

Bug in es1968 driver

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-03/msg12055.html>

- *From:* Arjan van de Ven <arjan@xxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 31 Mar 2008 08:55:29 -0700
-

www.kerneloops.org has found a bug in the es1968 driver (as one of the more popular issues):

```
static unsigned short snd_es1968_ac97_read(struct snd_ac97 *ac97, unsigned short reg)
{
    u16 data = 0;
    struct es1968 *chip = ac97->private_data;
    unsigned long flags;

    snd_es1968_ac97_wait(chip);

    spin_lock_irqsave(&chip->ac97_lock, flags);
    outb(reg | 0x80, chip->io_port + ESM_AC97_INDEX);
    /*msleep(1);*/

    if (!snd_es1968_ac97_wait(chip)) {
        data = inw(chip->io_port + ESM_AC97_DATA);
        /*msleep(1);*/
    }
    spin_unlock_irqrestore(&chip->ac97_lock, flags);

    return data;
}
```

in this file, `snd_es1968_ac97_wait()` is a sleeping function... which conveniently gets called while the `ac97_lock` spinlock is held....

patch below (and attached in case thunderbird decides to pee all over it):

From: Arjan van de Ven <arjan@xxxxxxxxxxxxxxxxx>
Subject: [PATCH] Fix sleep-while-holding-lock bug in es1968

`snd_es1968_ac97_read()` calls `snd_es1968_ac97_wait()` first outside a locked area, and later, while holding a lock.
`snd_es1968_ac97_wait()` has a polling loop with a `cond_resched()` inside it.. which sleeps, so the second call is invalid.

This patch adds a version of the wait function that just pure polls. While this is not very elegant in principle, it's very likely the easiest thing to do here,

Bug in es1968 driver

we already checked if the chip was ready (while yielding) just before, so it is very unlikely to take a long time here.

Signed-off-by: Arjan van de Ven <arjan@xxxxxxxxxxxxxxxx>

sound/pci/es1968.c | 14 ++++++++
1 files changed, 13 insertions(+), 1 deletions(-)

```
diff --git a/sound/pci/es1968.c b/sound/pci/es1968.c
index 25ccfce..71d459d 100644
--- a/sound/pci/es1968.c
+++ b/sound/pci/es1968.c
@@ -617,6 +617,18 @@ static int snd_es1968_ac97_wait(struct es1968 *chip)
return 1; /* timeout */
}

+static int snd_es1968_ac97_wait_poll(struct es1968 *chip)
+{
+ int timeout = 100000;
+
+ while (timeout-- > 0) {
+ if (!(inb(chip->io_port + ESM_AC97_INDEX) & 1))
+ return 0;
+ }
+ snd_printd("es1968: ac97 timeout\n");
+ return 1; /* timeout */
+}
+
static void snd_es1968_ac97_write(struct snd_ac97 *ac97, unsigned short reg, unsigned short val)
{
struct es1968 *chip = ac97->private_data;
@@ -645,7 +657,7 @@ static unsigned short snd_es1968_ac97_read(struct snd_ac97 *ac97, unsigned short
outb(reg | 0x80, chip->io_port + ESM_AC97_INDEX);
/*msleep(1);*/

- if (! snd_es1968_ac97_wait(chip)) {
+ if (! snd_es1968_ac97_wait_poll(chip)) {
data = inw(chip->io_port + ESM_AC97_DATA);
/*msleep(1);*/
}
}
-----
```

1.5.4.4

From 1ab58f99320e4d842c7e6ccc6db76c286f1902 Mon Sep 17 00:00:00 2001

From: Arjan van de Ven <arjan@xxxxxxxxxxxxxxxx>
Date: Mon, 31 Mar 2008 05:20:58 -0700
Subject: [PATCH] Fix sleep-while-holding-lock bug in es1968

Bug in es1968 driver

Bug in es1968 driver

snd_es1968_ac97_read() calls snd_es1968_ac97_wait() first outside a locked area, and later, while holding a lock.

snd_es1968_ac97_wait() has a polling loop with a cond_resched() inside it.. which sleeps, so the second call is invalid.

This patch adds a version of the wait function that just pure polls. While this is not very elegant in principle, it's very likely the easiest thing to do here, we already checked if the chip was ready (while yielding) just before, so it is very unlikely to take a long time here.

Signed-off-by: Arjan van de Ven <arjan@xxxxxxxxxxxxxxxx>

sound/pci/es1968.c | 14 ++++++++
1 files changed, 13 insertions(+), 1 deletions(-)

diff --git a/sound/pci/es1968.c b/sound/pci/es1968.c
index 25ccfce..71d459d 100644

--- a/sound/pci/es1968.c

+++ b/sound/pci/es1968.c

```
@@ -617,6 +617,18 @@ static int snd_es1968_ac97_wait(struct es1968 *chip)
return 1; /* timeout */
}
```

```
+static int snd_es1968_ac97_wait_poll(struct es1968 *chip)
```

```
+{
```

```
+ int timeout = 100000;
```

```
+
```

```
+ while (timeout-- > 0) {
```

```
+ if (!(inb(chip->io_port + ESM_AC97_INDEX) & 1))
```

```
+ return 0;
```

```
+ }
```

```
+ snd_printd("es1968: ac97 timeout\n");
```

```
+ return 1; /* timeout */
```

```
+
```

```
static void snd_es1968_ac97_write(struct snd_ac97 *ac97, unsigned short reg, unsigned short val)
{
```

```
struct es1968 *chip = ac97->private_data;
```

```
@@ -645,7 +657,7 @@ static unsigned short snd_es1968_ac97_read(struct snd_ac97 *ac97, unsigned short
outb(reg | 0x80, chip->io_port + ESM_AC97_INDEX);
```

```
/*msleep(1);*/
```

```
- if (! snd_es1968_ac97_wait(chip)) {
```

```
+ if (! snd_es1968_ac97_wait_poll(chip)) {
```

```
data = inw(chip->io_port + ESM_AC97_DATA);
```

```
/*msleep(1);*/
```

```
}
```

1.5.4.4