Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by Cedric Le Goater on Fri, 16 Mar 2007 14:44:31 GMT
View Forum Message <> Reply to Message

> How about you send over the autofs4 bit and I'll have a look (the autofs
> patch looked fine). That would save me a bit of time and if there are
> any changes needed I can send an updated patch for you guys to review. I
> don't think autofs4 uses pids differently, in principle, than autofs so
> it "should" be straight forward.

Here's the latest.

thanks,

C.

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

Subject: Replace pid_t in autofs4 with struct pid reference.

Make autofs4 container-friendly by caching struct pid reference rather
than pid_t and using pid_nr() to retreive a task's pid_t.

ChangeLog:
 - Fix Eric Biederman's comments - Use find_get_pid() to hold a
   reference to oz_pgrp and release while unmounting; separate out
   changes to autofs and autofs4.
 - Also rollback my earlier change to autofs_wait_queue (pid and tgid
   in the wait queue are just used to write to a userspace daemon's
   pipe).
       - Fix Cedric's comments: retain old prototype of parse_options()
         and move necessary change to its caller.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Cc: Cedric Le Goater <clg@fr.ibm.com>
Cc: Dave Hansen <haveblue@us.ibm.com>
Cc: Serge Hallyn <serue@us.ibm.com>
Cc: Eric Biederman <ebiederm@xmission.com>
Cc: containers@lists.osdl.org

---
 fs/autofs4/autofs_i.h | 6 +++---
 fs/autofs4/inode.c | 21 +++++++++++++++++++-----
 fs/autofs4/root.c | 3 ++-
 fs/autofs4/waitq.c | 4 ++--
 4 files changed, 23 insertions(+), 11 deletions(-)

