Subject: [patch i2o 1/6] i2o_cfg_passthru cleanup
Posted by vaverin on Tue, 15 May 2007 12:42:26 GMT
View Forum Message <> Reply to Message

This patch fixes a number of issues in i2o_cfg_passthru{,32}:
- i2o_msg_get_wait() return vaile is not checked;
- i2o_message memory leaks on error paths;
- infinite loop to sg_list_cleanup in passthru32

it's important issue because of i2o_cfg_passthru is used by raidutils for monitorig controllers state, and in case of memory shortage it leads to the node crash or disk IO stall.

Signed-off-by: Vasily Averin <vvs@sw.ru>

```
--- lk2.6/drivers/message/i2o/i2o_config.c
+++ lk2.6/drivers/message/i2o/i2o_config.c
@@ -554,8 +554,6 @@ static int i2o_cfg_passthru32(struct fil
   return -ENXIO;
  }

- msg = i2o_msg_get_wait(c, I2O_TIMEOUT_MESSAGE_GET);
-
  sb = c->status_block.virt;

  if (get_user(size, &user_msg[0])) {
@@ -573,24 +571,30 @@ static int i2o_cfg_passthru32(struct fil

  size <<= 2;  // Convert to bytes

+ msg = i2o_msg_get_wait(c, I2O_TIMEOUT_MESSAGE_GET);
+ if (IS_ERR(msg))
+  return PTR_ERR(msg);
+
+ rcode = -EFAULT;
  /* Copy in the user's I2O command */
  if (copy_from_user(msg, user_msg, size)) {
   osm_warn("unable to copy user message\n");
-  return -EFAULT;
+  goto out;
  }
  i2o_dump_message(msg);

  if (get_user(reply_size, &user_reply[0]) < 0)
-  return -EFAULT;
+  goto out;

  reply_size >>= 16;
```

```
  reply_size <<= 2;

+ rcode = -ENOMEM;
  reply = kzalloc(reply_size, GFP_KERNEL);
  if (!reply) {
   printk(KERN_WARNING "%s: Could not allocate reply buffer\n",
          c->name);
-  return -ENOMEM;
+  goto out;
  }

  sg_offset = (msg->u.head[0] >> 4) & 0x0f;
@@ -661,13 +665,14 @@ static int i2o_cfg_passthru32(struct fil
  }

  rcode = i2o_msg_post_wait(c, msg, 60);
+ msg = NULL;
  if (rcode) {
   reply[4] = ((u32) rcode) << 24;
   goto sg_list_cleanup;
  }

  if (sg_offset) {
-  u32 msg[I2O_OUTBOUND_MSG_FRAME_SIZE];
+  u32 rmsg[I2O_OUTBOUND_MSG_FRAME_SIZE];
   /* Copy back the Scatter Gather buffers back to user space */
   u32 j;
   // TODO 64bit fix
@@ -675,7 +680,7 @@ static int i2o_cfg_passthru32(struct fil
   int sg_size;

   // re-acquire the original message to handle correctly the sg copy operation
-  memset(&msg, 0, I2O_OUTBOUND_MSG_FRAME_SIZE * 4);
+  memset(&rmsg, 0, I2O_OUTBOUND_MSG_FRAME_SIZE * 4);
   // get user msg size in u32s
   if (get_user(size, &user_msg[0])) {
    rcode = -EFAULT;
@@ -684,7 +689,7 @@ static int i2o_cfg_passthru32(struct fil
   size = size >> 16;
   size *= 4;
   /* Copy in the user's I2O command */
-  if (copy_from_user(msg, user_msg, size)) {
+  if (copy_from_user(rmsg, user_msg, size)) {
    rcode = -EFAULT;
    goto sg_list_cleanup;
   }
@@ -692,7 +697,7 @@ static int i2o_cfg_passthru32(struct fil
       (size - sg_offset * 4) / sizeof(struct sg_simple_element);
```

```
   // TODO 64bit fix
-  sg = (struct sg_simple_element *)(msg + sg_offset);
+  sg = (struct sg_simple_element *)(rmsg + sg_offset);
   for (j = 0; j < sg_count; j++) {
    /* Copy out the SG list to user's buffer if necessary */
    if (!
@@ -714,7 +719,7 @@ static int i2o_cfg_passthru32(struct fil
   }
  }

-      sg_list_cleanup:
+sg_list_cleanup:
  /* Copy back the reply to user space */
  if (reply_size) {
   // we wrote our own values for context - now restore the user supplied ones
