Subject: [PATCH v3] SUNRPC: set desired file system root before connecting local transports
Posted by Stanislav Kinsbursky on Mon, 08 Oct 2012 10:55:45 GMT
View Forum Message <> Reply to Message

Today, there is a problem in connecting of local SUNRPC thansports. These transports uses UNIX sockets and connection itself is done by rpciod workqueue.
But all local transports are connecting in rpciod context. I.e. UNIX sockets lookup is done in context of process file system root.
This works nice until we will try to mount NFS from process with other root - for example in container. This container can have it's own (nested) root and rcpbind process, listening on it's own unix sockets. But NFS mount attempt in this container will register new service (Lockd for example) in global rpcbind - not containers's one.
This patch solves the problem by switching rpciod kernel thread's file system root to the right one (stored on transport) while connecting of local transports.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 net/sunrpc/xprtsock.c | 52 ++--
 1 files changed, 50 insertions(+), 2 deletions(-)

diff --git a/net/sunrpc/xprtsock.c b/net/sunrpc/xprtsock.c
index aaaadfb..ecbced1 100644
--- a/net/sunrpc/xprtsock.c
+++ b/net/sunrpc/xprtsock.c
@@ -37,6 +37,7 @@
 #include <linux/sunrpc/svcsock.h>
 #include <linux/sunrpc/xprtsock.h>
 #include <linux/file.h>
+#include <linux/fs_struct.h>
 #ifdef CONFIG_SUNRPC_BACKCHANNEL
 #include <linux/sunrpc/bc_xprt.h>
 #endif
@@ -255,6 +256,11 @@ struct sock_xprt {
   void   (*old_state_change)(struct sock *);
   void   (*old_write_space)(struct sock *);
   void   (*old_error_report)(struct sock *);
+
+ /*
+ * Saved transport creator root. Required for local transports only.
+ */
+ struct path root;
 };

 /*

```
@@ -1876,6 +1882,30 @@ static int xs_local_finish_connecting(struct rpc_xprt *xprt,
  return kernel_connect(sock, xs_addr(xprt), xprt->addrlen, 0);
 }

+/*
+ * __xs_local_swap_root - swap current root to tranport's root helper.
+ * @new_root - path to set to current fs->root.
+ * @old_root - optinal paoinet to save current root to
+ *
+ * This routine is requieed to connecting unix sockets to absolute path in
+ * proper root environment.
+ * Note: no path_get() will be called. I.e. caller have to hold reference to
+ * new_root.
+ */
+static void __xs_local_swap_root(struct path *new_root, struct path *old_root)
+{
+ struct fs_struct *fs = current->fs;
+
+ spin_lock(&fs->lock);
+ write_seqcount_begin(&fs->seq);
+ if (old_root)
+  *old_root = fs->root;
+ fs->root = *new_root;
+ write_seqcount_end(&fs->seq);
+ spin_unlock(&fs->lock);
+}
+
+
 /**
  * xs_local_setup_socket - create AF_LOCAL socket, connect to a local endpoint
  * @xprt: RPC transport to connect
@@ -1891,6 +1921,7 @@ static void xs_local_setup_socket(struct work_struct *work)
  struct rpc_xprt *xprt = &transport->xprt;
  struct socket *sock;
  int status = -EIO;
+ struct path root;

  current->flags |= PF_FSTRANS;

@@ -1907,7 +1938,10 @@ static void xs_local_setup_socket(struct work_struct *work)
  dprintk("RPC:       worker connecting xprt %p via AF_LOCAL to %s\n",
    xprt, xprt->address_strings[RPC_DISPLAY_ADDR]);

+ __xs_local_swap_root(&transport->root, &root);
  status = xs_local_finish_connecting(xprt, sock);
+ __xs_local_swap_root(&root, NULL);
+
  switch (status) {
```

```
  case 0:
   dprintk("RPC:       xprt %p connected to %s\n",
@@ -2256,6 +2290,18 @@ static void xs_connect(struct rpc_task *task)
  }
 }

+static void xs_local_destroy(struct rpc_xprt *xprt)
+{
+ struct sock_xprt *transport = container_of(xprt, struct sock_xprt, xprt);
+ struct path root = transport->root;
+
+ dprintk("RPC:       xs_local_destroy xprt %p\n", xprt);
+
+ xs_destroy(xprt);
+
+ path_put(&root);
+}
+
 /**
  * xs_local_print_stats - display AF_LOCAL socket-specifc stats
  * @xprt: rpc_xprt struct containing statistics
@@ -2475,7 +2521,7 @@ static struct rpc_xprt_ops xs_local_ops = {
  .send_request  = xs_local_send_request,
  .set_retrans_timeout = xprt_set_retrans_timeout_def,
  .close   = xs_close,
- .destroy  = xs_destroy,
+ .destroy  = xs_local_destroy,
  .print_stats  = xs_local_print_stats,
 };

@@ -2654,8 +2700,10 @@ static struct rpc_xprt *xs_setup_local(struct xprt_create *args)
  dprintk("RPC:       set up xprt to %s via AF_LOCAL\n",
    xprt->address_strings[RPC_DISPLAY_ADDR]);

- if (try_module_get(THIS_MODULE))
+ if (try_module_get(THIS_MODULE)) {
+  get_fs_root(current->fs, &transport->root);
   return xprt;
+ }
  ret = ERR_PTR(-EINVAL);
 out_err:
  xprt_free(xprt);
```

