Subject: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by Sukadev Bhattiprolu on Tue, 13 Mar 2007 04:51:00 GMT
View Forum Message <> Reply to Message

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [PATCH 2/2] Replace pid_t in autofs with struct pid reference.

Make autofs 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.
 - 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
Acked-by: Eric W. Biederman <ebiederm@xmission.com>

---
 fs/autofs/autofs_i.h | 4 ++--
 fs/autofs/inode.c | 20 ++++++++++++++++----
 fs/autofs/root.c | 6 ++++--
 3 files changed, 22 insertions(+), 8 deletions(-)

Index: lx26-21-rc3-mm2/fs/autofs/autofs_i.h
===
--- lx26-21-rc3-mm2.orig/fs/autofs/autofs_i.h 2007-03-12 17:12:05.000000000 -0700
+++ lx26-21-rc3-mm2/fs/autofs/autofs_i.h 2007-03-12 17:18:55.000000000 -0700
@@ -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: lx26-21-rc3-mm2/fs/autofs/inode.c
===================================================================
--- lx26-21-rc3-mm2.orig/fs/autofs/inode.c 2007-03-12 17:18:48.000000000 -0700
+++ lx26-21-rc3-mm2/fs/autofs/inode.c 2007-03-12 17:18:55.000000000 -0700
@@ -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))
@@ -139,6 +141,7 @@ int autofs_fill_super(struct super_block
  int pipefd;
  struct autofs_sb_info *sbi;
  int minproto, maxproto;
+ pid_t pgid;

  sbi = kzalloc(sizeof(*sbi), GFP_KERNEL);
  if (!sbi)
@@ -150,7 +153,6 @@ int autofs_fill_super(struct super_block
  sbi->pipe = NULL;
  sbi->catatonic = 1;
  sbi->exp_timeout = 0;
- sbi->oz_pgrp = process_group(current);
  autofs_initialize_hash(&sbi->dirhash);
  sbi->queues = NULL;
  memset(sbi->symlink_bitmap, 0, sizeof(long)*AUTOFS_SYMLINK_BITMAP_LEN);
@@ -171,7 +173,7 @@ int autofs_fill_super(struct super_block

  /* 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,
+    &root_inode->i_gid, &pgid, &minproto,
     &maxproto)) {
   printk("autofs: called with bogus options\n");
   goto fail_dput;
@@ -184,13 +186,21 @@ int autofs_fill_super(struct super_block
   goto fail_dput;
  }

- DPRINTK(("autofs: pipe fd = %d, pgrp = %u\n", pipefd, sbi->oz_pgrp));
```

```
+ DPRINTK(("autofs: pipe fd = %d, pgrp = %u\n", 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;
  sbi->pipe = pipe;
@@ -205,6 +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: lx26-21-rc3-mm2/fs/autofs/root.c
===================================================================
--- lx26-21-rc3-mm2.orig/fs/autofs/root.c 2007-03-12 17:18:48.000000000 -0700
+++ lx26-21-rc3-mm2/fs/autofs/root.c 2007-03-12 17:18:55.000000000 -0700
@@ -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
```

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by Ian Kent on Fri, 16 Mar 2007 04:04:52 GMT
View Forum Message <> Reply to Message

On Mon, 12 Mar 2007, sukadev@us.ibm.com wrote:

>
> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> Subject: [PATCH 2/2] Replace pid_t in autofs with struct pid reference.
>
> Make autofs 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.

What changes to autofs4?
Do you intend this change to be made for autofs4 also?
Perhaps you expected me to do them, in which case you probably should
ask me to do the patch.

> - 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
> Acked-by: Eric W. Biederman <ebiederm@xmission.com>
>
> ---
> fs/autofs/autofs_i.h | 4 ++--
> fs/autofs/inode.c | 20 ++++++++++++++++----
> fs/autofs/root.c | 6 ++++--
> 3 files changed, 22 insertions(+), 8 deletions(-)
>
> Index: lx26-21-rc3-mm2/fs/autofs/autofs_i.h
> ===
> --- lx26-21-rc3-mm2.orig/fs/autofs/autofs_i.h 2007-03-12 17:12:05.000000000 -0700
> +++ lx26-21-rc3-mm2/fs/autofs/autofs_i.h 2007-03-12 17:18:55.000000000 -0700
> @@ -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: lx26-21-rc3-mm2/fs/autofs/inode.c
> ===================================================================
> --- lx26-21-rc3-mm2.orig/fs/autofs/inode.c 2007-03-12 17:18:48.000000000 -0700
> +++ lx26-21-rc3-mm2/fs/autofs/inode.c 2007-03-12 17:18:55.000000000 -0700
> @@ -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))
> @@ -139,6 +141,7 @@ int autofs_fill_super(struct super_block
>   int pipefd;
>   struct autofs_sb_info *sbi;
>   int minproto, maxproto;
> + pid_t pgid;
>
>   sbi = kzalloc(sizeof(*sbi), GFP_KERNEL);
>   if (!sbi)
> @@ -150,7 +153,6 @@ int autofs_fill_super(struct super_block
>   sbi->pipe = NULL;
>   sbi->catatonic = 1;
>   sbi->exp_timeout = 0;
> - sbi->oz_pgrp = process_group(current);
>   autofs_initialize_hash(&sbi->dirhash);
>   sbi->queues = NULL;
>   memset(sbi->symlink_bitmap, 0, sizeof(long)*AUTOFS_SYMLINK_BITMAP_LEN);
> @@ -171,7 +173,7 @@ int autofs_fill_super(struct super_block
>
```

```
>   /* 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,
> +    &root_inode->i_gid, &pgid, &minproto,
>      &maxproto)) {
>    printk("autofs: called with bogus options\n");
>    goto fail_dput;
> @@ -184,13 +186,21 @@ int autofs_fill_super(struct super_block
>    goto fail_dput;
>   }
>
> - DPRINTK(("autofs: pipe fd = %d, pgrp = %u\n", pipefd, sbi->oz_pgrp));
> + DPRINTK(("autofs: pipe fd = %d, pgrp = %u\n", 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;
>   sbi->pipe = pipe;
> @@ -205,6 +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: lx26-21-rc3-mm2/fs/autofs/root.c
> ===================================================================
> --- lx26-21-rc3-mm2.orig/fs/autofs/root.c 2007-03-12 17:18:48.000000000 -0700
> +++ lx26-21-rc3-mm2/fs/autofs/root.c 2007-03-12 17:18:55.000000000 -0700
> @@ -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
> -
> To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html
> Please read the FAQ at http://www.tux.org/lkml/
>

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

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by ebiederm on Fri, 16 Mar 2007 11:32:28 GMT
View Forum Message <> Reply to Message

Ian Kent <raven@themaw.net> writes:

> On Mon, 12 Mar 2007, sukadev@us.ibm.com wrote:
>
>>
>> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
>> Subject: [PATCH 2/2] Replace pid_t in autofs with struct pid reference.
>>
>> Make autofs 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.
>
> What changes to autofs4?
> Do you intend this change to be made for autofs4 also?
> Perhaps you expected me to do them, in which case you probably should
> ask me to do the patch.

The review history goes something like this.
- That's a big patch why are you touching autofs and autofs4 at the

same time?
<split>
- Hmm. That change in the autofs4 patch looks fishy.
<patch postponed until the issue was addressed>

autofs4 uses pids more extensively than autofs and so the change is
correspondingly larger.

If you would like to look at what it would take to get autofs4 to only
store values as struct pid * instead of storing pid_t that would be great.
But for the most part I this is a massive global change that those of us
pushing it are responsible for changing as fixing up as much of the code
as we can, as is usual kernel practice.

Eric

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

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by Ian Kent on Fri, 16 Mar 2007 14:31:07 GMT
View Forum Message <> Reply to Message

On Fri, 2007-03-16 at 05:32 -0600, Eric W. Biederman wrote:
> Ian Kent <raven@themaw.net> writes:
>
> > On Mon, 12 Mar 2007, sukadev@us.ibm.com wrote:
> >
> >>
> >> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> >> Subject: [PATCH 2/2] Replace pid_t in autofs with struct pid reference.
> >>
> >> Make autofs 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.
> >
> > What changes to autofs4?
> > Do you intend this change to be made for autofs4 also?
> > Perhaps you expected me to do them, in which case you probably should
> > ask me to do the patch.
>
> The review history goes something like this.

> - That's a big patch why are you touching autofs and autofs4 at the
>   same time?
> <split>
> - Hmm. That change in the autofs4 patch looks fishy.
> <patch postponed until the issue was addressed>
>
> autofs4 uses pids more extensively than autofs and so the change is
> correspondingly larger.
>
> If you would like to look at what it would take to get autofs4 to only
> store values as struct pid * instead of storing pid_t that would be great.
> But for the most part I this is a massive global change that those of us
> pushing it are responsible for changing as fixing up as much of the code
> as we can, as is usual kernel practice.

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.

Ian

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);

_______________________________________________
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers
```

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by Ian Kent on Fri, 16 Mar 2007 16:46:58 GMT
View Forum Message <> Reply to Message

On Fri, 2007-03-16 at 15:44 +0100, Cedric Le Goater wrote:
> > 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.

That looks OK to me, assuming the "find_get_pid" and friends do what
they suggest, but I'll give it a closer look tomorrow.

A ref count is used here, what affect does that have on a thread (or

process) that may go away (or be summarily killed) without umounting the
mount?

