Subject: Racy /proc creations interfaces
Posted by adobriyan on Wed, 27 Dec 2006 13:36:04 GMT
View Forum Message <> Reply to Message

1. Not setting ->owner for your favourite /proc file is oopsable.

```
 $ while true; do cat /proc/fs/xfs/stat 1>/dev/null 2>/dev/null; done
 # while true; do modprobe xfs; rmmod xfs; done
```

```
BUG: unable to handle kernel paging request at virtual address fc281049
 printing eip:
fc281049
*pde = 37ca7067
Oops: 0000 [#1]
Modules linked in: af_packet ohci_hcd e1000 ehci_hcd uhci_hcd usbcore
CPU:    0
EIP:    0060:[<fc281049>]    Not tainted VLI
EFLAGS: 00010282   (2.6.19 #2)
EIP is at 0xfc281049
eax: f0705000   ebx: 00000000   ecx: 00000105   edx: f1e5ff68
esi: f0705000   edi: fc281049   ebp: 00000c00   esp: f1e5ff4c
ds: 007b   es: 007b   ss: 0068
Process cat (pid: 32588, ti=f1e5e000 task=f4377030 task.ti=f1e5e000)
Stack: c0177d9c 00000c00 f1e5ff6c 00000000 00001000 08051000 f7ccc240 00000000
       00000000 edbe82c0 00001000 08051000 f1e5ffa4 c0149f05 f1e5ffa4 c0177be5
       edbe82c0 fffffff7 08051000 f1e5e000 c014a26d f1e5ffa4 00000105 00000000
Call Trace:
 [<c0177d9c>] proc_file_read+0x1b7/0x262
 [<c0149f05>] vfs_read+0x85/0xf3
 [<c0177be5>] proc_file_read+0x0/0x262
 [<c014a26d>] sys_read+0x41/0x6a
 [<c0102be1>] sysenter_past_esp+0x56/0x79
 =======================
Code:  Bad EIP value.
EIP: [<fc281049>] 0xfc281049 SS:ESP 0068:f1e5ff4c
```

2. After adding ->owner for your favourite /proc file it's still looks racy

```
 pde = create_proc_entry("foo", 0644, NULL);
 /* so what? pde->owner is NULL here */
 if (pde)
  pde->owner = THIS_MODULE;
```

3. It's more: setting both ->read_proc and ->data after proc entry creation
   should be oopsable too, if ->read_proc operates on data in a non-trivial
   way (or ->write_proc).

sound/oss/trident.c:

```
 4446    res = create_proc_entry("ALi5451", 0, NULL);
 4447    if (res) {
 4448     res->write_proc = ali_write_proc;
      /* ->data is NULL here. */
 4449     res->data = card;
 4450    }
```

ali_write_proc() does spin_lock_irqsave on private card lock which is believed to be valid.

4. Proposed semantics: PDE should be in maximally working state before gluing into main /proc tree (proc_register()).

```
   struct proc_entry_raw foo_pe_raw = {
 .owner = THIS_MODULE,
 .name = "foo",
 .mode = 0644,
 .read_proc = foo_read_proc,
 .data = foo_data,
 .parent = foo_parent,
};

pde = create_proc_entry(&foo_pe_raw);
if (!pde)
 return -ENOMEM;
```

where "struct proc_entry_raw" is cut down version of "struct proc_dir_entry" where new create_proc_entry() would do

```
   proc_create() or kmalloc()
fill in fields from passed "struct proc_entry_raw"
proc_register()
```

Comments?