Subject: Re: [PATCH v3] SUNRPC: set desired file system root before connecting local transports
Posted by bfields on Tue, 09 Oct 2012 19:35:06 GMT

Cc'ing Eric since I seem to recall he suggested doing it this way?

Seems OK to me, but maybe that swap_root should be in common code? (Or
maybe we could use set_fs_root()?)

I'm assuming it's up to Trond to take this.--b.

On Mon, Oct 08, 2012 at 02:56:32PM +0400, Stanislav Kinsbursky wrote:
> Today, there is a problem in connecting of local SUNRPC thansports. These
> transports uses UNIX sockets and connection itself is done by rpciod
> workqueue.
> But all local transports are connecting in rpciod context. I.e. UNIX sockets
> lookup is done in context of process file system root.
> This works nice until we will try to mount NFS from process with other root -
> for example in container. This container can have it's own (nested) root and
> rcpbind process, listening on it's own unix sockets. But NFS mount attempt in
> this container will register new service (Lockd for example) in global rpcbind
> - not containers's one.
> This patch solves the problem by switching rpciod kernel thread's file system
> root to the right one (stored on transport) while connecting of local
> transports.
>
> Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
> ---
> net/sunrpc/xprtsock.c | 52 ++--
> 1 files changed, 50 insertions(+), 2 deletions(-)
>
> diff --git a/net/sunrpc/xprtsock.c b/net/sunrpc/xprtsock.c
> index aaaadfb..ecbced1 100644
> --- a/net/sunrpc/xprtsock.c
> +++ b/net/sunrpc/xprtsock.c
> @@ -37,6 +37,7 @@
> #include <linux/sunrpc/svcsock.h>
> #include <linux/sunrpc/xprtsock.h>
> #include <linux/file.h>
> +#include <linux/fs_struct.h>
> #ifdef CONFIG_SUNRPC_BACKCHANNEL
> #include <linux/sunrpc/bc_xprt.h>
> #endif
> @@ -255,6 +256,11 @@ struct sock_xprt {
> void (*old_state_change)(struct sock *);
> void (*old_write_space)(struct sock *);
> void (*old_error_report)(struct sock *);
> +
> + /*
> + * Saved transport creator root. Required for local transports only.
> + */

```
> + struct path  root;
>  };
>
>  /*
> @@ -1876,6 +1882,30 @@ static int xs_local_finish_connecting(struct rpc_xprt *xprt,
>   return kernel_connect(sock, xs_addr(xprt), xprt->addrlen, 0);
>  }
>
> +/*
> + * __xs_local_swap_root - swap current root to tranport's root helper.
> + * @new_root - path to set to current fs->root.
> + * @old_root - optinal paoinet to save current root to
> + *
> + * This routine is requieed to connecting unix sockets to absolute path in
> + * proper root environment.
> + * Note: no path_get() will be called. I.e. caller have to hold reference to
> + * new_root.
> + */
> +static void __xs_local_swap_root(struct path *new_root, struct path *old_root)
> +{
> + struct fs_struct *fs = current->fs;
> +
> + spin_lock(&fs->lock);
> + write_seqcount_begin(&fs->seq);
> + if (old_root)
> +  *old_root = fs->root;
> + fs->root = *new_root;
> + write_seqcount_end(&fs->seq);
> + spin_unlock(&fs->lock);
> +}
> +
> +
>  /**
>   * xs_local_setup_socket - create AF_LOCAL socket, connect to a local endpoint
>   * @xprt: RPC transport to connect
> @@ -1891,6 +1921,7 @@ static void xs_local_setup_socket(struct work_struct *work)
>   struct rpc_xprt *xprt = &transport->xprt;
>   struct socket *sock;
>   int status = -EIO;
> + struct path root;
>
>   current->flags |= PF_FSTRANS;
>
> @@ -1907,7 +1938,10 @@ static void xs_local_setup_socket(struct work_struct *work)
>   dprintk("RPC:       worker connecting xprt %p via AF_LOCAL to %s\n",
>     xprt, xprt->address_strings[RPC_DISPLAY_ADDR]);
>
> + __xs_local_swap_root(&transport->root, &root);
```

```
>   status = xs_local_finish_connecting(xprt, sock);
> + __xs_local_swap_root(&root, NULL);
> +
>   switch (status) {
>   case 0:
>    dprintk("RPC:       xprt %p connected to %s\n",
> @@ -2256,6 +2290,18 @@ static void xs_connect(struct rpc_task *task)
>   }
>  }
>
> +static void xs_local_destroy(struct rpc_xprt *xprt)
> +{
> + struct sock_xprt *transport = container_of(xprt, struct sock_xprt, xprt);
> + struct path root = transport->root;
> +
> + dprintk("RPC:       xs_local_destroy xprt %p\n", xprt);
> +
> + xs_destroy(xprt);
> +
> + path_put(&root);
> +}
> +
>  /**
>   * xs_local_print_stats - display AF_LOCAL socket-specifc stats
>   * @xprt: rpc_xprt struct containing statistics
> @@ -2475,7 +2521,7 @@ static struct rpc_xprt_ops xs_local_ops = {
>   .send_request  = xs_local_send_request,
>   .set_retrans_timeout = xprt_set_retrans_timeout_def,
>   .close   = xs_close,
> - .destroy  = xs_destroy,
> + .destroy  = xs_local_destroy,
>   .print_stats  = xs_local_print_stats,
>  };
>
> @@ -2654,8 +2700,10 @@ static struct rpc_xprt *xs_setup_local(struct xprt_create *args)
>   dprintk("RPC:       set up xprt to %s via AF_LOCAL\n",
>     xprt->address_strings[RPC_DISPLAY_ADDR]);
>
> - if (try_module_get(THIS_MODULE))
> + if (try_module_get(THIS_MODULE)) {
> +  get_fs_root(current->fs, &transport->root);
>    return xprt;
> + }
>   ret = ERR_PTR(-EINVAL);
>  out_err:
>   xprt_free(xprt);
>
```