Thanks
Ian

>
> 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);
```


Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by ebiederm on Fri, 16 Mar 2007 19:21:51 GMT

Ian Kent <raven@themaw.net> writes:

> On Fri, 2007-03-16 at 15:44 +0100, Cedric Le Goater wrote:
>> > 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.
>
> That looks OK to me, assuming the "find_get_pid" and friends do what
> they suggest, but I'll give it a closer look tomorrow.
>
> A ref count is used here, what affect does that have on a thread (or
> process) that may go away (or be summarily killed) without umounting the
> mount?

Nothing.

The primary advantage is that you are pid wrap around safe as the struct
pid will never point to another process after one of those events occurs.

struct pid is a very small structure so not freeing it when the process
it originally referred to goes away is no big deal. Although not leaking
when you stop using it is still important.

The other big use of struct pid is that to get the user space pid value
you call pid_nr(). Depending on the pid namespace of the caller the return
value of pid_nr() can be different. So when you store a pid or pass a pid
between processes that should be done by passing a struct pid because those
processes could be in different pid namespaces.

>> 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);

I have a concern with this bit as I my quick review said the wait queue
persists, and if so we should be cache the struct pid pointer, not the
pid_t value. Heck the whol pid_nr(task_xxx(current)) idiom I find very
suspicious.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by serue on Mon, 19 Mar 2007 20:08:39 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> Ian Kent <raven@themaw.net> writes:
>
> > On Fri, 2007-03-16 at 15:44 +0100, Cedric Le Goater wrote:
> >> > 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.
> >
> > That looks OK to me, assuming the "find_get_pid" and friends do what
> > they suggest, but I'll give it a closer look tomorrow.
> >
> > A ref count is used here, what affect does that have on a thread (or
> > process) that may go away (or be summarily killed) without umounting the
> > mount?
>
> Nothing.
>
> The primary advantage is that you are pid wrap around safe as the struct
> pid will never point to another process after one of those events occurs.
>
> struct pid is a very small structure so not freeing it when the process
> it originally referred to goes away is no big deal. Although not leaking
> when you stop using it is still important.
>
> The other big use of struct pid is that to get the user space pid value

> you call pid_nr(). Depending on the pid namespace of the caller the return
> value of pid_nr() can be different. So when you store a pid or pass a pid
> between processes that should be done by passing a struct pid because those
> processes could be in different pid namespaces.
>
> >> 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);
>
> I have a concern with this bit as I my quick review said the wait queue
> persists, and if so we should be cache the struct pid pointer, not the
> pid_t value. Heck the whol pid_nr(task_xxx(current)) idiom I find very
> suspicious.

Based just on what I see right here I agree it seems like we would want
to store a ref to the pid, not store the pid_nr(pid) output, so in this
context it is suspicious.

OTOH if you're saying that using pid_nr(task_pid(current)) anywhere
should always be 'wrong', then please explain why, as I think we have a
disagreement on the meanings of the structs involved. In other words,
at some point I expect the only way to get a "pid number" out of a task
would be using this exact idiom, "pid_nr(task_pid(current))".

-serge

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by ebiederm on Mon, 19 Mar 2007 20:40:06 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

>>
>> >> 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);
>>
>> I have a concern with this bit as I my quick review said the wait queue
>> persists, and if so we should be cache the struct pid pointer, not the
>> pid_t value. Heck the whol pid_nr(task_xxx(current)) idiom I find very
>> suspicious.
>
> Based just on what I see right here I agree it seems like we would want
> to store a ref to the pid, not store the pid_nr(pid) output, so in this
> context it is suspicious.

So that far we are in agreement.

> OTOH if you're saying that using pid_nr(task_pid(current)) anywhere
> should always be 'wrong', then please explain why, as I think we have a
> disagreement on the meanings of the structs involved. In other words,
> at some point I expect the only way to get a "pid number" out of a task
> would be using this exact idiom, "pid_nr(task_pid(current))".

Dealing with the current process is very common, and
"pid_nr(task_pid(current)" is very long winded. Therefore I think it
makes sense to have a specialized helper for that case.

I don't think "current->pid" and "current->tgid" are necessarily
wrong.

For "process_session(current)", and "process_group(current)" I think
they are fine but we might optimize them to something like:
"current_session()" and "current_group()".

The important part is that we have clearly detectable idioms for
finding the pid values. So we can find the users and audit the code.

Having a little more change so that the problem cases don't compile when they comes from a patch that hasn't caught up yet with the changes is also useful.

The only advantage I see in making everything go through something like: pid_nr(task_pid(current)) is that we don't have the problem of storing the pid value twice. However if we have short hand helper functions for that case it will still work and we won't be horribly wordy.

Further I don't know how expensive pid_nr is going to be, I don't think it will be very expensive. But I still think it may be reasonable to cache the answers for the current process on the task_struct. Fewer cache lines and all of that jazz.

Mostly I just think pid_nr(task_pid(xxx)) looks ugly is rarely needed and is frequently associated with a bad conversion.

Eric

_____

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by serue on Mon, 19 Mar 2007 21:19:59 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> >>
> >> >> 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);

> >>
> >> I have a concern with this bit as I my quick review said the wait queue
> >> persists, and if so we should be cache the struct pid pointer, not the
> >> pid_t value. Heck the whol pid_nr(task_xxx(current)) idiom I find very
> >> suspicious.
> >
> > Based just on what I see right here I agree it seems like we would want
> > to store a ref to the pid, not store the pid_nr(pid) output, so in this
> > context it is suspicious.
>
> So that far we are in agreement.
>
> > OTOH if you're saying that using pid_nr(task_pid(current)) anywhere
> > should always be 'wrong', then please explain why, as I think we have a
> > disagreement on the meanings of the structs involved. In other words,
> > at some point I expect the only way to get a "pid number" out of a task
> > would be using this exact idiom, "pid_nr(task_pid(current))".
>
> Dealing with the current process is very common, and
> "pid_nr(task_pid(current)" is very long winded. Therefore I think it
> makes sense to have a specialized helper for that case.
>
> I don't think "current->pid" and "current->tgid" are necessarily
> wrong.

True, current->pid can probably always be legitimately taken as the pid
number in the current task's cloning namespace. But task->pid is wrong.
So if as you say it's worth caching (not saying I doubt you, just that I
haven't verified), then ideally we could cache current->pid but only
access it using current_pid(). Does that seem worth doing?

In any case, certainly adding a task_pid_nr() helper which for starters
returns pid_nr(task_pid(task)) seems reasonable. Note that Suka's about
ready to send a new iteration of the pidns patchset, so I'd like this to
be considered something to clean up on top of that patchset.

-serge

> For "process_session(current)", and "process_group(current)" I think
> they are fine but we might optimize them to something like:
> "current_session()" and "current_group()".
>
> The important part is that we have clearly detectable idioms for
> finding the pid values. So we can find the users and audit the code.
> Having a little more change so that the problem cases don't compile
> when they comes from a patch that hasn't caught up yet with the changes
> is also useful.
>

> The only advantage I see in making everything go through something
> like: pid_nr(task_pid(current)) is that we don't have the problem of
> storing the pid value twice. However if we have short hand helper
> functions for that case it will still work and we won't be horribly
> wordy.
>
> Further I don't know how expensive pid_nr is going to be, I don't
> think it will be very expensive. But I still think it may be
> reasonable to cache the answers for the current process on the
> task_struct. Fewer cache lines and all of that jazz.
>
> Mostly I just think pid_nr(task_pid(xxx)) looks ugly is rarely needed
> and is frequently associated with a bad conversion.
>
> Eric

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by ebiederm on Mon, 19 Mar 2007 21:43:27 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

> True, current->pid can probably always be legitimately taken as the pid
> number in the current task's cloning namespace. But task->pid is
> wrong.

Agreed.

> So if as you say it's worth caching (not saying I doubt you, just that I
> haven't verified), then ideally we could cache current->pid but only
> access it using current_pid(). Does that seem worth doing?

Doing current_pid() and friends certainly seem to be worth doing.
Mostly for the secondary effect that we can then breaking any code we
miss in our conversion. I don't know if the caching is worth it but
it is so easy to implement I don't see a reason to avoid it.

> In any case, certainly adding a task_pid_nr() helper which for starters
> returns pid_nr(task_pid(task)) seems reasonable. Note that Suka's about
> ready to send a new iteration of the pidns patchset, so I'd like this to
> be considered something to clean up on top of that patchset.

I will keep it in mind. As long a we keep patches that actually change the

code separate from patches that change the set of helper functions I think
we will be ok. Hopefully Suka has learned enough by now that reviewing
his patches will be easier than just writing them myself. There were
so many silly little issues to clean up in the last round of review
by the time the patch was clean enough to look at the meat I missed the
deep issues :(At least things were clear enough that Oleg managed to
catch some of the deeper issues.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by serue on Tue, 20 Mar 2007 20:15:48 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> >>
> >> >> 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);
> >>
> >> I have a concern with this bit as I my quick review said the wait queue
> >> persists, and if so we should be cache the struct pid pointer, not the
> >> pid_t value. Heck the whol pid_nr(task_xxx(current)) idiom I find very
> >> suspicious.
> >
> > Based just on what I see right here I agree it seems like we would want
> > to store a ref to the pid, not store the pid_nr(pid) output, so in this
> > context it is suspicious.
>

> So that far we are in agreement.

So how about the following on top of the patch Cedric sent out?

Subject: [PATCH] autofs4: store struct pids in autofs_waitqs
From: Serge Hallyn <serue@us.ibm.com>
Date: 1174412305 -0500

Store struct pids in autofs_waitqs in place of pidnrs to prevent
pid overflow problems.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>

---

 fs/autofs4/autofs_i.h | 13 ++++++++++--
 fs/autofs4/waitq.c | 15 ++++++++-------
 2 files changed, 19 insertions(+), 9 deletions(-)

86a5866c7672b88d48380504977496d5637642ab
diff --git a/fs/autofs4/autofs_i.h b/fs/autofs4/autofs_i.h
index 3ccec0a..5d119e3 100644
--- a/fs/autofs4/autofs_i.h
+++ b/fs/autofs4/autofs_i.h
@@ -79,8 +79,8 @@ struct autofs_wait_queue {
  u64 ino;
  uid_t uid;
  gid_t gid;
- pid_t pid;
- pid_t tgid;
+ struct pid *pid;
+ struct pid *tgid;
  /* This is for status reporting upon return */
  int status;
  atomic_t wait_ctr;
@@ -228,5 +228,14 @@ out:
  return ret;
 }

+static void autofs_free_wait_queue(struct autofs_wait_queue *wq)
+{
+ if (wq->pid)
+ put_pid(wq->pid);
+ if (wq->tgid)
+ put_pid(wq->tgid);
+ kfree(wq);
+}
+

```
 void autofs4_dentry_release(struct dentry *);
 extern void autofs4_kill_sb(struct super_block *);
diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
index 9857543..4a9ad9b 100644
--- a/fs/autofs4/waitq.c
+++ b/fs/autofs4/waitq.c
@@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct
   packet->ino = wq->ino;
   packet->uid = wq->uid;
   packet->gid = wq->gid;
-  packet->pid = wq->pid;
-  packet->tgid = wq->tgid;
+  packet->pid = pid_nr(wq->pid);
+  packet->tgid = pid_nr(wq->tgid);
   break;
  }
  default:
@@ -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 = pid_nr(task_pid(current));
-  wq->tgid = pid_nr(task_tgid(current));
+  wq->pid = get_pid(task_pid(current));
+  wq->tgid = get_pid(task_tgid(current));
   wq->status = -EINTR; /* Status return if interrupted */
   atomic_set(&wq->wait_ctr, 2);
   mutex_unlock(&sbi->wq_mutex);
@@ -360,8 +360,9 @@ int autofs4_wait(struct autofs_sb_info *
  status = wq->status;