Below is 0-order approximation to what I mean (not tested at all):

```
--- a/fs/afs/proc.c
+++ b/fs/afs/proc.c
@@ -146,17 +146,23 @@ int afs_proc_init(void)
   goto error;
  proc_afs->owner = THIS_MODULE;

- p = create_proc_entry("cells", 0, proc_afs);
+ p = __create_proc_entry(&(struct proc_entry_raw){
+   .name = "cells",
+   .parent = proc_afs,
+   .proc_fops = &afs_proc_cells_fops,
```

```
+   .owner = THIS_MODULE,
+  });
  if (!p)
   goto error_proc;
- p->proc_fops = &afs_proc_cells_fops;
- p->owner = THIS_MODULE;

- p = create_proc_entry("rootcell", 0, proc_afs);
+ p = __create_proc_entry(&(struct proc_entry_raw){
+   .name = "rootcell",
+   .parent = proc_afs,
+   .proc_fops = &afs_proc_rootcell_fops,
+   .owner = THIS_MODULE,
+  });
  if (!p)
   goto error_cells;
- p->proc_fops = &afs_proc_rootcell_fops;
- p->owner = THIS_MODULE;

  _leave(" = 0");
  return 0;
@@ -429,26 +435,35 @@ int afs_proc_cell_setup(struct afs_cell
  if (!cell->proc_dir)
   return -ENOMEM;

- p = create_proc_entry("servers", 0, cell->proc_dir);
+ p = __create_proc_entry(&(struct proc_entry_raw){
+   .name = "servers",
+   .parent = cell->proc_dir,
+   .proc_fops = &afs_proc_cell_servers_fops,
+   .owner = THIS_MODULE,
+   .data = cell,
+  });
  if (!p)
   goto error_proc;
- p->proc_fops = &afs_proc_cell_servers_fops;
- p->owner = THIS_MODULE;
- p->data = cell;

- p = create_proc_entry("vlservers", 0, cell->proc_dir);
+ p = __create_proc_entry(&(struct proc_entry_raw){
+   .name = "vlservers",
+   .parent = cell->proc_dir,
+   .proc_fops = &afs_proc_cell_vlservers_fops,
+   .owner = THIS_MODULE,
+   .data = cell,
+  });
  if (!p)
```

```
   goto error_servers;
- p->proc_fops = &afs_proc_cell_vlservers_fops;
- p->owner = THIS_MODULE;
- p->data = cell;

- p = create_proc_entry("volumes", 0, cell->proc_dir);
+ p = __create_proc_entry(&(struct proc_entry_raw){
+    .name = "volumes",
+    .parent = cell->proc_dir,
+    .proc_fops = &afs_proc_cell_volumes_fops,
+    .owner = THIS_MODULE,
+    .data = cell,
+  });
  if (!p)
   goto error_vlservers;
- p->proc_fops = &afs_proc_cell_volumes_fops;
- p->owner = THIS_MODULE;
- p->data = cell;

  _leave(" = 0");
  return 0;
diff --git a/fs/jbd/journal.c b/fs/jbd/journal.c
index 10fff94..faf0470 100644
--- a/fs/jbd/journal.c
+++ b/fs/jbd/journal.c
@@ -1975,12 +1975,12 @@ #define JBD_PROC_NAME "sys/fs/jbd-debug"

 static void __init create_jbd_proc_entry(void)
 {
- proc_jbd_debug = create_proc_entry(JBD_PROC_NAME, 0644, NULL);
- if (proc_jbd_debug) {
-  /* Why is this so hard? */
-  proc_jbd_debug->read_proc = read_jbd_debug;
-  proc_jbd_debug->write_proc = write_jbd_debug;
- }
+ __create_proc_entry(&(struct proc_entry_raw){
+    .name = JBD_PROC_NAME,
+    .mode = 0644,
+    .read_proc = read_jbd_debug,
+    .write_proc = write_jbd_debug,
+  });
 }

 static void __exit remove_jbd_proc_entry(void)