Subject: Re: [PATCH v3] SUNRPC: set desired file system root before connecting local transports
Posted by Myklebust, Trond on Tue, 09 Oct 2012 19:49:46 GMT
View Forum Message <> Reply to Message

On Tue, 2012-10-09 at 15:35 -0400, J. Bruce Fields wrote:
> Cc'ing Eric since I seem to recall he suggested doing it this way?
>
> Seems OK to me, but maybe that swap_root should be in common code? (Or
> maybe we could use set_fs_root()?)
>
> I'm assuming it's up to Trond to take this.--b.

I'm reluctant to do that at this time since the original proposal was precisely that of export set_fs_root() and using it around the AF_LOCAL socket bind. That proposal was NACKed by Al Viro.

If Al is OK with the idea of us creating a private version of set_fs_root, then I'd like to see an official Acked-by: that we can append to this commit.

Cheers
Trond

> On Mon, Oct 08, 2012 at 02:56:32PM +0400, Stanislav Kinsbursky wrote:
> > Today, there is a problem in connecting of local SUNRPC thansports. These
> > transports uses UNIX sockets and connection itself is done by rpciod
> > workqueue.
> > But all local transports are connecting in rpciod context. I.e. UNIX sockets
> > lookup is done in context of process file system root.
> > This works nice until we will try to mount NFS from process with other root -
> > for example in container. This container can have it's own (nested) root and
> > rcpbind process, listening on it's own unix sockets. But NFS mount attempt in
> > this container will register new service (Lockd for example) in global rpcbind
> > - not containers's one.
> > This patch solves the problem by switching rpciod kernel thread's file system
> > root to the right one (stored on transport) while connecting of local
> > transports.
> >
> > Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
> > ---
> > net/sunrpc/xprtsock.c | 52 ++--
> > 1 files changed, 50 insertions(+), 2 deletions(-)
> >
> > diff --git a/net/sunrpc/xprtsock.c b/net/sunrpc/xprtsock.c
> > index aaaadfb..ecbced1 100644
> > --- a/net/sunrpc/xprtsock.c
> > +++ b/net/sunrpc/xprtsock.c
> > @@ -37,6 +37,7 @@

```
> >  #include <linux/sunrpc/svcsock.h>
> >  #include <linux/sunrpc/xprtsock.h>
> >  #include <linux/file.h>
> > +#include <linux/fs_struct.h>
> >  #ifdef CONFIG_SUNRPC_BACKCHANNEL
> >  #include <linux/sunrpc/bc_xprt.h>
> >  #endif
> > @@ -255,6 +256,11 @@ struct sock_xprt {
> >   void   (*old_state_change)(struct sock *);
> >   void   (*old_write_space)(struct sock *);
> >   void   (*old_error_report)(struct sock *);
> > +
> > + /*
> > +  * Saved transport creator root. Required for local transports only.
> > +  */
> > + struct path  root;
> >  };
> >
> >  /*
> > @@ -1876,6 +1882,30 @@ static int xs_local_finish_connecting(struct rpc_xprt *xprt,
> >   return kernel_connect(sock, xs_addr(xprt), xprt->addrlen, 0);
> >  }
> >
> > +/*
> > + * __xs_local_swap_root - swap current root to tranport's root helper.
> > + * @new_root - path to set to current fs->root.
> > + * @old_root - optinal paoinet to save current root to
> > + *
> > + * This routine is requieed to connecting unix sockets to absolute path in
> > + * proper root environment.
> > + * Note: no path_get() will be called. I.e. caller have to hold reference to
> > + * new_root.
> > + */