  /* Are we the last process to need status? */
- if (atomic_dec_and_test(&wq->wait_ctr))
-  kfree(wq);
+ if (atomic_dec_and_test(&wq->wait_ctr)) {
+  autofs_free_wait_queue(wq);
+ }

  return status;
 }
@@ -390,7 +391,7 @@ int autofs4_wait_release(struct autofs_s
  wq->status = status;

  if (atomic_dec_and_test(&wq->wait_ctr)) /* Is anyone still waiting for this guy? */
-  kfree(wq);
+  autofs_free_wait_queue(wq);
  else
   wake_up_interruptible(&wq->queue);
```

--
1.1.6

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by ebiederm on Tue, 20 Mar 2007 20:45:03 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

> So how about the following on top of the patch Cedric sent out?
>
> Subject: [PATCH] autofs4: store struct pids in autofs_waitqs
> From: Serge Hallyn <serue@us.ibm.com>
> Date: 1174412305 -0500
>
> Store struct pids in autofs_waitqs in place of pidnrs to prevent
> pid overflow problems.

At first skip it doesn't looks ok.

> Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
>
> ---
>
> fs/autofs4/autofs_i.h | 13 ++++++++++--
> fs/autofs4/waitq.c | 15 ++++++++-------
> 2 files changed, 19 insertions(+), 9 deletions(-)
>
> 86a5866c7672b88d48380504977496d5637642ab
> diff --git a/fs/autofs4/autofs_i.h b/fs/autofs4/autofs_i.h
> index 3ccec0a..5d119e3 100644
> --- a/fs/autofs4/autofs_i.h
> +++ b/fs/autofs4/autofs_i.h
> @@ -79,8 +79,8 @@ struct autofs_wait_queue {
>   u64 ino;
>   uid_t uid;
>   gid_t gid;
> - pid_t pid;
> - pid_t tgid;
> + struct pid *pid;
> + struct pid *tgid;
>   /* This is for status reporting upon return */

>   int status;
>   atomic_t wait_ctr;
> @@ -228,5 +228,14 @@ out:
>   return ret;
> }
>
> +static void autofs_free_wait_queue(struct autofs_wait_queue *wq)
> +{
> + if (wq->pid)
> + put_pid(wq->pid);
> + if (wq->tgid)
> + put_pid(wq->tgid);
> + kfree(wq);
> +}
> +
> void autofs4_dentry_release(struct dentry *);
> extern void autofs4_kill_sb(struct super_block *);
> diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
> index 9857543..4a9ad9b 100644
> --- a/fs/autofs4/waitq.c
> +++ b/fs/autofs4/waitq.c
> @@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct
>   packet->ino = wq->ino;
>   packet->uid = wq->uid;
>   packet->gid = wq->gid;
> - packet->pid = wq->pid;
> - packet->tgid = wq->tgid;
> + packet->pid = pid_nr(wq->pid);
> + packet->tgid = pid_nr(wq->tgid);
>   break;

I'm assuming we build the packet in the process context of the
daemon we are sending it to. If not we have a problem here.

>   }
>   default:
> @@ -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 = pid_nr(task_pid(current));
> - wq->tgid = pid_nr(task_tgid(current));
> + wq->pid = get_pid(task_pid(current));
> + wq->tgid = get_pid(task_tgid(current));
>   wq->status = -EINTR; /* Status return if interrupted */
>   atomic_set(&wq->wait_ctr, 2);
>   mutex_unlock(&sbi->wq_mutex);
> @@ -360,8 +360,9 @@ int autofs4_wait(struct autofs_sb_info *

>   status = wq->status;
>
>   /* Are we the last process to need status? */
> - if (atomic_dec_and_test(&wq->wait_ctr))
> - kfree(wq);
> + if (atomic_dec_and_test(&wq->wait_ctr)) {
> + autofs_free_wait_queue(wq);
> + }

Unnecessary braces....

>   return status;
> }
> @@ -390,7 +391,7 @@ int autofs4_wait_release(struct autofs_s
>   wq->status = status;
>
>   if (atomic_dec_and_test(&wq->wait_ctr)) /* Is anyone still waiting for
> this guy? */
> - kfree(wq);
> + autofs_free_wait_queue(wq);
>   else
>   wake_up_interruptible(&wq->queue);
>
> --
> 1.1.6

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by serue on Tue, 20 Mar 2007 21:41:25 GMT

View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> > So how about the following on top of the patch Cedric sent out?
> >
> > Subject: [PATCH] autofs4: store struct pids in autofs_waitqs
> > From: Serge Hallyn <serue@us.ibm.com>
> > Date: 1174412305 -0500
> >
> > Store struct pids in autofs_waitqs in place of pidnrs to prevent

> > pid overflow problems.
>
> At first skip it doesn't looks ok.
>
> > Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
> >
> > ---
> >
> > fs/autofs4/autofs_i.h | 13 ++++++++++--
> > fs/autofs4/waitq.c | 15 ++++++++-------
> > 2 files changed, 19 insertions(+), 9 deletions(-)
> >
> > 86a5866c7672b88d48380504977496d5637642ab
> > diff --git a/fs/autofs4/autofs_i.h b/fs/autofs4/autofs_i.h
> > index 3ccec0a..5d119e3 100644
> > --- a/fs/autofs4/autofs_i.h
> > +++ b/fs/autofs4/autofs_i.h
> > @@ -79,8 +79,8 @@ struct autofs_wait_queue {
> >   u64 ino;
> >   uid_t uid;
> >   gid_t gid;
> > - pid_t pid;
> > - pid_t tgid;
> > + struct pid *pid;
> > + struct pid *tgid;
> >   /* This is for status reporting upon return */
> >   int status;
> >   atomic_t wait_ctr;
> > @@ -228,5 +228,14 @@ out:
> >   return ret;
> > }
> >
> > +static void autofs_free_wait_queue(struct autofs_wait_queue *wq)
> > +{
> > + if (wq->pid)
> > + put_pid(wq->pid);
> > + if (wq->tgid)
> > + put_pid(wq->tgid);
> > + kfree(wq);
> > +}
> > +
> > void autofs4_dentry_release(struct dentry *);
> > extern void autofs4_kill_sb(struct super_block *);
> > diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
> > index 9857543..4a9ad9b 100644
> > --- a/fs/autofs4/waitq.c
> > +++ b/fs/autofs4/waitq.c
> > @@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct

> >   packet->ino = wq->ino;
> >   packet->uid = wq->uid;
> >   packet->gid = wq->gid;
> > - packet->pid = wq->pid;
> > - packet->tgid = wq->tgid;
> > + packet->pid = pid_nr(wq->pid);
> > + packet->tgid = pid_nr(wq->tgid);
> >   break;
>
> I'm assuming we build the packet in the process context of the
> daemon we are sending it to. If not we have a problem here.

Yes this is data being sent to a userspace daemon (Ian pls correct me if
I'm wrong) so the pid_nr is the only thing we can send.

> >   }
> >   default:
> > @@ -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 = pid_nr(task_pid(current));
> > - wq->tgid = pid_nr(task_tgid(current));
> > + wq->pid = get_pid(task_pid(current));
> > + wq->tgid = get_pid(task_tgid(current));
> >   wq->status = -EINTR; /* Status return if interrupted */
> >   atomic_set(&wq->wait_ctr, 2);
> >   mutex_unlock(&sbi->wq_mutex);
> > @@ -360,8 +360,9 @@ int autofs4_wait(struct autofs_sb_info *
> >   status = wq->status;
> >
> >   /* Are we the last process to need status? */
> > - if (atomic_dec_and_test(&wq->wait_ctr))
> > - kfree(wq);
> > + if (atomic_dec_and_test(&wq->wait_ctr)) {
> > + autofs_free_wait_queue(wq);
> > + }
>
> Unnecessary braces....

Thanks. Also needed another tweak in the .h file. Here's a new patch.
Ian, please let me know if this (combined with the previous patch from
Cedric) is acceptable to you.

thanks,
-serge

Subject: [PATCH] autofs4: store struct pids in autofs_waitqs

From: Serge Hallyn <serue@us.ibm.com>
Date: 1174412305 -0500

Store struct pids in autofs_waitqs in place of pidnrs to prevent
pid overflow problems.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>

---

 fs/autofs4/autofs_i.h | 13 ++++++++++--
 fs/autofs4/waitq.c | 12 ++++++------
 2 files changed, 17 insertions(+), 8 deletions(-)

03944aae0b9cb4a36d091f275d2f2217447b27ec
diff --git a/fs/autofs4/autofs_i.h b/fs/autofs4/autofs_i.h
index 3ccec0a..8260743 100644
--- a/fs/autofs4/autofs_i.h
+++ b/fs/autofs4/autofs_i.h
@@ -79,8 +79,8 @@ struct autofs_wait_queue {
  u64 ino;
  uid_t uid;
  gid_t gid;
- pid_t pid;
- pid_t tgid;
+ struct pid *pid;
+ struct pid *tgid;
  /* This is for status reporting upon return */
  int status;
  atomic_t wait_ctr;
@@ -228,5 +228,14 @@ out:
  return ret;
 }

+static inline void autofs_free_wait_queue(struct autofs_wait_queue *wq)
+{
+ if (wq->pid)
+ put_pid(wq->pid);
+ if (wq->tgid)
+ put_pid(wq->tgid);
+ kfree(wq);
+}
+
 void autofs4_dentry_release(struct dentry *);
 extern void autofs4_kill_sb(struct super_block *);
diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
index 9857543..6c696ea 100644
--- a/fs/autofs4/waitq.c

```
+++ b/fs/autofs4/waitq.c
@@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct
   packet->ino = wq->ino;
   packet->uid = wq->uid;
   packet->gid = wq->gid;
-  packet->pid = wq->pid;
-  packet->tgid = wq->tgid;
+  packet->pid = pid_nr(wq->pid);
+  packet->tgid = pid_nr(wq->tgid);
   break;
  }
  default:
@@ -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 = pid_nr(task_pid(current));
-  wq->tgid = pid_nr(task_tgid(current));
+  wq->pid = get_pid(task_pid(current));
+  wq->tgid = get_pid(task_tgid(current));
   wq->status = -EINTR; /* Status return if interrupted */
   atomic_set(&wq->wait_ctr, 2);
   mutex_unlock(&sbi->wq_mutex);
@@ -361,7 +361,7 @@ int autofs4_wait(struct autofs_sb_info *

