Subject: [PATCH] single_open/seq_release leak diagnostics
Posted by Alexey Dobriyan on Tue, 31 Jul 2007 12:05:26 GMT
View Forum Message <> Reply to Message

"[PATCH] Fix leaks on /proc/{*/sched,sched_debug,timer_list,timer_stats}" and
"[PATCH] Fix leak on /proc/lockdep_stats" fixed 5 leaks which happen if one
uses single_open() as .open function and seq_release() as .release function.

Let's add small amount of runtime checking.

Sample output is:

```
 memory leak: 'timer_list'
 WARNING: at fs/seq_file.c:289 seq_release()
  [<c015e311>] seq_release+0x53/0x68
  [<c0171bd8>] proc_reg_release+0x63/0x74
  [<c0149877>] __fput+0x28/0xd3
  [<c0147681>] filp_close+0x48/0x4f
  [<c014876a>] sys_close+0x74/0xbe
  [<c010248e>] sysenter_past_esp+0x5f/0x85
  =======================
```

Signed-off-by: Alexey Dobriyan <adobriyan@sw.ru>

```
 fs/seq_file.c            |   25 +++++++++++++++++++++-----
 include/linux/seq_file.h |    1 +
 2 files changed, 21 insertions(+), 5 deletions(-)

--- a/fs/seq_file.c
+++ b/fs/seq_file.c
@@ -281,6 +281,13 @@ EXPORT_SYMBOL(seq_lseek);
 int seq_release(struct inode *inode, struct file *file)
 {
  struct seq_file *m = (struct seq_file *)file->private_data;
+
+ if (m->seq_ops_allocated) {
+  struct dentry *dentry = file->f_dentry;
+  printk("memory leak: '%.*s'\n",
+   dentry->d_name.len, dentry->d_name.name);
+  WARN_ON(1);
+ }
  kfree(m->buf);
  kfree(m);
  return 0;
@@ -401,9 +408,12 @@ int single_open(struct file *file, int (*show)(struct seq_file *, void *),
   op->stop = single_stop;
   op->show = show;
```

```
   res = seq_open(file, op);
-  if (!res)
-   ((struct seq_file *)file->private_data)->private = data;
-  else
+  if (!res) {
+   struct seq_file *seq = file->private_data;
+
+   seq->private = data;
+   seq->seq_ops_allocated = 1;
+  } else
    kfree(op);
  }
  return res;
@@ -412,8 +422,13 @@ EXPORT_SYMBOL(single_open);

 int single_release(struct inode *inode, struct file *file)
 {
- const struct seq_operations *op = ((struct seq_file *)file->private_data)->op;
- int res = seq_release(inode, file);
+ struct seq_file *seq = file->private_data;
+ const struct seq_operations *op = seq->op;
+ int res;
+
+ /* All roads lead to seq_release(), so... */
+ seq->seq_ops_allocated = 0;
+ res = seq_release(inode, file);
  kfree(op);
  return res;
 }
--- a/include/linux/seq_file.h
+++ b/include/linux/seq_file.h
@@ -22,6 +22,7 @@ struct seq_file {
  struct mutex lock;
  const struct seq_operations *op;
  void *private;
+ unsigned int seq_ops_allocated:1;
 };

 struct seq_operations {
```