> > +static void __xs_local_swap_root(struct path *new_root, struct path *old_root)
> > +{
> > + struct fs_struct *fs = current->fs;
> > +
> > + spin_lock(&fs->lock);
> > + write_seqcount_begin(&fs->seq);
> > + if (old_root)
> > +  *old_root = fs->root;
> > + fs->root = *new_root;
> > + write_seqcount_end(&fs->seq);
> > + spin_unlock(&fs->lock);
> > +}
> > +
> > +
> >  /**
```

```
> >   * xs_local_setup_socket - create AF_LOCAL socket, connect to a local endpoint
> >   * @xprt: RPC transport to connect
> > @@ -1891,6 +1921,7 @@ static void xs_local_setup_socket(struct work_struct *work)
> >   struct rpc_xprt *xprt = &transport->xprt;
> >   struct socket *sock;
> >   int status = -EIO;
> > + struct path root;
> >
> >   current->flags |= PF_FSTRANS;
> >
> > @@ -1907,7 +1938,10 @@ static void xs_local_setup_socket(struct work_struct *work)
> >   dprintk("RPC:       worker connecting xprt %p via AF_LOCAL to %s\n",
> >     xprt, xprt->address_strings[RPC_DISPLAY_ADDR]);
> >
> > + __xs_local_swap_root(&transport->root, &root);
> >   status = xs_local_finish_connecting(xprt, sock);
> > + __xs_local_swap_root(&root, NULL);
> > +
> >   switch (status) {
> >   case 0:
> >    dprintk("RPC:       xprt %p connected to %s\n",
> > @@ -2256,6 +2290,18 @@ static void xs_connect(struct rpc_task *task)
> >   }
> >  }
> >
> > +static void xs_local_destroy(struct rpc_xprt *xprt)
> > +{
> > + struct sock_xprt *transport = container_of(xprt, struct sock_xprt, xprt);
> > + struct path root = transport->root;
> > +
> > + dprintk("RPC:       xs_local_destroy xprt %p\n", xprt);
> > +
> > + xs_destroy(xprt);
> > +
> > + path_put(&root);
> > +}
> > +
> >  /**
> >   * xs_local_print_stats - display AF_LOCAL socket-specifc stats
> >   * @xprt: rpc_xprt struct containing statistics
> > @@ -2475,7 +2521,7 @@ static struct rpc_xprt_ops xs_local_ops = {
> >   .send_request  = xs_local_send_request,
> >   .set_retrans_timeout = xprt_set_retrans_timeout_def,
> >   .close   = xs_close,
> > - .destroy  = xs_destroy,
> > + .destroy  = xs_local_destroy,
> >   .print_stats  = xs_local_print_stats,
> >  };
```

> >
> > @@ -2654,8 +2700,10 @@ static struct rpc_xprt *xs_setup_local(struct xprt_create *args)
> > dprintk("RPC: set up xprt to %s via AF_LOCAL\n",
> > xprt->address_strings[RPC_DISPLAY_ADDR]);
> >
> > - if (try_module_get(THIS_MODULE))
> > + if (try_module_get(THIS_MODULE)) {
> > + get_fs_root(current->fs, &transport->root);
> > return xprt;
> > + }
> > ret = ERR_PTR(-EINVAL);
> > out_err:
> > xprt_free(xprt);
> >

--
Trond Myklebust
Linux NFS client maintainer

NetApp
Trond.Myklebust@netapp.com
www.netapp.com

Subject: Re: [PATCH v3] SUNRPC: set desired file system root before connecting local transports
Posted by ebiederm on Tue, 09 Oct 2012 20:20:48 GMT
View Forum Message <> Reply to Message

"Myklebust, Trond" <Trond.Myklebust@netapp.com> writes:

> On Tue, 2012-10-09 at 15:35 -0400, J. Bruce Fields wrote:
>> Cc'ing Eric since I seem to recall he suggested doing it this way?

Yes. On second look setting fs->root won't work. We need to change fs.
The problem is that by default all kernel threads share fs so changing
fs->root will have non-local consequences.

I very much believe we want if at all possible to perform a local
modification.

Changing fs isn't all that different from what devtmpfs is doing.