diff --git a/fs/jbd2/journal.c b/fs/jbd2/journal.c
index 44fc32b..32de473 100644
--- a/fs/jbd2/journal.c
+++ b/fs/jbd2/journal.c
```

```
@@ -1986,12 +1986,12 @@ #define JBD_PROC_NAME "sys/fs/jbd2-debug

 static void __init create_jbd_proc_entry(void)
 {
- proc_jbd_debug = create_proc_entry(JBD_PROC_NAME, 0644, NULL);
- if (proc_jbd_debug) {
-  /* Why is this so hard? */
-  proc_jbd_debug->read_proc = read_jbd_debug;
-  proc_jbd_debug->write_proc = write_jbd_debug;
- }
+ __create_proc_entry(&(struct proc_entry_raw){
+   .name = JBD_PROC_NAME,
+   .mode = 0644,
+   .read_proc = read_jbd_debug,
+   .write_proc = write_jbd_debug,
+  });
 }

 static void __exit jbd2_remove_jbd_proc_entry(void)
diff --git a/fs/jffs/jffs_proc.c b/fs/jffs/jffs_proc.c
index 9bdd99a..ec00942 100644
--- a/fs/jffs/jffs_proc.c
+++ b/fs/jffs/jffs_proc.c
@@ -85,12 +85,12 @@ int jffs_register_jffs_proc_dir(int mtd,
  if (!part_root)
   goto out1;

- /* Create entry for 'info' file */
- part_info = create_proc_entry ("info", 0, part_root);
- if (!part_info)
+ if (!__create_proc_entry(&(struct proc_entry_raw){
+   .name = "info",
+   .parent = part_root,
+   .read_proc = jffs_proc_info_read,
+   .data = (void *)c,}))
   goto out2;
- part_info->read_proc = jffs_proc_info_read;
- part_info->data = (void *) c;

  /* Create entry for 'layout' file */
  part_layout = create_proc_entry ("layout", 0, part_root);
diff --git a/fs/nfs/client.c b/fs/nfs/client.c
index 23ab145..44ca278 100644
--- a/fs/nfs/client.c
+++ b/fs/nfs/client.c
@@ -1406,20 +1406,27 @@ int __init nfs_fs_proc_init(void)
  proc_fs_nfs->owner = THIS_MODULE;
```

```
  /* a file of servers with which we're dealing */
- p = create_proc_entry("servers", S_IFREG|S_IRUGO, proc_fs_nfs);
+ p = __create_proc_entry(&(struct proc_entry_raw){
+    .name = "servers",
+    .mode = S_IFREG|S_IRUGO,
+    .parent = proc_fs_nfs,
+    .proc_fops = &nfs_server_list_fops,
+    .owner = THIS_MODULE,
+  });
  if (!p)
   goto error_1;

- p->proc_fops = &nfs_server_list_fops;
- p->owner = THIS_MODULE;
-
  /* a file of volumes that we have mounted */
- p = create_proc_entry("volumes", S_IFREG|S_IRUGO, proc_fs_nfs);
+ p = __create_proc_entry(&(struct proc_entry_raw){
+    .name = "volumes",
+    .mode = S_IFREG|S_IRUGO,
+    .parent = proc_fs_nfs,
+    .proc_fops = &nfs_volume_list_fops,
+    .owner = THIS_MODULE,
+  });
  if (!p)
   goto error_2;

- p->proc_fops = &nfs_volume_list_fops;
- p->owner = THIS_MODULE;
  return 0;

 error_2:
diff --git a/fs/proc/generic.c b/fs/proc/generic.c
index 853cb87..0bce166 100644
--- a/fs/proc/generic.c
+++ b/fs/proc/generic.c
@@ -657,6 +657,47 @@ struct proc_dir_entry *proc_mkdir(const
  return proc_mkdir_mode(name, S_IRUGO | S_IXUGO, parent);
 }

+struct proc_dir_entry *__create_proc_entry(struct proc_entry_raw *raw)
+{
+ struct proc_dir_entry *ent;
+ mode_t mode = raw->mode;
+ nlink_t nlink;