  /* Are we the last process to need status? */
  if (atomic_dec_and_test(&wq->wait_ctr))
-  kfree(wq);
+  autofs_free_wait_queue(wq);

  return status;
 }
@@ -390,7 +390,7 @@ int autofs4_wait_release(struct autofs_s
  wq->status = status;

  if (atomic_dec_and_test(&wq->wait_ctr)) /* Is anyone still waiting for this guy? */
-  kfree(wq);
+  autofs_free_wait_queue(wq);
  else
   wake_up_interruptible(&wq->queue);

--
1.1.6
```

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by ebiederm on Tue, 20 Mar 2007 22:01:38 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

>> > void autofs4_dentry_release(struct dentry *);
>> > extern void autofs4_kill_sb(struct super_block *);
>> > diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
>> > index 9857543..4a9ad9b 100644
>> > --- a/fs/autofs4/waitq.c
>> > +++ b/fs/autofs4/waitq.c
>> > @@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct
>> >   packet->ino = wq->ino;
>> >   packet->uid = wq->uid;
>> >   packet->gid = wq->gid;
>> > - packet->pid = wq->pid;
>> > - packet->tgid = wq->tgid;
>> > + packet->pid = pid_nr(wq->pid);
>> > + packet->tgid = pid_nr(wq->tgid);
>> >   break;
>>
>> I'm assuming we build the packet in the process context of the
>> daemon we are sending it to. If not we have a problem here.
>
> Yes this is data being sent to a userspace daemon (Ian pls correct me if
> I'm wrong) so the pid_nr is the only thing we can send.

Agreed. The question is are we in the user space daemon's process when
we generate the pid_nr. Or do we stuff this in some kind of socket,
and the socket switch locations of the packet.

Basically I'm just trying to be certain we are calling pid_nr in the
proper context. Otherwise we could get the wrong pid when we have
multiple pid namespaces in play.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by serue on Wed, 21 Mar 2007 20:58:34 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):

> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> >> > void autofs4_dentry_release(struct dentry *);
> >> > extern void autofs4_kill_sb(struct super_block *);
> >> > diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
> >> > index 9857543..4a9ad9b 100644
> >> > --- a/fs/autofs4/waitq.c
> >> > +++ b/fs/autofs4/waitq.c
> >> > @@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct
> >> >   packet->ino = wq->ino;
> >> >   packet->uid = wq->uid;
> >> >   packet->gid = wq->gid;
> >> > - packet->pid = wq->pid;
> >> > - packet->tgid = wq->tgid;
> >> > + packet->pid = pid_nr(wq->pid);
> >> > + packet->tgid = pid_nr(wq->tgid);
> >> >   break;
> >>
> >> I'm assuming we build the packet in the process context of the
> >> daemon we are sending it to. If not we have a problem here.
> >
> > Yes this is data being sent to a userspace daemon (Ian pls correct me if
> > I'm wrong) so the pid_nr is the only thing we can send.
>
> Agreed. The question is are we in the user space daemon's process when
> we generate the pid_nr. Or do we stuff this in some kind of socket,
> and the socket switch locations of the packet.
>
> Basically I'm just trying to be certain we are calling pid_nr in the
> proper context. Otherwise we could get the wrong pid when we have
> multiple pid namespaces in play.

We need to know what the userspace daemon being written to is doing
with autofs_ptype_{missing,expire}_{in,}direct() messages.

If I understand correctly, the pid being sent is of a process which
tried to automount some directory. The message is being sent to the
autofs daemon, which should be running in the root pid namespace.

So as it is, the pid_nr(wq->pid) should be done under the init
pid_namespace, since it's a kthread. So as long as the userspace
automount daemon is started in the root pid namespace, the pid it gets
will be the right one.

Ian, does what I'm saying make sense, or am I wrong about how things
work for autofs?

thanks,

-serge

PS
Note that if I'm right, but some machine starts autofs in a child
pid_namespace, the pid_nr() the way I have it is wrong. I'm not sure in
that case how we go about fixing that. Somehow we need to store the
autofs userspace daemon's pid namespace pointer to help us find the
proper pid_nr.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by Ian Kent on Thu, 22 Mar 2007 02:00:51 GMT
View Forum Message <> Reply to Message

On Tue, 2007-03-20 at 16:01 -0600, Eric W. Biederman wrote:
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> >> > > void autofs4_dentry_release(struct dentry *);
> >> > > extern void autofs4_kill_sb(struct super_block *);
> >> > > diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
> >> > > index 9857543..4a9ad9b 100644
> >> > > --- a/fs/autofs4/waitq.c
> >> > > +++ b/fs/autofs4/waitq.c
> >> > > @@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct
> >> > > packet->ino = wq->ino;
> >> > > packet->uid = wq->uid;
> >> > > packet->gid = wq->gid;
> >> > > - packet->pid = wq->pid;
> >> > > - packet->tgid = wq->tgid;
> >> > > + packet->pid = pid_nr(wq->pid);
> >> > > + packet->tgid = pid_nr(wq->tgid);
> >> > > break;
> >>
> >> I'm assuming we build the packet in the process context of the
> >> daemon we are sending it to. If not we have a problem here.
> >
> > Yes this is data being sent to a userspace daemon (Ian pls correct me if
> > I'm wrong) so the pid_nr is the only thing we can send.
>
> Agreed. The question is are we in the user space daemon's process when
> we generate the pid_nr. Or do we stuff this in some kind of socket,
> and the socket switch locations of the packet.

The context here is the automount daemon only for expire runs.

Mount request packets are triggered by user processes walking over an
autofs mount point directory. So "current" in this case isn't the autofs
daemon.

Requests are sent via a pipe to the daemon.

Ian

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by serue on Thu, 22 Mar 2007 02:19:31 GMT
View Forum Message <> Reply to Message

Quoting Ian Kent (raven@themaw.net):
> On Tue, 2007-03-20 at 16:01 -0600, Eric W. Biederman wrote:
> > "Serge E. Hallyn" <serue@us.ibm.com> writes:
> >
> > > > > void autofs4_dentry_release(struct dentry *);
> > > > > extern void autofs4_kill_sb(struct super_block *);
> > > > > diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
> > > > > index 9857543..4a9ad9b 100644
> > > > > --- a/fs/autofs4/waitq.c
> > > > > +++ b/fs/autofs4/waitq.c
> > > > > @@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct
> > > > >   packet->ino = wq->ino;
> > > > >   packet->uid = wq->uid;
> > > > >   packet->gid = wq->gid;
> > > > > -   packet->pid = wq->pid;
> > > > > -   packet->tgid = wq->tgid;
> > > > > +   packet->pid = pid_nr(wq->pid);
> > > > > +   packet->tgid = pid_nr(wq->tgid);
> > > > >   break;
> > > >
> > > > I'm assuming we build the packet in the process context of the
> > > > daemon we are sending it to. If not we have a problem here.
> > >
> > > Yes this is data being sent to a userspace daemon (Ian pls correct me if
> > > I'm wrong) so the pid_nr is the only thing we can send.
> >
> > Agreed. The question is are we in the user space daemon's process when
> > we generate the pid_nr. Or do we stuff this in some kind of socket,

> > and the socket switch locations of the packet.
>
> The context here is the automount daemon only for expire runs.
>
> Mount request packets are triggered by user processes walking over an
> autofs mount point directory. So "current" in this case isn't the autofs
> daemon.
>
> Requests are sent via a pipe to the daemon.

So is the pid used for anything other than debugging?

In any case, here is a replacement patch which sends the pid number
in the pid_namespace of the process which did the autofs4 mount.

Still not sure whether that is actually what makes sense...

From: "Serge E. Hallyn" <serue@us.ibm.com>
Subject: [PATCH] autofs: prevent pid wraparound in waitqs

Instead of storing pid numbers for waitqs, store references
to struct pids. Also store a reference to the mounter's pid
namespace in the autofs4 sb info so that pid numbers for
mount miss and expiry msgs can send the pid# in the mounter's
pidns.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>

---

 fs/autofs4/autofs_i.h | 14 ++++++++++--
 fs/autofs4/inode.c | 5 +++++
 fs/autofs4/waitq.c | 12 ++++++------
 include/linux/pid.h | 1 +
 kernel/pid.c | 13 +++++++++---
 5 files changed, 34 insertions(+), 11 deletions(-)

e4184e6923f811f8a025b831ea33541fa820fd62
diff --git a/fs/autofs4/autofs_i.h b/fs/autofs4/autofs_i.h
index 3ccec0a..55026dd 100644
--- a/fs/autofs4/autofs_i.h
+++ b/fs/autofs4/autofs_i.h
@@ -79,8 +79,8 @@ struct autofs_wait_queue {
  u64 ino;
  uid_t uid;
  gid_t gid;
- pid_t pid;
- pid_t tgid;

```
+ struct pid *pid;
+ struct pid *tgid;
  /* This is for status reporting upon return */
  int status;
  atomic_t wait_ctr;
@@ -97,6 +97,7 @@ struct autofs_sb_info {
  int pipefd;
  struct file *pipe;
  struct pid *oz_pgrp;
+ struct pid_namespace *pidns;
  int catatonic;
  int version;
  int sub_version;
@@ -228,5 +229,14 @@ out:
  return ret;
 }

+static inline void autofs_free_wait_queue(struct autofs_wait_queue *wq)
+{
+ if (wq->pid)
+  put_pid(wq->pid);
+ if (wq->tgid)
+  put_pid(wq->tgid);
+ kfree(wq);
+}
+
 void autofs4_dentry_release(struct dentry *);
 extern void autofs4_kill_sb(struct super_block *);
diff --git a/fs/autofs4/inode.c b/fs/autofs4/inode.c
index c34131a..294efd8 100644
--- a/fs/autofs4/inode.c
+++ b/fs/autofs4/inode.c
@@ -20,6 +20,7 @@
 #include <linux/bitops.h>
 #include <linux/smp_lock.h>
 #include <linux/magic.h>
+#include <linux/pid_namespace.h>
 #include "autofs_i.h"
 #include <linux/module.h>

