
Subject: [RFC][PATCH 1/3]: Replace pid_t in autofs with struct pid reference.
Posted by [Sukadev Bhattiprolu](#) on Thu, 25 Jan 2007 03:53:31 GMT
[View Forum Message](#) <> [Reply to Message](#)

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.

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

Subject: Replace pid_t in autofs with struct pid reference.

Make autofs container-friendly by caching struct pid reference rather than pid_t.

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/autofs/autofs_i.h | 4 ++--
fs/autofs/inode.c    | 33 ++++++-----
fs/autofs/root.c     | 6 ++++--
3 files changed, 30 insertions(+), 13 deletions(-)
```

Index: lx26-20-rc4-mm1/fs/autofs/autofs_i.h

```
=====
--- lx26-20-rc4-mm1.orig/fs/autofs/autofs_i.h 2007-01-24 17:29:32.471909792 -0800
+++ lx26-20-rc4-mm1/fs/autofs/autofs_i.h 2007-01-24 17:31:10.277041152 -0800
@@ -101,7 +101,7 @@ struct autofs_symlink {
 struct autofs_sb_info {
     u32 magic;
     struct file *pipe;
- pid_t oz_pgrp;
+ struct pid *oz_pgrp;
     int catatonic;
     struct super_block *sb;
     unsigned long exp_timeout;
@@ -122,7 +122,7 @@ static inline struct autofs_sb_info *aut
     filesystem without "magic".) */

 static inline int autofs_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;
}

/* Hash operations */
Index: lx26-20-rc4-mm1/fs/autofs/inode.c
=====
--- lx26-20-rc4-mm1.orig/fs/autofs/inode.c 2007-01-24 17:29:32.471909792 -0800
+++ lx26-20-rc4-mm1/fs/autofs/inode.c 2007-01-24 18:39:20.834181944 -0800
@@ -37,6 +37,8 @@ void autofs_kill_sb(struct super_block *
    if ( !sbi->catatonic )
        autofs_catatonic_mode(sbi); /* Free wait queues, close pipe */

+ put_pid(sbi->oz_pgrp);
+
    autofs_hash_nuke(sbi);
    for ( n = 0 ; n < AUTOFS_MAX_SYMLINKS ; n++ ) {
        if ( test_bit(n, sbi->symlink_bitmap) )
@@ -69,15 +71,17 @@ static match_table_t autofs_tokens = {
    {Opt_err, NULL}
};

-static int parse_options(char *options, int *pipefd, uid_t *uid, gid_t *gid, pid_t *pgrp, int *minproto,
int *maxproto)
+static int parse_options(char *options, int *pipefd, uid_t *uid, gid_t *gid,
+ struct pid **pgrp, int *minproto, int *maxproto)
{
    char *p;
    substring_t args[MAX_OPT_ARGS];
    int option;
+ pid_t pgid;

    *uid = current->uid;
    *gid = current->gid;
- *pgrp = process_group(current);
+ pgid = process_group(current);

    *minproto = *maxproto = AUTOFS_PROTO_VERSION;

@@ -111,7 +115,7 @@ static int parse_options(char *options,
    case Opt_pgrp:
        if (match_int(&args[0], &option))
            return 1;
- *pgrp = option;
+ pgid = option;
        break;
    case Opt_minproto:
        if (match_int(&args[0], &option))
@@ -127,7 +131,13 @@ static int parse_options(char *options,

```

```

    return 1;
}
}
- return (*pipefd < 0);
+
+ if (*pipefd < 0)
+ return 1;
+
+ *pgrp = find_get_pid(pgid);
+
+ return (*pgrp == NULL);
}

int autofs_fill_super(struct super_block *s, void *data, int silent)
@@ -149,7 +159,7 @@ int autofs_fill_super(struct super_block
    sbi->pipe = NULL;
    sbi->catatonic = 1;
    sbi->exp_timeout = 0;
- sbi->oz_pgrp = process_group(current);
+ sbi->oz_pgrp = NULL;
    autofs_initialize_hash(&sbi->dirhash);
    sbi->queues = NULL;
    memset(sbi->symlink_bitmap, 0, sizeof(long)*AUTOFS_SYMLINK_BITMAP_LEN);
@@ -169,7 +179,9 @@ int autofs_fill_super(struct super_block
    goto fail_iput;

    /* Can this call block? - WTF cares? s is locked. */
- if (
parse_options(data,&pipefd,&root_inode->i_uid,&root_inode->i_gid,&sbi->oz_pgrp,&minproto,&maxproto) ) {
+ if ( parse_options(data, &pipefd, &root_inode->i_uid,
+ &root_inode->i_gid, &sbi->oz_pgrp, &minproto,
+ &maxproto) ) {
    printk("autofs: called with bogus options\n");
    goto fail_dput;
}
@@ -178,15 +190,16 @@ int autofs_fill_super(struct super_block
    if ( minproto > AUTOFS_PROTO_VERSION ||
        maxproto < AUTOFS_PROTO_VERSION ) {
        printk("autofs: kernel does not match daemon version\n");
- goto fail_dput;
+ goto fail_put_pid;
    }

- DPRINTK(("autofs: pipe fd = %d, pgrp = %u\n", pipefd, sbi->oz_pgrp));
+ DPRINTK(("autofs: pipe fd = %d, pgrp = %u\n", pipefd,
+ pid_nr(sbi->oz_pgrp)));
    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;
@@ -202,6 +215,8 @@ int autofs_fill_super(struct super_block
fail_fput:
    printk("autofs: pipe file descriptor does not contain proper ops\n");
    fput(pipe);
+fail_put_pid:
+ put_pid(sbi->oz_pgrp);
fail_dput:
    dput(root);
    goto fail_free;
Index: lx26-20-rc4-mm1/fs/autofs/root.c
=====
--- lx26-20-rc4-mm1.orig/fs/autofs/root.c 2007-01-24 17:29:32.471909792 -0800
+++ lx26-20-rc4-mm1/fs/autofs/root.c 2007-01-24 18:39:20.834181944 -0800
@@ -213,8 +213,10 @@ static struct dentry *autofs_root_lookup
    sbi = autofs_sbi(dir->i_sb);

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

/*
 * Mark the dentry incomplete, but add it. This is needed so

```

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