>> Seems OK to me, but maybe that swap_root should be in common code? (Or
>> maybe we could use set_fs_root()?)
>>
>> I'm assuming it's up to Trond to take this.--b.
>

> I'm reluctant to do that at this time since the original proposal was
> precisely that of export set_fs_root() and using it around the AF_LOCAL
> socket bind. That proposal was NACKed by Al Viro.

> If Al is OK with the idea of us creating a private version of
> set_fs_root, then I'd like to see an official Acked-by: that we can
> append to this commit.

Eric

Subject: Re: [PATCH v3] SUNRPC: set desired file system root before connecting local transports
Posted by bfields on Tue, 09 Oct 2012 22:31:39 GMT
View Forum Message <> Reply to Message

On Tue, Oct 09, 2012 at 01:20:48PM -0700, Eric W. Biederman wrote:
> "Myklebust, Trond" <Trond.Myklebust@netapp.com> writes:
>
> > On Tue, 2012-10-09 at 15:35 -0400, J. Bruce Fields wrote:
> >> Cc'ing Eric since I seem to recall he suggested doing it this way?
>
> Yes. On second look setting fs->root won't work. We need to change fs.
> The problem is that by default all kernel threads share fs so changing
> fs->root will have non-local consequences.

Oh, huh. And we can't "unshare" it somehow?

Or, previously you suggested:

 - introduce sockaddr_fd that can be applied to AF_UNIX sockets,
   and teach unix_bind and unix_connect how to deal with a second
   type of sockaddr, AT_FD:
   struct sockaddr_fd { short fd_family; short pad; int fd; }

 - introduce sockaddr_unix_at that takes a directory file
   descriptor as well as a unix path, and teach unix_bind and
   unix_connect to deal with a second sockaddr type, AF_UNIX_AT:
   struct sockaddr_unix_at { short family; short pad; int dfd; char path[102]; }

Any other options?

> I very much believe we want if at all possible to perform a local
> modification.
>
> Changing fs isn't all that different from what devtmpfs is doing.

Sorry, I don't know much about devtmpfs, are you suggesting it as a

model? What exactly should we look at?

--b.

Subject: Re: [PATCH v3] SUNRPC: set desired file system root before connecting local transports
Posted by ebiederm on Tue, 09 Oct 2012 22:47:42 GMT
View Forum Message <> Reply to Message

"J. Bruce Fields" <bfields@fieldses.org> writes:

> On Tue, Oct 09, 2012 at 01:20:48PM -0700, Eric W. Biederman wrote:
>> "Myklebust, Trond" <Trond.Myklebust@netapp.com> writes:
>>
>> > On Tue, 2012-10-09 at 15:35 -0400, J. Bruce Fields wrote:
>> >> Cc'ing Eric since I seem to recall he suggested doing it this way?
>>
>> Yes. On second look setting fs->root won't work. We need to change fs.
>> The problem is that by default all kernel threads share fs so changing
>> fs->root will have non-local consequences.
>
> Oh, huh. And we can't "unshare" it somehow?

I don't fully understand how nfs uses kernel threads and work queues.
My general understanding is work queues reuse their kernel threads
between different users. So it is mostly a don't pollute your
environment thing. If there was a dedicated kernel thread for each
environment this would be trivial.

What I was suggesting here is changing task->fs instead of
task->fs.root. That should just require task_lock().

> Or, previously you suggested:
>
> - introduce sockaddr_fd that can be applied to AF_UNIX sockets,
>   and teach unix_bind and unix_connect how to deal with a second
>   type of sockaddr, AT_FD:
>   struct sockaddr_fd { short fd_family; short pad; int fd; }
>
> - introduce sockaddr_unix_at that takes a directory file
>   descriptor as well as a unix path, and teach unix_bind and
>   unix_connect to deal with a second sockaddr type, AF_UNIX_AT:
>   struct sockaddr_unix_at { short family; short pad; int dfd; char path[102]; }
>
> Any other options?

I am still half hoping we don't have to change the userspace API/ABI.

There is sanity checking on that path that no one seems interested in to solve this problem.

This is a weird issue as we are dealing with both the vfs and the networking stack. Fundamentally we need to change task->fs.root or we need to capitialize on the openat functionality in the kernel, so that we don't create mountains of special cases to support this.

I think swapping task->fs instead of task->fs.root is effecitely the same complexity.

>> I very much believe we want if at all possible to perform a local
>> modification.
>>
>> Changing fs isn't all that different from what devtmpfs is doing.
>
> Sorry, I don't know much about devtmpfs, are you suggesting it as a
> model? What exactly should we look at?

Roughly all I meant was that devtmpsfsd is a kernel thread that runs with an unshared fs struct. Although I admit devtmpfsd is for all practical purposes a userspace daemon that just happens to run in kernel space.

