
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);
```

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