@@ -164,6 +165,7 @@ void autofs4_kill_sb(struct super_block
   autofs4_catatonic_mode(sbi); /* Free wait queues, close pipe */

  put_pid(sbi->oz_pgrp);
+ put_pid_ns(sbi->pidns);

  /* Clean up and release dangling references */
  autofs4_force_release(sbi);
```

```
@@ -334,6 +336,8 @@ int autofs4_fill_super(struct super_bloc
  sbi->type = 0;
  sbi->min_proto = 0;
  sbi->max_proto = 0;
+ sbi->pidns = task_pid_ns(current);
+ get_pid_ns(sbi->pidns);
  mutex_init(&sbi->wq_mutex);
  spin_lock_init(&sbi->fs_lock);
  sbi->queues = NULL;
@@ -435,6 +439,7 @@ fail_iput:
 fail_ino:
  kfree(ino);
 fail_free:
+ put_pid_ns(sbi->pidns);
  kfree(sbi);
  s->s_fs_info = NULL;
 fail_unlock:
diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
index 9857543..30e2a90 100644
--- a/fs/autofs4/waitq.c
+++ b/fs/autofs4/waitq.c
@@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct
   packet->ino = wq->ino;
   packet->uid = wq->uid;
   packet->gid = wq->gid;
-  packet->pid = wq->pid;
-  packet->tgid = wq->tgid;
+  packet->pid = __pid_nr(sbi->pidns, wq->pid);
+  packet->tgid = __pid_nr(sbi->pidns, wq->tgid);
   break;
  }
  default:
@@ -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 = pid_nr(task_pid(current));
-  wq->tgid = pid_nr(task_tgid(current));
+  wq->pid = get_pid(task_pid(current));
+  wq->tgid = get_pid(task_tgid(current));
   wq->status = -EINTR; /* Status return if interrupted */
   atomic_set(&wq->wait_ctr, 2);
   mutex_unlock(&sbi->wq_mutex);
@@ -361,7 +361,7 @@ int autofs4_wait(struct autofs_sb_info *

  /* Are we the last process to need status? */
  if (atomic_dec_and_test(&wq->wait_ctr))
-  kfree(wq);
```

```
+  autofs_free_wait_queue(wq);

  return status;
 }
@@ -390,7 +390,7 @@ int autofs4_wait_release(struct autofs_s
  wq->status = status;

  if (atomic_dec_and_test(&wq->wait_ctr)) /* Is anyone still waiting for this guy? */
-  kfree(wq);
+  autofs_free_wait_queue(wq);
  else
   wake_up_interruptible(&wq->queue);

diff --git a/include/linux/pid.h b/include/linux/pid.h
index d399679..4f043bf 100644
--- a/include/linux/pid.h
+++ b/include/linux/pid.h
@@ -120,6 +120,7 @@ extern void free_pid_nr(struct pid_nr *p
 extern struct pid_nr *alloc_pid_nr(struct pid_namespace *pid_ns);
 extern struct pid *alloc_pid(int clone_flags);
 extern void FASTCALL(free_pid(struct pid *pid));
+extern pid_t __pid_nr(struct pid_namespace *ns, struct pid *pid);
 extern pid_t pid_nr(struct pid *pid);


diff --git a/kernel/pid.c b/kernel/pid.c
index b040e19..aa61b7e 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
@@ -299,23 +299,30 @@ int attach_pid_nr(struct pid *pid, struc
  return 0;
 }

-pid_t pid_nr(struct pid *pid)
+pid_t __pid_nr(struct pid_namespace *ns, struct pid *pid)
 {
  struct pid_nr* pid_nr;
  struct hlist_node *pos;

  if (!pid)
   return 0;
-
  rcu_read_lock();
  hlist_for_each_entry_rcu(pid_nr, pos, &pid->pid_nrs, node)
-  if (pid_nr->pid_ns == task_pid_ns(current)) {
+  if (pid_nr->pid_ns == ns) {
    rcu_read_unlock();
    return pid_nr->nr;
```

```
  }

 rcu_read_unlock();
 return 0;
+
+}
+
+EXPORT_SYMBOL_GPL(__pid_nr);
+
+pid_t pid_nr(struct pid *pid)
+{
+ return __pid_nr(task_pid_ns(current), pid);
 }

 EXPORT_SYMBOL_GPL(pid_nr);
--
1.1.6
```

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by Ian Kent on Thu, 22 Mar 2007 02:28:43 GMT
View Forum Message <> Reply to Message

On Wed, 2007-03-21 at 15:58 -0500, Serge E. Hallyn wrote:
> Quoting Eric W. Biederman (ebiederm@xmission.com):
> > "Serge E. Hallyn" <serue@us.ibm.com> writes:
> >
> > > > > void autofs4_dentry_release(struct dentry *);
> > > > > extern void autofs4_kill_sb(struct super_block *);
> > > > > diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
> > > > > index 9857543..4a9ad9b 100644
> > > > > --- a/fs/autofs4/waitq.c
> > > > > +++ b/fs/autofs4/waitq.c
> > > > > @@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct
> > > > >   packet->ino = wq->ino;
> > > > >   packet->uid = wq->uid;
> > > > >   packet->gid = wq->gid;
> > > > > -   packet->pid = wq->pid;
> > > > > -   packet->tgid = wq->tgid;
> > > > > +   packet->pid = pid_nr(wq->pid);
> > > > > +   packet->tgid = pid_nr(wq->tgid);
> > > > >   break;
> > > > >
> > > > I'm assuming we build the packet in the process context of the

> > >> daemon we are sending it to. If not we have a problem here.
> > >
> > > Yes this is data being sent to a userspace daemon (Ian pls correct me if
> > > I'm wrong) so the pid_nr is the only thing we can send.
> >
> > Agreed. The question is are we in the user space daemon's process when
> > we generate the pid_nr. Or do we stuff this in some kind of socket,
> > and the socket switch locations of the packet.
> >
> > Basically I'm just trying to be certain we are calling pid_nr in the
> > proper context. Otherwise we could get the wrong pid when we have
> > multiple pid namespaces in play.
>
> We need to know what the userspace daemon being written to is doing
> with autofs_ptype_{missing,expire}_{in,}direct() messages.

At the moment autofs only uses the packet->pid for logging purposes.
This solves an age old problem of not knowing who is causing mount
requests.

I'm not aware of any other applications that use version 5 yet but that
of course could change. So we can't really know what will be done with
these ids at some point in the future.

>
> If I understand correctly, the pid being sent is of a process which
> tried to automount some directory. The message is being sent to the
> autofs daemon, which should be running in the root pid namespace.

Yes, but it could be the autofs daemon itself in the expire case.

Usually it doesn't make sense to run an automounting application as
other than "root" but I'm not familiar with other possible userspace
applications. Perhaps User Mode Linux could be an issue?

>
> So as it is, the pid_nr(wq->pid) should be done under the init
> pid_namespace, since it's a kthread. So as long as the userspace
> automount daemon is started in the root pid namespace, the pid it gets
> will be the right one.
>
> Ian, does what I'm saying make sense, or am I wrong about how things
> work for autofs?

Yep. That's the way it is.

>
> thanks,

> -serge
>
> PS
> Note that if I'm right, but some machine starts autofs in a child
> pid_namespace, the pid_nr() the way I have it is wrong. I'm not sure in
> that case how we go about fixing that. Somehow we need to store the
> autofs userspace daemon's pid namespace pointer to help us find the
> proper pid_nr.

In order for any user space application to use the module it must mount the autofs file system, passing a file handle for the pipe to use for communication. This must always be done. Can we grab the process pid namespace at that time and store it in the superblock info structure?

How does this affect getting ids for waitq request packets of other user space processes triggering mounts? I'm guessing that they would need to belong to the appropriate namespace for this mechanism to funtion correctly.

Ian

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by Ian Kent on Thu, 22 Mar 2007 03:18:14 GMT
View Forum Message <> Reply to Message

On Wed, 2007-03-21 at 21:19 -0500, Serge E. Hallyn wrote:
> Quoting Ian Kent (raven@themaw.net):
> > On Tue, 2007-03-20 at 16:01 -0600, Eric W. Biederman wrote:
> > > "Serge E. Hallyn" <serue@us.ibm.com> writes:
> > >
> > > >> void autofs4_dentry_release(struct dentry *);
> > > >> extern void autofs4_kill_sb(struct super_block *);
> > > >> diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
> > > >> index 9857543..4a9ad9b 100644
> > > >> --- a/fs/autofs4/waitq.c
> > > >> +++ b/fs/autofs4/waitq.c
> > > >> @@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct
> > > >>   packet->ino = wq->ino;
> > > >>   packet->uid = wq->uid;
> > > >>   packet->gid = wq->gid;
> > > >> -   packet->pid = wq->pid;

> > > >> > - packet->tgid = wq->tgid;
> > > >> > + packet->pid = pid_nr(wq->pid);
> > > >> > + packet->tgid = pid_nr(wq->tgid);
> > > >> >   break;
> > > >>
> > > >> I'm assuming we build the packet in the process context of the
> > > >> daemon we are sending it to. If not we have a problem here.
> > > >
> > > > Yes this is data being sent to a userspace daemon (Ian pls correct me if
> > > > I'm wrong) so the pid_nr is the only thing we can send.
> > >
> > > Agreed. The question is are we in the user space daemon's process when
> > > we generate the pid_nr. Or do we stuff this in some kind of socket,
> > > and the socket switch locations of the packet.
> >
> > The context here is the automount daemon only for expire runs.
> >
> > Mount request packets are triggered by user processes walking over an
> > autofs mount point directory. So "current" in this case isn't the autofs
> > daemon.
> >
> > Requests are sent via a pipe to the daemon.
>
> So is the pid used for anything other than debugging?
>
> In any case, here is a replacement patch which sends the pid number
> in the pid_namespace of the process which did the autofs4 mount.
>
> Still not sure whether that is actually what makes sense...
>
> From: "Serge E. Hallyn" <serue@us.ibm.com>
> Subject: [PATCH] autofs: prevent pid wraparound in waitqs
>
> Instead of storing pid numbers for waitqs, store references
> to struct pids. Also store a reference to the mounter's pid
> namespace in the autofs4 sb info so that pid numbers for
> mount miss and expiry msgs can send the pid# in the mounter's
> pidns.

I think this amounts to what I suggested in my previous replies.
Hopefully my comments are enough to clear up any questions on
correctness of this approach.

Sorry to be a pain but I'm having a little trouble reviewing the patch
because I'm not clear on where the code to handle the automount process
group (so called oz_pgrp), from the first patch, fits in with this.

Is this patch in addition to the original?

If so are the references to pid_nr still OK?

Ian

>
> Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
>
> ---
>
> fs/autofs4/autofs_i.h | 14 ++++++++++--
> fs/autofs4/inode.c | 5 +++++
> fs/autofs4/waitq.c | 12 ++++++------
> include/linux/pid.h | 1 +
> kernel/pid.c | 13 +++++++++---
> 5 files changed, 34 insertions(+), 11 deletions(-)
>
> e4184e6923f811f8a025b831ea33541fa820fd62
> diff --git a/fs/autofs4/autofs_i.h b/fs/autofs4/autofs_i.h
> index 3ccec0a..55026dd 100644
> --- a/fs/autofs4/autofs_i.h
> +++ b/fs/autofs4/autofs_i.h
> @@ -79,8 +79,8 @@ struct autofs_wait_queue {
>   u64 ino;
>   uid_t uid;
>   gid_t gid;
> - pid_t pid;
> - pid_t tgid;
> + struct pid *pid;
> + struct pid *tgid;
>   /* This is for status reporting upon return */
>   int status;
>   atomic_t wait_ctr;
> @@ -97,6 +97,7 @@ struct autofs_sb_info {
>   int pipefd;
>   struct file *pipe;
>   struct pid *oz_pgrp;
> + struct pid_namespace *pidns;
>   int catatonic;
>   int version;
>   int sub_version;
> @@ -228,5 +229,14 @@ out:
>   return ret;
> }
>
> +static inline void autofs_free_wait_queue(struct autofs_wait_queue *wq)
> +{
> + if (wq->pid)
> + put_pid(wq->pid);

> + if (wq->tgid)
> + put_pid(wq->tgid);
> + kfree(wq);
> +}
> +
> void autofs4_dentry_release(struct dentry *);
> extern void autofs4_kill_sb(struct super_block *);
> diff --git a/fs/autofs4/inode.c b/fs/autofs4/inode.c
> index c34131a..294efd8 100644
> --- a/fs/autofs4/inode.c
> +++ b/fs/autofs4/inode.c
> @@ -20,6 +20,7 @@
> #include <linux/bitops.h>
> #include <linux/smp_lock.h>
> #include <linux/magic.h>
> +#include <linux/pid_namespace.h>
> #include "autofs_i.h"
> #include <linux/module.h>
>
> @@ -164,6 +165,7 @@ void autofs4_kill_sb(struct super_block
>   autofs4_catatonic_mode(sbi); /* Free wait queues, close pipe */
>
>   put_pid(sbi->oz_pgrp);
> + put_pid_ns(sbi->pidns);
>
>   /* Clean up and release dangling references */
>   autofs4_force_release(sbi);
> @@ -334,6 +336,8 @@ int autofs4_fill_super(struct super_bloc
>   sbi->type = 0;
>   sbi->min_proto = 0;
>   sbi->max_proto = 0;
> + sbi->pidns = task_pid_ns(current);
> + get_pid_ns(sbi->pidns);
>   mutex_init(&sbi->wq_mutex);
>   spin_lock_init(&sbi->fs_lock);
>   sbi->queues = NULL;
> @@ -435,6 +439,7 @@ fail_iput:
> fail_ino:
>   kfree(ino);
> fail_free:
> + put_pid_ns(sbi->pidns);
>   kfree(sbi);
>   s->s_fs_info = NULL;
> fail_unlock:
> diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
> index 9857543..30e2a90 100644
> --- a/fs/autofs4/waitq.c
> +++ b/fs/autofs4/waitq.c