Eric

Subject: Re: [PATCH v3] SUNRPC: set desired file system root before connecting local transports
Posted by bfields on Wed, 10 Oct 2012 01:23:36 GMT
View Forum Message <> Reply to Message

On Tue, Oct 09, 2012 at 03:47:42PM -0700, Eric W. Biederman wrote:
> "J. Bruce Fields" <bfields@fieldses.org> writes:
>
> > On Tue, Oct 09, 2012 at 01:20:48PM -0700, Eric W. Biederman wrote:
> >> "Myklebust, Trond" <Trond.Myklebust@netapp.com> writes:
> >>
> >> > On Tue, 2012-10-09 at 15:35 -0400, J. Bruce Fields wrote:
> >> >> Cc'ing Eric since I seem to recall he suggested doing it this way?
> >>
> >> Yes. On second look setting fs->root won't work. We need to change fs.
> >> The problem is that by default all kernel threads share fs so changing
> >> fs->root will have non-local consequences.
> >
> > Oh, huh. And we can't "unshare" it somehow?
>
> I don't fully understand how nfs uses kernel threads and work queues.

> My general understanding is work queues reuse their kernel threads
> between different users. So it is mostly a don't pollute your
> environment thing. If there was a dedicated kernel thread for each
> environment this would be trivial.
>
> What I was suggesting here is changing task->fs instead of
> task->fs.root. That should just require task_lock().

Oh, OK, got it--if that works, great.

> > Sorry, I don't know much about devtmpfs, are you suggesting it as a
> > model? What exactly should we look at?
>
> Roughly all I meant was that devtmpsfsd is a kernel thread that runs
> with an unshared fs struct. Although I admit devtmpfsd is for all
> practical purposes a userspace daemon that just happens to run in kernel
> space.

Thanks for the explanation.

--b.

Subject: Re: [PATCH v3] SUNRPC: set desired file system root before connecting local transports
Posted by ebiederm on Wed, 10 Oct 2012 02:00:35 GMT
View Forum Message <> Reply to Message

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

> "J. Bruce Fields" <bfields@fieldses.org> writes:
>
>> On Tue, Oct 09, 2012 at 01:20:48PM -0700, Eric W. Biederman wrote:
>>> "Myklebust, Trond" <Trond.Myklebust@netapp.com> writes:
>>>
>>> > On Tue, 2012-10-09 at 15:35 -0400, J. Bruce Fields wrote:
>>> >> Cc'ing Eric since I seem to recall he suggested doing it this way?
>>>
>>> Yes. On second look setting fs->root won't work. We need to change fs.
>>> The problem is that by default all kernel threads share fs so changing
>>> fs->root will have non-local consequences.
>>
>> Oh, huh. And we can't "unshare" it somehow?
>
> I don't fully understand how nfs uses kernel threads and work queues.
> My general understanding is work queues reuse their kernel threads
> between different users. So it is mostly a don't pollute your
> environment thing. If there was a dedicated kernel thread for each

> environment this would be trivial.
>
> What I was suggesting here is changing task->fs instead of
> task->fs.root. That should just require task_lock().
>
>> Or, previously you suggested:
>>
>> - introduce sockaddr_fd that can be applied to AF_UNIX sockets,
>>   and teach unix_bind and unix_connect how to deal with a second
>>   type of sockaddr, AT_FD:
>>   struct sockaddr_fd { short fd_family; short pad; int fd; }
>>
>> - introduce sockaddr_unix_at that takes a directory file
>>   descriptor as well as a unix path, and teach unix_bind and
>>   unix_connect to deal with a second sockaddr type, AF_UNIX_AT:
>>   struct sockaddr_unix_at { short family; short pad; int dfd; char path[102]; }
>>
>> Any other options?
>
> I am still half hoping we don't have to change the userspace API/ABI.
> There is sanity checking on that path that no one seems interested in to
> solve this problem.

There is a good option if we are up to userspace ABI extensions.

Implement open(2) on unix domain sockets. Where open(2) would
essentially equal connect(2) on unix domain sockets.

With an open(2) implementation we could use file_open_path and the
implementation should be pretty straight forward and maintainable.
So implementing open(2) looks like a good alternative implementation
route.

Eric

Subject: Re: [PATCH v3] SUNRPC: set desired file system root before connecting local transports
Posted by Stanislav Kinsbursky on Wed, 10 Oct 2012 05:03:41 GMT
View Forum Message <> Reply to Message

10.10.2012 02:47, Eric W. Biederman
> "J. Bruce Fields" <bfields@fieldses.org> writes:
>
>> On Tue, Oct 09, 2012 at 01:20:48PM -0700, Eric W. Biederman wrote:
>>> "Myklebust, Trond" <Trond.Myklebust@netapp.com> writes:
>>>
>>>> On Tue, 2012-10-09 at 15:35 -0400, J. Bruce Fields wrote:

>>>>> Cc'ing Eric since I seem to recall he suggested doing it this way?
>>> Yes. On second look setting fs->root won't work. We need to change fs.
>>> The problem is that by default all kernel threads share fs so changing
>>> fs->root will have non-local consequences.
>> Oh, huh. And we can't "unshare" it somehow?
> I don't fully understand how nfs uses kernel threads and work queues.
> My general understanding is work queues reuse their kernel threads
> between different users. So it is mostly a don't pollute your
> environment thing. If there was a dedicated kernel thread for each
> environment this would be trivial.