Index: 2.6.20/fs/autofs4/autofs_i.h

```
===================================================================
--- 2.6.20.orig/fs/autofs4/autofs_i.h
+++ 2.6.20/fs/autofs4/autofs_i.h
@@ -35,7 +35,7 @@
 /* #define DEBUG */

 #ifdef DEBUG
-#define DPRINTK(fmt,args...) do { printk(KERN_DEBUG "pid %d: %s: " fmt "\n" , current->pid ,
__FUNCTION__ , ##args); } while(0)
+#define DPRINTK(fmt,args...) do { printk(KERN_DEBUG "pid %d: %s: " fmt "\n" ,
pid_nr(task_pid(current)), __FUNCTION__ , ##args); } while(0)
 #else
 #define DPRINTK(fmt,args...) do {} while(0)
 #endif
@@ -96,7 +96,7 @@ struct autofs_sb_info {
  u32 magic;
  int pipefd;
  struct file *pipe;
- pid_t oz_pgrp;
+ struct pid *oz_pgrp;
  int catatonic;
  int version;
  int sub_version;
@@ -127,7 +127,7 @@ static inline struct autofs_info *autofs
    filesystem without "magic".) */

 static inline int autofs4_oz_mode(struct autofs_sb_info *sbi) {
- return sbi->catatonic || process_group(current) == sbi->oz_pgrp;
+ return sbi->catatonic || task_pgrp(current) == sbi->oz_pgrp;
 }

 /* Does a dentry have some pending activity? */
Index: 2.6.20/fs/autofs4/inode.c
===================================================================
--- 2.6.20.orig/fs/autofs4/inode.c
+++ 2.6.20/fs/autofs4/inode.c
@@ -163,6 +163,8 @@ void autofs4_kill_sb(struct super_block
  if ( !sbi->catatonic )
   autofs4_catatonic_mode(sbi); /* Free wait queues, close pipe */

+ put_pid(sbi->oz_pgrp);
+
  /* Clean up and release dangling references */
  autofs4_force_release(sbi);

@@ -181,7 +183,7 @@ static int autofs4_show_options(struct s
   return 0;
```

```
  seq_printf(m, ",fd=%d", sbi->pipefd);
- seq_printf(m, ",pgrp=%d", sbi->oz_pgrp);
+ seq_printf(m, ",pgrp=%d", pid_nr(sbi->oz_pgrp));
  seq_printf(m, ",timeout=%lu", sbi->exp_timeout/HZ);
  seq_printf(m, ",minproto=%d", sbi->min_proto);
  seq_printf(m, ",maxproto=%d", sbi->max_proto);
@@ -311,6 +313,7 @@ int autofs4_fill_super(struct super_bloc
  int pipefd;
  struct autofs_sb_info *sbi;
  struct autofs_info *ino;
+ pid_t pgid;

  sbi = kmalloc(sizeof(*sbi), GFP_KERNEL);
  if (!sbi)
@@ -325,7 +328,6 @@ int autofs4_fill_super(struct super_bloc
  sbi->pipe = NULL;
  sbi->catatonic = 1;
  sbi->exp_timeout = 0;
- sbi->oz_pgrp = process_group(current);
  sbi->sb = s;
  sbi->version = 0;
  sbi->sub_version = 0;
@@ -361,7 +363,7 @@ int autofs4_fill_super(struct super_bloc

  /* Can this call block? */
  if (parse_options(data, &pipefd, &root_inode->i_uid, &root_inode->i_gid,
-    &sbi->oz_pgrp, &sbi->type, &sbi->min_proto,
+    &pgid, &sbi->type, &sbi->min_proto,
     &sbi->max_proto)) {
   printk("autofs: called with bogus options\n");
   goto fail_dput;
@@ -389,12 +391,19 @@ int autofs4_fill_super(struct super_bloc
   sbi->version = sbi->max_proto;
  sbi->sub_version = AUTOFS_PROTO_SUBVERSION;

- DPRINTK("pipe fd = %d, pgrp = %u", pipefd, sbi->oz_pgrp);
+ DPRINTK("pipe fd = %d, pgrp = %u", pipefd, pgid);
+
+ sbi->oz_pgrp = find_get_pid(pgid);
+ if (!sbi->oz_pgrp) {
+  printk("autofs: could not find process group %d\n", pgid);
+  goto fail_dput;
+ }
+
  pipe = fget(pipefd);

  if (!pipe) {
   printk("autofs: could not open pipe file descriptor\n");
```

```
-  goto fail_dput;
+  goto fail_put_pid;
  }
  if (!pipe->f_op || !pipe->f_op->write)
   goto fail_fput;
@@ -415,6 +424,8 @@ fail_fput:
  printk("autofs: pipe file descriptor does not contain proper ops\n");
  fput(pipe);
  /* fall through */
+fail_put_pid:
+ put_pid(sbi->oz_pgrp);
 fail_dput:
  dput(root);
  goto fail_free;
Index: 2.6.20/fs/autofs4/root.c
===================================================================
--- 2.6.20.orig/fs/autofs4/root.c
+++ 2.6.20/fs/autofs4/root.c
@@ -495,7 +495,8 @@ static struct dentry *autofs4_lookup(str
  oz_mode = autofs4_oz_mode(sbi);

  DPRINTK("pid = %u, pgrp = %u, catatonic = %d, oz_mode = %d",
-   current->pid, process_group(current), sbi->catatonic, oz_mode);
+   pid_nr(task_pid(current)), process_group(current),
+   sbi->catatonic, oz_mode);

  /*
   * Mark the dentry incomplete, but add it. This is needed so
Index: 2.6.20/fs/autofs4/waitq.c
===================================================================
--- 2.6.20.orig/fs/autofs4/waitq.c
+++ 2.6.20/fs/autofs4/waitq.c
@@ -292,8 +292,8 @@ int autofs4_wait(struct autofs_sb_info *
   wq->ino = autofs4_get_ino(sbi);
   wq->uid = current->uid;
   wq->gid = current->gid;
-  wq->pid = current->pid;
-  wq->tgid = current->tgid;
+  wq->pid = pid_nr(task_pid(current));
+  wq->tgid = pid_nr(task_tgid(current));
   wq->status = -EINTR; /* Status return if interrupted */
   atomic_set(&wq->wait_ctr, 2);
   mutex_unlock(&sbi->wq_mutex);
```