```
> @@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct
>    packet->ino = wq->ino;
>    packet->uid = wq->uid;
>    packet->gid = wq->gid;
> -  packet->pid = wq->pid;
> -  packet->tgid = wq->tgid;
> +  packet->pid = __pid_nr(sbi->pidns, wq->pid);
> +  packet->tgid = __pid_nr(sbi->pidns, wq->tgid);
>    break;
>   }
>   default:
> @@ -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 = pid_nr(task_pid(current));
> -  wq->tgid = pid_nr(task_tgid(current));
> +  wq->pid = get_pid(task_pid(current));
> +  wq->tgid = get_pid(task_tgid(current));
>    wq->status = -EINTR; /* Status return if interrupted */
>    atomic_set(&wq->wait_ctr, 2);
>    mutex_unlock(&sbi->wq_mutex);
> @@ -361,7 +361,7 @@ int autofs4_wait(struct autofs_sb_info *
>
>   /* Are we the last process to need status? */
>   if (atomic_dec_and_test(&wq->wait_ctr))
> -  kfree(wq);
> +  autofs_free_wait_queue(wq);
>
>   return status;
>  }
> @@ -390,7 +390,7 @@ int autofs4_wait_release(struct autofs_s
>   wq->status = status;
>
>   if (atomic_dec_and_test(&wq->wait_ctr)) /* Is anyone still waiting for this guy? */
> -  kfree(wq);
> +  autofs_free_wait_queue(wq);
>   else
>    wake_up_interruptible(&wq->queue);
>
> diff --git a/include/linux/pid.h b/include/linux/pid.h
> index d399679..4f043bf 100644
> --- a/include/linux/pid.h
> +++ b/include/linux/pid.h
> @@ -120,6 +120,7 @@ extern void free_pid_nr(struct pid_nr *p
>  extern struct pid_nr *alloc_pid_nr(struct pid_namespace *pid_ns);
>  extern struct pid *alloc_pid(int clone_flags);
>  extern void FASTCALL(free_pid(struct pid *pid));
```