One kernel thread per environment is exactly what we are trying to avoid
if possible.
And the reason why we don't want to do this is that it's really looks
like overkill.

> What I was suggesting here is changing task->fs instead of
> task->fs.root. That should just require task_lock().
>
>> Or, previously you suggested:
>>
>> - introduce sockaddr_fd that can be applied to AF_UNIX sockets,
>>   and teach unix_bind and unix_connect how to deal with a second
>>   type of sockaddr, AT_FD:
>>   struct sockaddr_fd { short fd_family; short pad; int fd; }
>>
>> - introduce sockaddr_unix_at that takes a directory file
>>   descriptor as well as a unix path, and teach unix_bind and
>>   unix_connect to deal with a second sockaddr type, AF_UNIX_AT:
>>   struct sockaddr_unix_at { short family; short pad; int dfd; char path[102]; }
>>
>> Any other options?
> I am still half hoping we don't have to change the userspace API/ABI.
> There is sanity checking on that path that no one seems interested in to
> solve this problem.
>
> This is a weird issue as we are dealing with both the vfs and the
> networking stack. Fundamentally we need to change task->fs.root or
> we need to capitialize on the openat functionality in the kernel, so
> that we don't create mountains of special cases to support this.
>
> I think swapping task->fs instead of task->fs.root is effecitely the
> same complexity.

Thanks for the idea. And for mentioning, that kernel threads shares fs
struct. I missed it.

>

>>> I very much believe we want if at all possible to perform a local
>>> modification.
>>>
>>> Changing fs isn't all that different from what devtmpfs is doing.
>> Sorry, I don't know much about devtmpfs, are you suggesting it as a
>> model? What exactly should we look at?
> Roughly all I meant was that devtmpsfsd is a kernel thread that runs
> with an unshared fs struct. Although I admit devtmpfsd is for all
> practical purposes a userspace daemon that just happens to run in kernel
> space.
>
> Eric
>
> --
> To unsubscribe from this list: send the line "unsubscribe linux-nfs" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html

---

Subject: Re: [PATCH v3] SUNRPC: set desired file system root before connecting local transports
Posted by Stanislav Kinsbursky on Wed, 10 Oct 2012 05:09:24 GMT
View Forum Message <> Reply to Message

10.10.2012 06:00, Eric W. Biederman
> ebiederm@xmission.com (Eric W. Biederman) writes:
>
>> "J. Bruce Fields" <bfields@fieldses.org> writes:
>>
>>> On Tue, Oct 09, 2012 at 01:20:48PM -0700, Eric W. Biederman wrote:
>>>> "Myklebust, Trond" <Trond.Myklebust@netapp.com> writes:
>>>>
>>>>> On Tue, 2012-10-09 at 15:35 -0400, J. Bruce Fields wrote:
>>>>>> Cc'ing Eric since I seem to recall he suggested doing it this way?
>>>> Yes. On second look setting fs->root won't work. We need to change fs.
>>>> The problem is that by default all kernel threads share fs so changing
>>>> fs->root will have non-local consequences.
>>> Oh, huh. And we can't "unshare" it somehow?
>> I don't fully understand how nfs uses kernel threads and work queues.
>> My general understanding is work queues reuse their kernel threads
>> between different users. So it is mostly a don't pollute your
>> environment thing. If there was a dedicated kernel thread for each
>> environment this would be trivial.
>>
>> What I was suggesting here is changing task->fs instead of
>> task->fs.root. That should just require task_lock().
>>
>>> Or, previously you suggested:

>>>
>>> - introduce sockaddr_fd that can be applied to AF_UNIX sockets,
>>>   and teach unix_bind and unix_connect how to deal with a second
>>>   type of sockaddr, AT_FD:
>>>   struct sockaddr_fd { short fd_family; short pad; int fd; }
>>>
>>> - introduce sockaddr_unix_at that takes a directory file
>>>   descriptor as well as a unix path, and teach unix_bind and
>>>   unix_connect to deal with a second sockaddr type, AF_UNIX_AT:
>>>   struct sockaddr_unix_at { short family; short pad; int dfd; char path[102]; }
>>>
>>> Any other options?
>> I am still half hoping we don't have to change the userspace API/ABI.
>> There is sanity checking on that path that no one seems interested in to
>> solve this problem.
> There is a good option if we are up to userspace ABI extensions.
>
> Implement open(2) on unix domain sockets. Where open(2) would
> essentially equal connect(2) on unix domain sockets.
>
> With an open(2) implementation we could use file_open_path and the
> implementation should be pretty straight forward and maintainable.
> So implementing open(2) looks like a good alternative implementation
> route.