+
+ if (S_ISDIR(mode)) {
+  if ((mode & S_IALLUGO) == 0)
```

```
+   mode |= S_IRUGO | S_IXUGO;
+  nlink = 2;
+ } else {
+  if ((mode & S_IFMT) == 0)
+   mode |= S_IFREG;
+  if ((mode & S_IALLUGO) == 0)
+   mode |= S_IRUGO;
+  nlink = 1;
+ }
+
+ ent = proc_create(&raw->parent, raw->name, mode, nlink);
+ if (ent) {
+  if (S_ISDIR(mode)) {
+   ent->proc_fops = &proc_dir_operations;
+   ent->proc_iops = &proc_dir_inode_operations;
+  }
+
+  ent->size = raw->size;
+  if (raw->proc_fops)
+   ent->proc_fops = raw->proc_fops;
+  ent->data = raw->data;
+  ent->read_proc = raw->read_proc;
+  ent->write_proc = raw->write_proc;
+  ent->owner = raw->owner;
+
+  if (proc_register(raw->parent, ent) < 0) {
+   kfree(ent);
+   ent = NULL;
+  }
+ }
+ return ent;
+}
+
 struct proc_dir_entry *create_proc_entry(const char *name, mode_t mode,
       struct proc_dir_entry *parent)
 {
diff --git a/fs/proc/root.c b/fs/proc/root.c
index 64d242b..f7d3fda 100644
--- a/fs/proc/root.c
+++ b/fs/proc/root.c
@@ -166,6 +166,7 @@ struct proc_dir_entry proc_root = {
 EXPORT_SYMBOL(proc_symlink);
 EXPORT_SYMBOL(proc_mkdir);
 EXPORT_SYMBOL(create_proc_entry);
+EXPORT_SYMBOL(__create_proc_entry);
 EXPORT_SYMBOL(remove_proc_entry);
 EXPORT_SYMBOL(proc_root);
 EXPORT_SYMBOL(proc_root_fs);
```

```
diff --git a/include/linux/proc_fs.h b/include/linux/proc_fs.h
index 87dec8f..0523180 100644
--- a/include/linux/proc_fs.h
+++ b/include/linux/proc_fs.h
@@ -68,6 +68,18 @@ struct proc_dir_entry {
  void *set;
 };

+struct proc_entry_raw {
+ const char *name;
+ mode_t mode;
+ loff_t size;
+ const struct file_operations *proc_fops;
+ struct module *owner;
+ struct proc_dir_entry *parent;
+ void *data;
+ read_proc_t *read_proc;
+ write_proc_t *write_proc;
+};
+
 struct kcore_list {
  struct kcore_list *next;
  unsigned long addr;
@@ -107,6 +119,7 @@ char *task_mem(struct mm_struct *, char

 extern struct proc_dir_entry *create_proc_entry(const char *name, mode_t mode,
       struct proc_dir_entry *parent);
+struct proc_dir_entry *__create_proc_entry(struct proc_entry_raw *raw);
 extern void remove_proc_entry(const char *name, struct proc_dir_entry *parent);

 extern struct vfsmount *proc_mnt;
@@ -214,6 +227,11 @@ static inline void proc_flush_task(struc
 static inline struct proc_dir_entry *create_proc_entry(const char *name,
  mode_t mode, struct proc_dir_entry *parent) { return NULL; }

+static inline struct proc_dir_entry *__create_proc_entry(struct proc_entry_raw *raw)
+{
+ return NULL;
+}
+
 #define remove_proc_entry(name, parent) do {} while (0)

 static inline struct proc_dir_entry *proc_symlink(const char *name,
diff --git a/kernel/configs.c b/kernel/configs.c
index 8fa1fb2..1694608 100644
--- a/kernel/configs.c
+++ b/kernel/configs.c
@@ -85,18 +85,16 @@ static const struct file_operations ikco
```

```
 static int __init ikconfig_init(void)
 {