@@ -723,7 +728,6 @@ static int i2o_cfg_passthru32(struct fil
           "%s: Could not copy message context FROM user\n",
           c->name);
    rcode = -EFAULT;
-   goto sg_list_cleanup;
   }
   if (copy_to_user(user_reply, reply, reply_size)) {
    printk(KERN_WARNING
@@ -731,12 +735,14 @@ static int i2o_cfg_passthru32(struct fil
    rcode = -EFAULT;
   }
  }
-
  for (i = 0; i < sg_index; i++)
   i2o_dma_free(&c->pdev->dev, &sg_list[i]);

-      cleanup:
+cleanup:
  kfree(reply);
+ if (msg)
+out:
+  i2o_msg_nop(c, msg);
  return rcode;
 }

@@ -793,8 +799,6 @@ static int i2o_cfg_passthru(unsigned lon
   return -ENXIO;
  }

- msg = i2o_msg_get_wait(c, I2O_TIMEOUT_MESSAGE_GET);
-
  sb = c->status_block.virt;
```

```
  if (get_user(size, &user_msg[0]))
@@ -810,12 +814,17 @@ static int i2o_cfg_passthru(unsigned lon

  size <<= 2;  // Convert to bytes

+ msg = i2o_msg_get_wait(c, I2O_TIMEOUT_MESSAGE_GET);
+ if (IS_ERR(msg))
+  return PTR_ERR(msg);
+
+ rcode = -EFAULT;
  /* Copy in the user's I2O command */
  if (copy_from_user(msg, user_msg, size))
-  return -EFAULT;
+  goto out;

  if (get_user(reply_size, &user_reply[0]) < 0)
-  return -EFAULT;
+  goto out;

  reply_size >>= 16;
  reply_size <<= 2;
@@ -824,7 +833,8 @@ static int i2o_cfg_passthru(unsigned lon
  if (!reply) {
   printk(KERN_WARNING "%s: Could not allocate reply buffer\n",
          c->name);
-  return -ENOMEM;
+  rcode = -ENOMEM;
+  goto out;
  }

  sg_offset = (msg->u.head[0] >> 4) & 0x0f;
@@ -891,13 +901,14 @@ static int i2o_cfg_passthru(unsigned lon
  }

  rcode = i2o_msg_post_wait(c, msg, 60);
+ msg = NULL;
  if (rcode) {
   reply[4] = ((u32) rcode) << 24;
   goto sg_list_cleanup;
  }

  if (sg_offset) {
-  u32 msg[128];
+  u32 rmsg[128];
   /* Copy back the Scatter Gather buffers back to user space */
   u32 j;
   // TODO 64bit fix
```

```
@@ -905,7 +916,7 @@ static int i2o_cfg_passthru(unsigned lon
   int sg_size;

   // re-acquire the original message to handle correctly the sg copy operation
-  memset(&msg, 0, I2O_OUTBOUND_MSG_FRAME_SIZE * 4);
+  memset(&rmsg, 0, I2O_OUTBOUND_MSG_FRAME_SIZE * 4);
   // get user msg size in u32s
   if (get_user(size, &user_msg[0])) {
    rcode = -EFAULT;
@@ -914,7 +925,7 @@ static int i2o_cfg_passthru(unsigned lon
   size = size >> 16;
   size *= 4;
   /* Copy in the user's I2O command */
-  if (copy_from_user(msg, user_msg, size)) {
+  if (copy_from_user(rmsg, user_msg, size)) {
    rcode = -EFAULT;
    goto sg_list_cleanup;
   }
@@ -922,7 +933,7 @@ static int i2o_cfg_passthru(unsigned lon
       (size - sg_offset * 4) / sizeof(struct sg_simple_element);

   // TODO 64bit fix
-  sg = (struct sg_simple_element *)(msg + sg_offset);
+  sg = (struct sg_simple_element *)(rmsg + sg_offset);
   for (j = 0; j < sg_count; j++) {
    /* Copy out the SG list to user's buffer if necessary */
    if (!
@@ -944,7 +955,7 @@ static int i2o_cfg_passthru(unsigned lon
   }
  }

-      sg_list_cleanup:
+sg_list_cleanup:
  /* Copy back the reply to user space */
  if (reply_size) {
   // we wrote our own values for context - now restore the user supplied ones
@@ -964,8 +975,11 @@ static int i2o_cfg_passthru(unsigned lon
  for (i = 0; i < sg_index; i++)
   kfree(sg_list[i]);

-      cleanup:
+cleanup:
  kfree(reply);
+ if (msg)
+out:
+  i2o_msg_nop(c, msg);
  return rcode;
 }
```

#endif