
Subject: [PATCH 3/3] Replace pid_t in autofs4 with struct pid reference.
Posted by [Sukadev Bhattiprolu](#) on Thu, 01 Mar 2007 04:01:27 GMT
[View Forum Message](#) <> [Reply to Message](#)

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [PATCH 3/3] 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 retrieve 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: lx26-20-mm2c/fs/autofs4/autofs_i.h

```
=====
--- lx26-20-mm2c.orig/fs/autofs4/autofs_i.h 2007-02-28 14:48:35.000000000 -0800
+++ lx26-20-mm2c/fs/autofs4/autofs_i.h 2007-02-28 15:47:21.000000000 -0800
@@ -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
@@ -98,7 +98,7 @@ 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;
@@ -131,7 +131,7 @@ 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: lx26-20-mm2c/fs/autofs4/inode.c
=====
--- lx26-20-mm2c.orig/fs/autofs4/inode.c 2007-02-28 15:47:09.000000000 -0800
+++ lx26-20-mm2c/fs/autofs4/inode.c 2007-02-28 15:47:21.000000000 -0800
@@ -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);

@@ -182,7 +184,7 @@ 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);
@@ -312,6 +314,7 @@ 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)
@@ -326,7 +329,6 @@ 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;
@@ -364,7 +366,7 @@ 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;
@@ -392,12 +394,19 @@ 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;
@@ -418,6 +427,8 @@ 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: lx26-20-mm2c/fs/autofs4/root.c

--- lx26-20-mm2c.orig/fs/autofs4/root.c 2007-02-28 15:47:09.000000000 -0800

+++ lx26-20-mm2c/fs/autofs4/root.c 2007-02-28 15:47:21.000000000 -0800

@@ -583,7 +583,8 @@ 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);

unhashed = autofs4_lookup_unhashed(sbi, dentry->d_parent, &dentry->d_name);

if (!unhashed) {

Index: lx26-20-mm2c/fs/autofs4/waitq.c

--- lx26-20-mm2c.orig/fs/autofs4/waitq.c 2007-02-28 14:48:35.000000000 -0800

+++ lx26-20-mm2c/fs/autofs4/waitq.c 2007-02-28 15:47:21.000000000 -0800

@@ -296,8 +296,8 @@ 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);

Containers mailing list

Containers@lists.osdl.org

<https://lists.osdl.org/mailman/listinfo/containers>

Subject: Re: [PATCH 3/3] Replace pid_t in autofs4 with struct pid reference.

Posted by [ebiederm](#) on Fri, 09 Mar 2007 17:37:59 GMT

[View Forum Message](#) <> [Reply to Message](#)

sukadev@us.ibm.com writes:

> Index: lx26-20-mm2c/fs/autofs4/waitq.c

> =====

> --- lx26-20-mm2c.orig/fs/autofs4/waitq.c 2007-02-28 14:48:35.000000000 -0800

> +++ lx26-20-mm2c/fs/autofs4/waitq.c 2007-02-28 15:47:21.000000000 -0800

> @@ -296,8 +296,8 @@ 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);
```

I don't quite follow all of the waitq logic but it looks like these values are cached in the kernel and then eventually passed to a user space daemon.

Which looks to me like the wait queue should have a struct pid * and we should call pid_nr in the daemon process context. Otherwise the daemon might see the wrong pid values.

So I think autofs4 needs some more work.

Eric

Containers mailing list
Containers@lists.osdl.org
<https://lists.osdl.org/mailman/listinfo/containers>