- struct proc_dir_entry *entry;
+ struct proc_entry_raw pe_raw = {
+  .name  = "config.gz",
+  .mode  = S_IFREG | S_IRUGO,
+  .parent  = &proc_root,
+  .proc_fops = &ikconfig_file_ops,
+  .size  = kernel_config_data_size,
+ };

  /* create the current config file */
- entry = create_proc_entry("config.gz", S_IFREG | S_IRUGO,
-       &proc_root);
- if (!entry)
-  return -ENOMEM;
-
- entry->proc_fops = &ikconfig_file_ops;
- entry->size = kernel_config_data_size;
-
- return 0;
+ return __create_proc_entry(&pe_raw) ? 0 : -ENOMEM;
 }

 /*****************************************************/
diff --git a/kernel/dma.c b/kernel/dma.c
index 937b13c..51726c0 100644
--- a/kernel/dma.c
+++ b/kernel/dma.c
@@ -149,12 +149,11 @@ static const struct file_operations proc

 static int __init proc_dma_init(void)
 {
- struct proc_dir_entry *e;
-
- e = create_proc_entry("dma", 0, NULL);
- if (e)
-  e->proc_fops = &proc_dma_operations;
-
+ struct proc_entry_raw pe_raw = {
+  .name    = "dma",
+  .proc_fops = &proc_dma_operations,
+ };
+ __create_proc_entry(&pe_raw);
  return 0;
 }
```

```
diff --git a/kernel/irq/proc.c b/kernel/irq/proc.c
index 61f5c71..aecabf5 100644
--- a/kernel/irq/proc.c
+++ b/kernel/irq/proc.c
@@ -130,17 +130,17 @@ void register_irq_proc(unsigned int irq)

 #ifdef CONFIG_SMP
  {
-  struct proc_dir_entry *entry;
+  struct proc_entry_raw pe_raw = {
+   .name = "smp_affinity",
+   .mode = 0600,
+   .parent = irq_desc[irq].dir,
+   .data = (void *)(long)irq,
+   .read_proc = irq_affinity_read_proc,
+   .write_proc = irq_affinity_write_proc,
+  };

   /* create /proc/irq/<irq>/smp_affinity */
-  entry = create_proc_entry("smp_affinity", 0600, irq_desc[irq].dir);
-
-  if (entry) {
-   entry->nlink = 1;
-   entry->data = (void *)(long)irq;
-   entry->read_proc = irq_affinity_read_proc;
-   entry->write_proc = irq_affinity_write_proc;
-  }
+  __create_proc_entry(&pe_raw);
  }
 #endif
 }
diff --git a/kernel/kallsyms.c b/kernel/kallsyms.c
index 6f294ff..3f26794 100644
--- a/kernel/kallsyms.c
+++ b/kernel/kallsyms.c
@@ -442,11 +442,12 @@ static const struct file_operations kall

 static int __init kallsyms_init(void)
 {
- struct proc_dir_entry *entry;
-
- entry = create_proc_entry("kallsyms", 0444, NULL);
- if (entry)
-  entry->proc_fops = &kallsyms_operations;
+ struct proc_entry_raw pe_raw = {
+  .name    = "kallsyms",
+  .mode    = 0444,
+  .proc_fops = &kallsyms_operations,
```

```
+ };
+ __create_proc_entry(&pe_raw);
  return 0;
 }
 __initcall(kallsyms_init);
diff --git a/kernel/lockdep_proc.c b/kernel/lockdep_proc.c
index b554b40..3d297d4 100644
--- a/kernel/lockdep_proc.c
+++ b/kernel/lockdep_proc.c
@@ -328,16 +328,19 @@ static const struct file_operations proc

 static int __init lockdep_proc_init(void)
 {
- struct proc_dir_entry *entry;
-
- entry = create_proc_entry("lockdep", S_IRUSR, NULL);
- if (entry)
-  entry->proc_fops = &proc_lockdep_operations;