```
> +extern pid_t __pid_nr(struct pid_namespace *ns, struct pid *pid);
>  extern pid_t pid_nr(struct pid *pid);
>
>
> diff --git a/kernel/pid.c b/kernel/pid.c
> index b040e19..aa61b7e 100644
> --- a/kernel/pid.c
> +++ b/kernel/pid.c
> @@ -299,23 +299,30 @@ int attach_pid_nr(struct pid *pid, struc
>   return 0;
>  }
>
> -pid_t pid_nr(struct pid *pid)
> +pid_t __pid_nr(struct pid_namespace *ns, struct pid *pid)
>  {
>   struct pid_nr* pid_nr;
>   struct hlist_node *pos;
>
>   if (!pid)
>    return 0;
> -
>   rcu_read_lock();
>   hlist_for_each_entry_rcu(pid_nr, pos, &pid->pid_nrs, node)
> -  if (pid_nr->pid_ns == task_pid_ns(current)) {
> +  if (pid_nr->pid_ns == ns) {
>     rcu_read_unlock();
>     return pid_nr->nr;
>    }
>
>   rcu_read_unlock();
>   return 0;
> +
> +}
> +
> +EXPORT_SYMBOL_GPL(__pid_nr);
> +
> +pid_t pid_nr(struct pid *pid)
> +{
> + return __pid_nr(task_pid_ns(current), pid);
>  }
>
>  EXPORT_SYMBOL_GPL(pid_nr);
```

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by ebiederm on Thu, 22 Mar 2007 06:43:08 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> So is the pid used for anything other than debugging?
>
> In any case, here is a replacement patch which sends the pid number
> in the pid_namespace of the process which did the autofs4 mount.
>
> Still not sure whether that is actually what makes sense...
>
> From: "Serge E. Hallyn" <serue@us.ibm.com>
> Subject: [PATCH] autofs: prevent pid wraparound in waitqs
>
> Instead of storing pid numbers for waitqs, store references
> to struct pids. Also store a reference to the mounter's pid
> namespace in the autofs4 sb info so that pid numbers for
> mount miss and expiry msgs can send the pid# in the mounter's
> pidns.

Hmm. Not quite what I would have expected but given that
we are sending data over a pipe that sounds reasonable.

If it wasn't a pipe we would really want to do this in
the context of the process receiving the message, but since
a pipe can receive a message, and then be passed to another
process we clearly can't know the pid namespace of the
process receiving the message.

Therefore just caching the pid namespace either on pipe
open or on mount makes sense. pipe open might be better.

Serge we really need to introduce __pid_nr in a separate
patch. And we really seem to be confusing Ian.

Plus we have some pid namespace ref counting issues we need
to handle carefully.

Let's stop working on autofs4 for a bit, fix the pid namespace
infrastructure so there is enough of it to handle autofs4 and
then come back.

Either that or take autofs4 in two passes. Pass one we do what
we can with the current infrastructure. Pass two after we fix up
the infrastructure including introducing __pid_nr we come back
and update autofs4 to handle multiple pid namespaces properly.

Eric

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by serue on Thu, 22 Mar 2007 13:28:53 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
> >
> > So is the pid used for anything other than debugging?
> >
> > In any case, here is a replacement patch which sends the pid number
> > in the pid_namespace of the process which did the autofs4 mount.
> >
> > Still not sure whether that is actually what makes sense...
> >
> > From: "Serge E. Hallyn" <serue@us.ibm.com>
> > Subject: [PATCH] autofs: prevent pid wraparound in waitqs
> >
> > Instead of storing pid numbers for waitqs, store references
> > to struct pids. Also store a reference to the mounter's pid
> > namespace in the autofs4 sb info so that pid numbers for
> > mount miss and expiry msgs can send the pid# in the mounter's
> > pidns.
>
> Hmm. Not quite what I would have expected but given that
> we are sending data over a pipe that sounds reasonable.
>
> If it wasn't a pipe we would really want to do this in
> the context of the process receiving the message, but since
> a pipe can receive a message, and then be passed to another
> process we clearly can't know the pid namespace of the
> process receiving the message.
>
> Therefore just caching the pid namespace either on pipe
> open or on mount makes sense. pipe open might be better.

Right, but the pipe is always opened on mount I think. (at
autofs4_fill_super)

> Serge we really need to introduce __pid_nr in a separate
> patch.

Agreed.

> And we really seem to be confusing Ian.
>
> Plus we have some pid namespace ref counting issues we need
> to handle carefully.
>
> Let's stop working on autofs4 for a bit, fix the pid namespace
> infrastructure so there is enough of it to handle autofs4 and
> then come back.

Agreed. I just wasn't comfortable stopping until I felt we knew how
autofs4 was going to be addressed. I think we know now, plus we've
verified another definite need for the __pid_nr(pidns, pid) helper.

> Either that or take autofs4 in two passes. Pass one we do what
> we can with the current infrastructure. Pass two after we fix up
> the infrastructure including introducing __pid_nr we come back
> and update autofs4 to handle multiple pid namespaces properly.

Nah, let's hold off, and I'll sit on a patch to send out once rest of
the infrastructure goes in.

-serge

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by serue on Thu, 22 Mar 2007 13:31:54 GMT
View Forum Message <> Reply to Message

Quoting Ian Kent (raven@themaw.net):
> On Wed, 2007-03-21 at 21:19 -0500, Serge E. Hallyn wrote:
> > Quoting Ian Kent (raven@themaw.net):
> > > On Tue, 2007-03-20 at 16:01 -0600, Eric W. Biederman wrote:
> > > > "Serge E. Hallyn" <serue@us.ibm.com> writes:
> > > >
> > > > > void autofs4_dentry_release(struct dentry *);
> > > > > extern void autofs4_kill_sb(struct super_block *);
> > > > > diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
> > > > > index 9857543..4a9ad9b 100644
> > > > > --- a/fs/autofs4/waitq.c
> > > > > +++ b/fs/autofs4/waitq.c
> > > > > @@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct
> > > > >   packet->ino = wq->ino;

> > > > >> >   packet->uid = wq->uid;
> > > > >> >   packet->gid = wq->gid;
> > > > >> > - packet->pid = wq->pid;
> > > > >> > - packet->tgid = wq->tgid;
> > > > >> > + packet->pid = pid_nr(wq->pid);
> > > > >> > + packet->tgid = pid_nr(wq->tgid);
> > > > >> >   break;
> > > > >>
> > > > >> I'm assuming we build the packet in the process context of the
> > > > >> daemon we are sending it to. If not we have a problem here.
> > > > >
> > > > > Yes this is data being sent to a userspace daemon (Ian pls correct me if
> > > > > I'm wrong) so the pid_nr is the only thing we can send.
> > > >
> > > > Agreed. The question is are we in the user space daemon's process when
> > > > we generate the pid_nr. Or do we stuff this in some kind of socket,
> > > > and the socket switch locations of the packet.
> > >
> > > The context here is the automount daemon only for expire runs.
> > >
> > > Mount request packets are triggered by user processes walking over an
> > > autofs mount point directory. So "current" in this case isn't the autofs
> > > daemon.
> > >
> > > Requests are sent via a pipe to the daemon.
> >
> > So is the pid used for anything other than debugging?
> >
> > In any case, here is a replacement patch which sends the pid number
> > in the pid_namespace of the process which did the autofs4 mount.
> >
> > Still not sure whether that is actually what makes sense...
> >
> > From: "Serge E. Hallyn" <serue@us.ibm.com>
> > Subject: [PATCH] autofs: prevent pid wraparound in waitqs
> >
> > Instead of storing pid numbers for waitqs, store references
> > to struct pids. Also store a reference to the mounter's pid
> > namespace in the autofs4 sb info so that pid numbers for
> > mount miss and expiry msgs can send the pid# in the mounter's
> > pidns.
>
> I think this amounts to what I suggested in my previous replies.
> Hopefully my comments are enough to clear up any questions on
> correctness of this approach.
>
> Sorry to be a pain but I'm having a little trouble reviewing the patch
> because I'm not clear on where the code to handle the automount process

> group (so called oz_pgrp), from the first patch, fits in with this.

It also has pidspace infrastructure code in it, so I think we will just
hold off on this until we have that infrastructure merged into the
pidspace code and into -mm. Then we can send you a single, more concise
patch.

> Is this patch in addition to the original?

Yes.

> If so are the references to pid_nr still OK?

I think so, because AIUI the rest are all executed in a context where
current is both the actor and recipient.

Thanks for your help.

thanks,
-serge

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by Herbert Poetzl on Thu, 22 Mar 2007 14:33:12 GMT
View Forum Message <> Reply to Message

On Thu, Mar 22, 2007 at 11:28:43AM +0900, Ian Kent wrote:
> On Wed, 2007-03-21 at 15:58 -0500, Serge E. Hallyn wrote:
> > Quoting Eric W. Biederman (ebiederm@xmission.com):
> > > "Serge E. Hallyn" <serue@us.ibm.com> writes:
> > >
> > > > void autofs4_dentry_release(struct dentry *);
> > > > extern void autofs4_kill_sb(struct super_block *);
> > > > diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
> > > > index 9857543..4a9ad9b 100644
> > > > --- a/fs/autofs4/waitq.c
> > > > +++ b/fs/autofs4/waitq.c
> > > > @@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct
> > > >   packet->ino = wq->ino;
> > > >   packet->uid = wq->uid;
> > > >   packet->gid = wq->gid;
> > > > - packet->pid = wq->pid;
> > > > - packet->tgid = wq->tgid;
> > > > + packet->pid = pid_nr(wq->pid);

> > > >> > + packet->tgid = pid_nr(wq->tgid);
> > > >> >   break;
> > > >>
> > > >> I'm assuming we build the packet in the process context of the
> > > >> daemon we are sending it to. If not we have a problem here.
> > > >
> > > > Yes this is data being sent to a userspace daemon (Ian pls
> > > > correct me if I'm wrong) so the pid_nr is the only thing we can
> > > > send.
> > >
> > > Agreed. The question is are we in the user space daemon's process
> > > when we generate the pid_nr. Or do we stuff this in some kind of
> > > socket, and the socket switch locations of the packet.
> > >
> > > Basically I'm just trying to be certain we are calling pid_nr in the
> > > proper context. Otherwise we could get the wrong pid when we have
> > > multiple pid namespaces in play.
> >
> > We need to know what the userspace daemon being written to is doing
> > with autofs_ptype_{missing,expire}_{in,}direct() messages.
>
> At the moment autofs only uses the packet->pid for logging purposes.
> This solves an age old problem of not knowing who is causing mount
> requests.

probably I'm wrong, but that sounds like the packet->pid
is supposed to be the pid of the process _causing_ the mount,
not the user space daemon communicating with the kernel ...

> I'm not aware of any other applications that use version 5 yet but
> that of course could change. So we can't really know what will be done
> with these ids at some point in the future.
>
> > If I understand correctly, the pid being sent is of a process which
> > tried to automount some directory. The message is being sent to the
> > autofs daemon, which should be running in the root pid namespace.
>
> Yes, but it could be the autofs daemon itself in the expire case.
>
> Usually it doesn't make sense to run an automounting application as
> other than "root" but I'm not familiar with other possible userspace
> applications. Perhaps User Mode Linux could be an issue?
>
> >
> > So as it is, the pid_nr(wq->pid) should be done under the init
> > pid_namespace, since it's a kthread. So as long as the userspace
> > automount daemon is started in the root pid namespace, the pid it
> > gets will be the right one.

> >
> > Ian, does what I'm saying make sense, or am I wrong about how things
> > work for autofs?
>
> Yep. That's the way it is.

assumed we allow auto mounter mounts inside a context
(I see no immediate reason not to do that) we want to
know the name/pid space the userspace daemon is running
in as well as the name/pid space of the trigger task

> > thanks,
> > -serge
> >
> > PS
> > Note that if I'm right, but some machine starts autofs in a child
> > pid_namespace, the pid_nr() the way I have it is wrong. I'm not sure in
> > that case how we go about fixing that. Somehow we need to store the
> > autofs userspace daemon's pid namespace pointer to help us find the
> > proper pid_nr.
>
> In order for any user space application to use the module it must mount
> the autofs file system, passing a file handle for the pipe to use for
> communication. This must always be done. Can we grab the process pid
> namespace at that time and store it in the superblock info structure?

probably yes, but if my assumption above is correct, that
isn't necessarily the pid/space for the trigger process
(although it makes sense that it _should_ be)

best,
Herbert

> How does this affect getting ids for waitq request packets of other user
> space processes triggering mounts? I'm guessing that they would need to
> belong to the appropriate namespace for this mechanism to funtion
> correctly.
>
> Ian
>

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference

On Thu, 2007-03-22 at 08:31 -0500, Serge E. Hallyn wrote:
> Quoting Ian Kent (raven@themaw.net):
> > On Wed, 2007-03-21 at 21:19 -0500, Serge E. Hallyn wrote:
> > > Quoting Ian Kent (raven@themaw.net):
> > > > On Tue, 2007-03-20 at 16:01 -0600, Eric W. Biederman wrote:
> > > > > "Serge E. Hallyn" <serue@us.ibm.com> writes:
> > > > >
> > > > > > void autofs4_dentry_release(struct dentry *);
> > > > > > extern void autofs4_kill_sb(struct super_block *);
> > > > > > diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
> > > > > > index 9857543..4a9ad9b 100644
> > > > > > --- a/fs/autofs4/waitq.c
> > > > > > +++ b/fs/autofs4/waitq.c
> > > > > > @@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct
> > > > > >   packet->ino = wq->ino;
> > > > > >   packet->uid = wq->uid;
> > > > > >   packet->gid = wq->gid;
> > > > > > - packet->pid = wq->pid;
> > > > > > - packet->tgid = wq->tgid;
> > > > > > + packet->pid = pid_nr(wq->pid);
> > > > > > + packet->tgid = pid_nr(wq->tgid);
> > > > > >   break;
> > > > > >
> > > > > I'm assuming we build the packet in the process context of the
> > > > > daemon we are sending it to. If not we have a problem here.
> > > > >
> > > > Yes this is data being sent to a userspace daemon (Ian pls correct me if
> > > > I'm wrong) so the pid_nr is the only thing we can send.
> > > >
> > > Agreed. The question is are we in the user space daemon's process when
> > > we generate the pid_nr. Or do we stuff this in some kind of socket,
> > > and the socket switch locations of the packet.
> > >
> > The context here is the automount daemon only for expire runs.
> >
> > Mount request packets are triggered by user processes walking over an
> > autofs mount point directory. So "current" in this case isn't the autofs
> > daemon.
> >
> > Requests are sent via a pipe to the daemon.
> >
> So is the pid used for anything other than debugging?
>
> In any case, here is a replacement patch which sends the pid number
> in the pid_namespace of the process which did the autofs4 mount.
>

> > > Still not sure whether that is actually what makes sense...
> > >
> > > From: "Serge E. Hallyn" <serue@us.ibm.com>
> > > Subject: [PATCH] autofs: prevent pid wraparound in waitqs
> > >
> > > Instead of storing pid numbers for waitqs, store references
> > > to struct pids. Also store a reference to the mounter's pid
> > > namespace in the autofs4 sb info so that pid numbers for
> > > mount miss and expiry msgs can send the pid# in the mounter's
> > > pidns.
> >
> > I think this amounts to what I suggested in my previous replies.
> > Hopefully my comments are enough to clear up any questions on
> > correctness of this approach.
> >
> > Sorry to be a pain but I'm having a little trouble reviewing the patch
> > because I'm not clear on where the code to handle the automount process
> > group (so called oz_pgrp), from the first patch, fits in with this.
>
> It also has pidspace infrastructure code in it, so I think we will just
> hold off on this until we have that infrastructure merged into the
> pidspace code and into -mm. Then we can send you a single, more concise
> patch.
>
> > Is this patch in addition to the original?
>
> Yes.
>
> > If so are the references to pid_nr still OK?
>
> I think so, because AIUI the rest are all executed in a context where
> current is both the actor and recipient.

Not sure about that but lets not worry about it until we have a full
patch to look at.

> Thanks for your help.

My pleasure.

Ian

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by Ian Kent on Thu, 22 Mar 2007 15:03:57 GMT
View Forum Message <> Reply to Message

On Thu, 2007-03-22 at 15:33 +0100, Herbert Poetzl wrote:
> On Thu, Mar 22, 2007 at 11:28:43AM +0900, Ian Kent wrote:
> > On Wed, 2007-03-21 at 15:58 -0500, Serge E. Hallyn wrote:
> > > Quoting Eric W. Biederman (ebiederm@xmission.com):
> > > > "Serge E. Hallyn" <serue@us.ibm.com> writes:
> > > >
> > > > > void autofs4_dentry_release(struct dentry *);
> > > > > extern void autofs4_kill_sb(struct super_block *);
> > > > > diff --git a/fs/autofs4/waitq.c b/fs/autofs4/waitq.c
> > > > > index 9857543..4a9ad9b 100644
> > > > > --- a/fs/autofs4/waitq.c
> > > > > +++ b/fs/autofs4/waitq.c
> > > > > @@ -141,8 +141,8 @@ static void autofs4_notify_daemon(struct
> > > > >   packet->ino = wq->ino;
> > > > >   packet->uid = wq->uid;
> > > > >   packet->gid = wq->gid;
> > > > > - packet->pid = wq->pid;
> > > > > - packet->tgid = wq->tgid;
> > > > > + packet->pid = pid_nr(wq->pid);
> > > > > + packet->tgid = pid_nr(wq->tgid);
> > > > >   break;
> > > > >
> > > > I'm assuming we build the packet in the process context of the
> > > > daemon we are sending it to. If not we have a problem here.
> > > >
> > > Yes this is data being sent to a userspace daemon (Ian pls
> > > correct me if I'm wrong) so the pid_nr is the only thing we can
> > > send.
> > >
> > Agreed. The question is are we in the user space daemon's process
> > when we generate the pid_nr. Or do we stuff this in some kind of
> > socket, and the socket switch locations of the packet.
> >
> Basically I'm just trying to be certain we are calling pid_nr in the
> proper context. Otherwise we could get the wrong pid when we have
> multiple pid namespaces in play.
>
We need to know what the userspace daemon being written to is doing
with autofs_ptype_{missing,expire}_{in,}direct() messages.
>
At the moment autofs only uses the packet->pid for logging purposes.
This solves an age old problem of not knowing who is causing mount
requests.
>
probably I'm wrong, but that sounds like the packet->pid

> is supposed to be the pid of the process _causing_ the mount,
> not the user space daemon communicating with the kernel ...

That's right. The only place it's used atm is to log the pid of
processes that are triggering mounts.

>
> > I'm not aware of any other applications that use version 5 yet but
> > that of course could change. So we can't really know what will be done
> > with these ids at some point in the future.
> >
> > > If I understand correctly, the pid being sent is of a process which
> > > tried to automount some directory. The message is being sent to the
> > > autofs daemon, which should be running in the root pid namespace.
> >
> > Yes, but it could be the autofs daemon itself in the expire case.
> >
> > Usually it doesn't make sense to run an automounting application as
> > other than "root" but I'm not familiar with other possible userspace
> > applications. Perhaps User Mode Linux could be an issue?
> >
> > >
> > > So as it is, the pid_nr(wq->pid) should be done under the init
> > > pid_namespace, since it's a kthread. So as long as the userspace
> > > automount daemon is started in the root pid namespace, the pid it
> > > gets will be the right one.
> > >
> > > Ian, does what I'm saying make sense, or am I wrong about how things
> > > work for autofs?
> >
> > Yep. That's the way it is.
>
> assumed we allow auto mounter mounts inside a context
> (I see no immediate reason not to do that) we want to
> know the name/pid space the userspace daemon is running
> in as well as the name/pid space of the trigger task
>
> > > thanks,
> > > -serge
> > >
> > > PS
> > > Note that if I'm right, but some machine starts autofs in a child
> > > pid_namespace, the pid_nr() the way I have it is wrong. I'm not sure in
> > > that case how we go about fixing that. Somehow we need to store the
> > > autofs userspace daemon's pid namespace pointer to help us find the
> > > proper pid_nr.
> >
> > In order for any user space application to use the module it must mount

> > the autofs file system, passing a file handle for the pipe to use for
> > communication. This must always be done. Can we grab the process pid
> > namespace at that time and store it in the superblock info structure?
>
> probably yes, but if my assumption above is correct, that
> isn't necessarily the pid/space for the trigger process
> (although it makes sense that it _should_ be)

btw please don't be confused by my use of "user space application". In my comments I'm talking about applications that are similar in function to autofs itself and not processes that might trigger mounts. For example the "autodir" package uses the autofs filesystem to provide dynamic home directory creation. The point being that we can't know what an application may do with the info in a request packet so we need to be sure we cover all the cases.

Ian

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by Ian Kent on Thu, 22 Mar 2007 15:06:20 GMT
View Forum Message <> Reply to Message

On Thu, 2007-03-22 at 08:31 -0500, Serge E. Hallyn wrote:
> > >
> > > From: "Serge E. Hallyn" <serue@us.ibm.com>
> > > Subject: [PATCH] autofs: prevent pid wraparound in waitqs
> > >
> > > Instead of storing pid numbers for waitqs, store references
> > > to struct pids. Also store a reference to the mounter's pid
> > > namespace in the autofs4 sb info so that pid numbers for
> > > mount miss and expiry msgs can send the pid# in the mounter's
> > > pidns.
> >
> > I think this amounts to what I suggested in my previous replies.
> > Hopefully my comments are enough to clear up any questions on
> > correctness of this approach.
> >
> > Sorry to be a pain but I'm having a little trouble reviewing the patch
> > because I'm not clear on where the code to handle the automount process
> > group (so called oz_pgrp), from the first patch, fits in with this.
>

> It also has pidspace infrastructure code in it, so I think we will just
> hold off on this until we have that infrastructure merged into the
> pidspace code and into -mm. Then we can send you a single, more concise
> patch.

OK great.

I'll have a close look through the code and it's related calls when I
get hold of a complete patch. That will be much easier and mistakes will
be less likely.

Ian

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by ebiederm on Thu, 22 Mar 2007 15:22:37 GMT
View Forum Message <> Reply to Message

Ian Kent <raven@themaw.net> writes:

> On Wed, 2007-03-21 at 15:58 -0500, Serge E. Hallyn wrote:

>> PS
>> Note that if I'm right, but some machine starts autofs in a child
>> pid_namespace, the pid_nr() the way I have it is wrong. I'm not sure in
>> that case how we go about fixing that. Somehow we need to store the
>> autofs userspace daemon's pid namespace pointer to help us find the
>> proper pid_nr.
>
> In order for any user space application to use the module it must mount
> the autofs file system, passing a file handle for the pipe to use for
> communication. This must always be done. Can we grab the process pid
> namespace at that time and store it in the superblock info structure?

I think this sounds like what Serge's did with his last patch, and
it sounds like the only reasonable thing we can do in this situation.

I had missed the fact the pipe is passed in at mount time.

> How does this affect getting ids for waitq request packets of other user
> space processes triggering mounts? I'm guessing that they would need to
> belong to the appropriate namespace for this mechanism to funtion

> correctly.

Not really. As long as the user space applications that trigger the mount
is in the pid namespace where the mount was performed or in a child of that
pid namespace it will has a process id that you can see.

If it happens that someone is entirely too creative and the process that
triggered the mount is in an entirely different pid namespace that does
not map __pid_nr will return 0. Not perfect but at least it is well defined.
And a reasonable thing to return for a process you cannot see.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by ebiederm on Thu, 22 Mar 2007 15:29:37 GMT
View Forum Message <> Reply to Message

Ian Kent <raven@themaw.net> writes:

> btw please don't be confused by my use of "user space application". In
> my comments I'm talking about applications that are similar in function
> to autofs itself and not processes that might trigger mounts. For
> example the "autodir" package uses the autofs filesystem to provide
> dynamic home directory creation. The point being that we can't know what
> an application may do with the info in a request packet so we need to be
> sure we cover all the cases.

A sentiment I share, and the reason why it is taking so long to come
up with a patch for autofs4 that passes review.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 2/2] Replace pid_t in autofs with struct pid reference
Posted by serue on Thu, 22 Mar 2007 18:07:07 GMT
View Forum Message <> Reply to Message

Quoting Ian Kent (raven@themaw.net):

> How does this affect getting ids for waitq request packets of other user
> space processes triggering mounts? I'm guessing that they would need to
> belong to the appropriate namespace for this mechanism to funtion
> correctly.

A feature of the pid namespaces is that any process in a cloned
namespace still has a valid pid in all ancestore pid namespaces. So
when a process triggers a mount, either it is in the same or a decendent
pid namespace as the process which did the mounting, in which case the
pid sent to the mounter is correct; or it is in some other namespace,
and it will get '0'. The latter shouldn't happen in a proper setup, and
should be safe to ignore in an improper setup.

For instance, so long as any clone(CLONE_NEWPID) is always done along
with a CLONE_NEWNS, and process in the child namespace which mounts an
autofs instance, processes in the parent pid namespace won't trigger
automounts. But if somehow they did (i.e. with shared submounts or by
not doing CLONE_NEWNS), pid 0 will be reported.

thanks,
-serge

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers