /* __ASSEMBLY__ */
+

```

[patch] espfix cleanup take 3

```
+/*
+ * GET_DESC_BASE reads the descriptor base of the specified segment.
+ *
+ * Args:
+ * idx – descriptor index
+ * gdt – GDT pointer
+ * base – 32bit register to which the base will be written
+ * lo_w – lo word of the "base" register
+ * lo_b – lo byte of the "base" register
+ * hi_b – hi byte of the low word of the "base" register
+ *
+ * Example:
+ * GET_DESC_BASE(GDT_ENTRY_ESPFISS, %ebx, %eax, %ax, %al, %ah)
+ * Will read the base address of GDT_ENTRY_ESPFISS and put it into %eax.
+ */
+#define GET_DESC_BASE(idx, gdt, base, lo_w, lo_b, hi_b) \
+ movb idx*8+4(gdt), lo_b; \
+ movb idx*8+7(gdt), hi_b; \
+ shll $16, base; \
+ movw idx*8+2(gdt), lo_w;
+
#endif /* !__ASSEMBLY__ */

#endif
diff -ur linux-2.6.18-mm2/include/asm-i386/percpu.h linux-2.6.18-mm2-stk/include/asm-i386/percpu.h
--- linux-2.6.18-mm2/include/asm-i386/percpu.h 2004-01-09 10:00:03.000000000 +0300
+++ linux-2.6.18-mm2-stk/include/asm-i386/percpu.h 2006-09-30 20:20:49.000000000 +0400
@@ -1,6 +1,27 @@
#ifndef __ARCH_I386_PERCPU__
#define __ARCH_I386_PERCPU__

#ifndef __ASSEMBLY__
#include <asm-generic/percpu.h>
#else
+
+/*
+ * PER_CPU finds an address of a per-cpu variable.
+ *
+ * Args:
+ * var – variable name
+ * cpu – 32bit register containing the current CPU number
+ *
+ * The resulting address is stored in the "cpu" argument.
+ *
+ * Example:
+ * PER_CPU(cpu_gdt_descr, %ebx)
+ */
+#define PER_CPU(var, cpu) \
+ shll $2, cpu; \
+ movl __per_cpu_offset(cpu), cpu; \
+ addl $per_cpu___*/var, cpu;
```

[patch] espfix cleanup take 3

```
+  
+#endif /* !__ASSEMBLY__ */  
  
#endif /* __ARCH_I386_PERCPU__ */
```