-
- entry = create_proc_entry("lockdep_stats", S_IRUSR, NULL);
- if (entry)
-  entry->proc_fops = &proc_lockdep_stats_operations;
-
+ struct proc_entry_raw pe_raw_lockdep = {
+  .name = "lockdep",
+  .mode = S_IRUSR,
+  .proc_fops = &proc_lockdep_operations,
+ };
+ struct proc_entry_raw pe_raw_lockdep_stats = {
+  .name = "lockdep_stats",
+  .mode = S_IRUSR,
+  .proc_fops = &proc_lockdep_stats_operations,
+ };
+
+ __create_proc_entry(&pe_raw_lockdep);
+ __create_proc_entry(&pe_raw_lockdep_stats);
  return 0;
 }

diff --git a/kernel/profile.c b/kernel/profile.c
index fb5e03d..19b73a0 100644
--- a/kernel/profile.c
+++ b/kernel/profile.c
@@ -429,15 +429,16 @@ static int prof_cpu_mask_write_proc (str

 void create_prof_cpu_mask(struct proc_dir_entry *root_irq_dir)
 {
- struct proc_dir_entry *entry;
```

```
-
+ struct proc_entry_raw pe_raw = {
+  .name = "prof_cpu_mask",
+  .mode = 0600,
+  .parent = root_irq_dir,
+  .data = &prof_cpu_mask,
+  .read_proc = prof_cpu_mask_read_proc,
+  .write_proc = prof_cpu_mask_write_proc,
+ };
   /* create /proc/irq/prof_cpu_mask */
- if (!(entry = create_proc_entry("prof_cpu_mask", 0600, root_irq_dir)))
-  return;
- entry->nlink = 1;
- entry->data = (void *)&prof_cpu_mask;
- entry->read_proc = prof_cpu_mask_read_proc;
- entry->write_proc = prof_cpu_mask_write_proc;
+ __create_proc_entry(&pe_raw);
 }

 /*
@@ -561,16 +562,19 @@ #endif

 static int __init create_proc_profile(void)
 {
- struct proc_dir_entry *entry;
+ struct proc_entry_raw pe_raw = {
+  .name    = "profile",
+  .mode    = S_IWUSR | S_IRUGO,
+  .proc_fops = &proc_profile_operations,
+  .size    = (1 + prof_len) * sizeof(atomic_t),
+ };

   if (!prof_on)
    return 0;
   if (create_hash_tables())
    return -1;
- if (!(entry = create_proc_entry("profile", S_IWUSR | S_IRUGO, NULL)))
+ if (!__create_proc_entry(&pe_raw))
    return 0;
- entry->proc_fops = &proc_profile_operations;
- entry->size = (1+prof_len) * sizeof(atomic_t);
   hotcpu_notifier(profile_cpu_callback, 0);
   return 0;
 }
diff --git a/kernel/resource.c b/kernel/resource.c
index 7b9a497..7caf831 100644
--- a/kernel/resource.c
+++ b/kernel/resource.c
```

```
@@ -131,14 +131,17 @@ static const struct file_operations proc

 static int __init ioresources_init(void)
 {
- struct proc_dir_entry *entry;
-
- entry = create_proc_entry("ioports", 0, NULL);
- if (entry)
-  entry->proc_fops = &proc_ioports_operations;
- entry = create_proc_entry("iomem", 0, NULL);
- if (entry)
-  entry->proc_fops = &proc_iomem_operations;
+ struct proc_entry_raw pe_raw_ioports = {
+  .name = "ioports",
+  .proc_fops = &proc_ioports_operations,
+ };
+ struct proc_entry_raw pe_raw_iomem = {
+  .name = "iomem",
+  .proc_fops = &proc_iomem_operations,
+ };
+
+ __create_proc_entry(&pe_raw_ioports);
+ __create_proc_entry(&pe_raw_iomem);
  return 0;
 }
 __initcall(ioresources_init);
```