Subject: [RFC PATCH] SUNRPC: connect local transports synchronously
Posted by Stanislav Kinsbursky on Thu, 16 Feb 2012 15:06:25 GMT
View Forum Message <> Reply to Message

Local tranports uses UNIX sockets and connecting of these sockets is done in
context of file system namespace (i.e. task file system root).
Currenly, all sockets connect operations are performed by rpciod work queue,
which actually means, that any service will be registered in the same rpcbind
instance regardless to process file system root.
This is not containers, which usually have it's own nested root. There are 2
approaches, how to solve the problem. First one is to store proper root in
tranport and switch to it in rpciod workqueue function for connect operations.
But this looks ugly. The second one is to connect to unix sockets
synchronously. This aptch implements the last one.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

---
 net/sunrpc/xprtsock.c | 22 ++++++++++++++++++++-
 1 files changed, 21 insertions(+), 1 deletions(-)

diff --git a/net/sunrpc/xprtsock.c b/net/sunrpc/xprtsock.c
index 55472c4..365cd6d 100644
--- a/net/sunrpc/xprtsock.c
+++ b/net/sunrpc/xprtsock.c
@@ -2177,6 +2177,26 @@ out:
 }

 /**
+ * xs_local_connect - connect a local (unix) socket to a remote endpoint
+ * @task: address of RPC task that manages state of connect request
+ *
+ * We have to connect unix sockets synchronously. Otherwise this connection
+ * will be done in file system context of rpciod queue, which is not suitable
+ * for processes with other root (changed root is a usual part of environment
+ * for containers).
+ */
+
+static void xs_local_connect(struct rpc_task *task)
+{
+ struct rpc_xprt *xprt = task->tk_xprt;
+ struct sock_xprt *transport = container_of(xprt, struct sock_xprt, xprt);
+ struct work_struct *work = &transport->connect_worker.work;
+
+ dprintk("RPC: xs_local_connect xprt %p\n", xprt);
+ work->func(work);
+}
+

```
+/**
  * xs_connect - connect a socket to a remote endpoint
  * @task: address of RPC task that manages state of connect request
  *
@@ -2414,7 +2434,7 @@ static struct rpc_xprt_ops xs_local_ops = {
  .release_xprt  = xs_tcp_release_xprt,
  .rpcbind  = xs_local_rpcbind,
  .set_port  = xs_local_set_port,
- .connect  = xs_connect,
+ .connect  = xs_local_connect,
  .buf_alloc  = rpc_malloc,
  .buf_free  = rpc_free,
  .send_request  = xs_local_send_request,
```

Subject: Re: [RFC PATCH] SUNRPC: connect local transports synchronously
Posted by Myklebust, Trond on Thu, 16 Feb 2012 15:13:53 GMT
View Forum Message <> Reply to Message

On Thu, 2012-02-16 at 19:06 +0400, Stanislav Kinsbursky wrote:
> Local tranports uses UNIX sockets and connecting of these sockets is done in
> context of file system namespace (i.e. task file system root).
> Currenly, all sockets connect operations are performed by rpciod work queue,
> which actually means, that any service will be registered in the same rpcbind
> instance regardless to process file system root.
> This is not containers, which usually have it's own nested root. There are 2
> approaches, how to solve the problem. First one is to store proper root in
> tranport and switch to it in rpciod workqueue function for connect operations.
> But this looks ugly. The second one is to connect to unix sockets
> synchronously. This aptch implements the last one.

That approach can fall afoul of the selinux restrictions on the process
context. Processes that are allowed to write data, may not be allowed to
create sockets or call connect(). That is the main reason for doing it
in the rpciod context, which is a clean kernel process context.

--
Trond Myklebust
Linux NFS client maintainer

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

Subject: Re: [RFC PATCH] SUNRPC: connect local transports synchronously
Posted by Stanislav Kinsbursky on Fri, 17 Feb 2012 08:25:45 GMT

> On Thu, 2012-02-16 at 19:06 +0400, Stanislav Kinsbursky wrote:
>> Local tranports uses UNIX sockets and connecting of these sockets is done in
>> context of file system namespace (i.e. task file system root).
>> Currenly, all sockets connect operations are performed by rpciod work queue,
>> which actually means, that any service will be registered in the same rpcbind
>> instance regardless to process file system root.
>> This is not containers, which usually have it's own nested root. There are 2
>> approaches, how to solve the problem. First one is to store proper root in
>> tranport and switch to it in rpciod workqueue function for connect operations.
>> But this looks ugly. The second one is to connect to unix sockets
>> synchronously. This aptch implements the last one.
>
> That approach can fall afoul of the selinux restrictions on the process
> context. Processes that are allowed to write data, may not be allowed to
> create sockets or call connect(). That is the main reason for doing it
> in the rpciod context, which is a clean kernel process context.
>

Thanks for explanation, Trond.
So, this connect have to be done in kernel process context.
Now I can see 2 ways how to meet this requirement and reach the goal:

1) Change the fs root for rpciod while connecting.
2) Do not touch rpciod and launch special "connect" kernel thread to perform
connect operations for unix sockets.

What do you think about this 2 ways above? Which one is less worse from your POW?
Maybe you have even a better solution for the problem?

--
Best regards,
Stanislav Kinsbursky