Subject: Re: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by Stanislav Kinsbursky on Tue, 14 Aug 2012 08:46:37 GMT
View Forum Message <> Reply to Message

> On Mon, Aug 13, 2012 at 09:39:53PM +0400, Stanislav Kinsbursky wrote:

>>> On Sat, Aug 11, 2012 at 03:15:24PM +0400, Stanislav Kinsbursky wrote:

```
>>>>> On 08/11/2012 03:09 AM, H. Peter Anvin wrote:
>>>>>> On 08/10/2012 12:28 PM, Alan Cox wrote:
>>>>>>> Explicitly for Linux yes - this is not generally true of the
>>>>>>> AF_UNIX socket domain and even the permissions aspect isn't
>>>>>>> guaranteed to be supported on some BSD environments !
>>>>>> Yes, but let's worry about what the Linux behavior should be.
>>>>>>
>>>>>>> The name is however just a proxy for the socket itself. You
>>>>>>> don't even get a device node in the usual sense or the same inode
>>>>>>> in the file system space.
>>>>>> No, but it is looked up the same way any other inode is (the
>>>>>> difference between FIFOs and sockets is that sockets have separate
>>>>>> connections, which is also why open() on sockets would be nice.)
>>>>>>
>>>>>> However, there is a fundamental difference between AF_UNIX sockets
>>>>>> and open(), and that is how the pathname is delivered.  It thus
>>>>>> would make more sense to provide the openat()-like information in
>>>>>> struct sockaddr_un, but that may be very hard to do in a sensible
>>>>>> way.  In that sense it perhaps would be cleaner to be able to do
>>>>>> an open[at]() on the socket node with O_PATH (perhaps there should
>>>>>> be an O_SOCKET option, even?) and pass the resulting file
>>>>>> descriptor to bind() or connect().
>>>>> I vote for this (openat + O_WHATEVER on a unix socket) as well. It
>>>>> will help us in checkpoint-restore, making handling of
>>>>> overmounted/unlinked sockets much cleaner.
>>>> I have to notice, that it's not enough and doesn't solve the issue.
>>>> There should be some way how to connect/bind already existent unix
>>>> socket (from kernel, at least), because socket can be created in user
>>>> space. And this way (sock operation or whatever) have to provide an
>>>> ability to lookup UNIX socket starting from specified root to support
>>>> containers.
>>> I don't understand--the rpcbind sockets are created by the kernel.  What
>>>  am I missing?
>>
>> Kernel preform connect to rpcbind socket (i.e. user-space binds it),
>> doesn't it?
>
> I'm confused, possibly because there are three "sockets" here: the
> client-side socket that's connected, the server-side socket that's bound,
```

> and the common object that exists in the filesystem namespace.
>
> Userland creates the server-side socket and binds to it. All of that is
> done in the context of the rpcbind process, so is created in rpcbind's
> namespace. That should be OK, right?
>
> The client side socket is created and connected in xs_local_setup_socket().
>
> Making sure they both end up with the same thing is a matter of making sure
> they lookup the same path in the same namespace. The difficult part of that
> is the in-kernel client-side socket connect, where we don't have the right
> process context any more.
>

Looks like I'm missing something important.
Where are these UNIX in-kernel created and listening sockets (in code, I mean)?

--
Best regards,
Stanislav Kinsbursky