This requires patching of vfs layer as well. I don't want to say, that
the idea is not good. But it requires much more time to implement and test.
And this patch addresses the problem, which exist already and would be
great to fix it as soon as possible.
So, probably, implementing open for unix sockets is the next and more
generic step.
Thanks.

> Eric
>
> --
> To unsubscribe from this list: send the line "unsubscribe linux-nfs" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html

---

Subject: Re: [PATCH v3] SUNRPC: set desired file system root before connecting local transports
Posted by Stanislav Kinsbursky on Wed, 10 Oct 2012 10:32:28 GMT
View Forum Message <> Reply to Message

10.10.2012 05:23, J. Bruce Fields
> On Tue, Oct 09, 2012 at 03:47:42PM -0700, Eric W. Biederman wrote:

>> "J. Bruce Fields" <bfields@fieldses.org> writes:
>>
>>> On Tue, Oct 09, 2012 at 01:20:48PM -0700, Eric W. Biederman wrote:
>>>> "Myklebust, Trond" <Trond.Myklebust@netapp.com> writes:
>>>>
>>>>> On Tue, 2012-10-09 at 15:35 -0400, J. Bruce Fields wrote:
>>>>>> Cc'ing Eric since I seem to recall he suggested doing it this way?
>>>>
>>>> Yes. On second look setting fs->root won't work. We need to change fs.
>>>> The problem is that by default all kernel threads share fs so changing
>>>> fs->root will have non-local consequences.
>>>
>>> Oh, huh. And we can't "unshare" it somehow?
>>
>> I don't fully understand how nfs uses kernel threads and work queues.
>> My general understanding is work queues reuse their kernel threads
>> between different users. So it is mostly a don't pollute your
>> environment thing. If there was a dedicated kernel thread for each
>> environment this would be trivial.
>>
>> What I was suggesting here is changing task->fs instead of
>> task->fs.root. That should just require task_lock().
>
> Oh, OK, got it--if that works, great.
>

The main problem with swapping fs struct is actually the same as in root
swapping. I.e. routines for copy fs_struct are not exported.
It could be done on place, but I don't think, that Al Viro would support such
implementation.
Trond?

>>> Sorry, I don't know much about devtmpfs, are you suggesting it as a
>>> model? What exactly should we look at?
>>
>> Roughly all I meant was that devtmpsfsd is a kernel thread that runs
>> with an unshared fs struct. Although I admit devtmpfsd is for all
>> practical purposes a userspace daemon that just happens to run in kernel
>> space.
>
> Thanks for the explanation.
>
> --b.
>

--
Best regards,

Subject: Re: [PATCH v3] SUNRPC: set desired file system root before connecting local transports
Posted by bfields on Fri, 26 Oct 2012 17:52:54 GMT
View Forum Message <> Reply to Message

On Wed, Oct 10, 2012 at 02:32:28PM +0400, Stanislav Kinsbursky wrote:
> 10.10.2012 05:23, J. Bruce Fields
> >On Tue, Oct 09, 2012 at 03:47:42PM -0700, Eric W. Biederman wrote:
> >>"J. Bruce Fields" <bfields@fieldses.org> writes:
> >>
> >>>On Tue, Oct 09, 2012 at 01:20:48PM -0700, Eric W. Biederman wrote:
> >>>>"Myklebust, Trond" <Trond.Myklebust@netapp.com> writes:
> >>>>
> >>>>>On Tue, 2012-10-09 at 15:35 -0400, J. Bruce Fields wrote:
> >>>>>>Cc'ing Eric since I seem to recall he suggested doing it this way?
> >>>>
> >>>>Yes. On second look setting fs->root won't work. We need to change fs.
> >>>>The problem is that by default all kernel threads share fs so changing
> >>>>fs->root will have non-local consequences.
> >>>
> >>>Oh, huh. And we can't "unshare" it somehow?
> >>
> >>I don't fully understand how nfs uses kernel threads and work queues.
> >>My general understanding is work queues reuse their kernel threads
> >>between different users. So it is mostly a don't pollute your
> >>environment thing. If there was a dedicated kernel thread for each
> >>environment this would be trivial.
> >>
> >>What I was suggesting here is changing task->fs instead of
> >>task->fs.root. That should just require task_lock().
> >
> >Oh, OK, got it--if that works, great.
> >
>
> The main problem with swapping fs struct is actually the same as in
> root swapping. I.e. routines for copy fs_struct are not exported.
> It could be done on place, but I don't think, that Al Viro would
> support such implementation.
> Trond?

It seems like we got stalled here.... Could you go ahead and try a
patch, and see what people think?

--b.