
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 transports. 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 rcpbind

> > - 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..ecbcd1 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 transport's root helper.
>> + * @new_root - path to set to current fs->root.
>> + * @old_root - optional pointer to save current root to
>> + *
>> + * This routine is required 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-specific 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
