Subject: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by Stanislav Kinsbursky on Fri, 10 Aug 2012 12:57:23 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 UNIX sockets lookup is done in context of process file system root. I.e. all local thunsports are connecting in rpciod context.
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 set introduces kernel connect helper for UNIX stream sockets and modifies unix_find_other() to be able to search from specified root.
It also replaces generic socket connect call for local transports by new helper in SUNRPC layer.

The following series implements...

Stanislav Kinsbursky (2):
    unix sockets: add ability for search for peer from passed root
    SUNRPC: connect local transports with unix_stream_connect_root() helper

 include/net/af_unix.h | 2 ++
 net/sunrpc/xprtsock.c | 28 ++++++++++++++++++++++++---
 net/unix/af_unix.c | 25 ++++++++++++++++++-------
 3 files changed, 45 insertions(+), 10 deletions(-)

Subject: [RFC PATCH 2/2] SUNRPC: connect local transports with unix_stream_connect_root() helper
Posted by Stanislav Kinsbursky on Fri, 10 Aug 2012 12:57:35 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 UNIX sockets lookup is done in context of process file system root. I.e. all local thunsports are connecting in rpciod context.
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 using special helper
unix_stream_connect_root(), which lookup socket file starting from passed
root.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 net/sunrpc/xprtsock.c | 28 ++++++++++++++++++++++++++---
 1 files changed, 25 insertions(+), 3 deletions(-)

diff --git a/net/sunrpc/xprtsock.c b/net/sunrpc/xprtsock.c
index 890b03f..01a6f2a 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
@@ -45,6 +46,7 @@
 #include <net/checksum.h>
 #include <net/udp.h>
 #include <net/tcp.h>
+#include <net/af_unix.h>

 #include "sunrpc.h"

@@ -255,6 +257,11 @@ 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;
 };

 /*
@@ -1873,7 +1880,8 @@ static int xs_local_finish_connecting(struct rpc_xprt *xprt,
  /* Tell the socket layer to start connecting... */
  xprt->stat.connect_count++;
  xprt->stat.connect_start = jiffies;
- return kernel_connect(sock, xs_addr(xprt), xprt->addrlen, 0);

```
+ return unix_stream_connect_root(&transport->root, sock, xs_addr(xprt),
+     xprt->addrlen, 0);
 }

 /**
@@ -2213,6 +2221,18 @@ 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
@@ -2431,7 +2451,7 @@ 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,
 };

@@ -2607,8 +2627,10 @@ 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: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by hpa on Fri, 10 Aug 2012 18:15:02 GMT
View Forum Message <> Reply to Message

On 08/10/2012 05:57 AM, 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 UNIX sockets lookup is done in context of process file system root. I.e.
> all local thunsports are connecting in rpciod context.
> 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 set introduces kernel connect helper for UNIX stream sockets and
> modifies unix_find_other() to be able to search from specified root.
> It also replaces generic socket connect call for local transports by new
> helper in SUNRPC layer.
>
> The following series implements...

On that whole subject...

Do we need a Unix domain socket equivalent to openat()?

-hpa

Subject: Re: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by Alan Cox on Fri, 10 Aug 2012 18:23:09 GMT
View Forum Message <> Reply to Message

> On that whole subject...
>
> Do we need a Unix domain socket equivalent to openat()?

I don't think so. The name is just a file system indexing trick, it's not
really the socket proper. It's little more than "ascii string with
permissions attached" - indeed we also support an abstract name space
which for a lot of uses is actually more convenient.

AF_UNIX between roots raises some interesting semantic questions when you
begin passing file descriptors down them as well.

Alan

Subject: Re: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by hpa on Fri, 10 Aug 2012 18:31:25 GMT
View Forum Message <> Reply to Message

On 08/10/2012 11:26 AM, Alan Cox wrote:
>> On that whole subject...
>>
>> Do we need a Unix domain socket equivalent to openat()?
>
> I don't think so. The name is just a file system indexing trick, it's not
> really the socket proper. It's little more than "ascii string with
> permissions attached" - indeed we also support an abstract name space
> which for a lot of uses is actually more convenient.
>

I don't really understand why Unix domain sockets is different than any other pathname users in this sense. (Actually, I have never understood why open() on a Unix domain socket doesn't give the equivalent of a socket() + connect() -- it would make logical sense and would provide additional functionality).

It would be different if the Unix domain sockets simply required an absolute pathname (it is not just about the root, it is also about the cwd, which is where the -at() functions come into play), but that is not the case.

The abstract namespace is irrelevant for this, obviously.

> AF_UNIX between roots raises some interesting semantic questions when
> you begin passing file descriptors down them as well.

Why is that? A file descriptor carries all that information with it...

-hpa

Subject: Re: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by Alan Cox on Fri, 10 Aug 2012 18:37:48 GMT
View Forum Message <> Reply to Message

> > AF_UNIX between roots raises some interesting semantic questions when
> > you begin passing file descriptors down them as well.
>
> Why is that? A file descriptor carries all that information with it...

Things like fchdir(). It's not a machine breaking problem but for containers as opposed to chroot we need to be clear what the expected isolation sematics are.

Agreed on open() for sockets.. the lack of open is a Berklix derived
pecularity of the interface. It would equally be useful to be able to
open "/dev/socket/ipv4/1.2.3.4/1135" and the like for scripts and stuff

That needs VFS changes however so you can pass the remainder of a path to
a device node. It also lets you do a lot of other sane stuff like

open /dev/ttyS0/9600/8n1

Subject: Re: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by hpa on Fri, 10 Aug 2012 18:42:32 GMT
View Forum Message <> Reply to Message

On 08/10/2012 11:40 AM, Alan Cox wrote:
>
> Agreed on open() for sockets.. the lack of open is a Berklix derived
> pecularity of the interface. It would equally be useful to be able to
> open "/dev/socket/ipv4/1.2.3.4/1135" and the like for scripts and stuff
>
> That needs VFS changes however so you can pass the remainder of a path to
> a device node. It also lets you do a lot of other sane stuff like
>
> open /dev/ttyS0/9600/8n1
>

Well, supporting device node subpaths would be nice, but I don't think
that that is a requirement either for being able to open() a socket (as
a Linux extension) nor for supporting something like your above
/dev/socket/... since that could be done with a filesystem rather than
just a device node.

-hpa

Subject: Re: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by Stanislav Kinsbursky on Fri, 10 Aug 2012 18:50:48 GMT
View Forum Message <> Reply to Message

> On 08/10/2012 05:57 AM, 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 UNIX sockets lookup is done in context of process file system root. I.e.
>> all local thunsports are connecting in rpciod context.

>> 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 set introduces kernel connect helper for UNIX stream sockets and
>> modifies unix_find_other() to be able to search from specified root.
>> It also replaces generic socket connect call for local transports by new
>> helper in SUNRPC layer.
>>
>> The following series implements...
> On that whole subject...
>
> Do we need a Unix domain socket equivalent to openat()?

It looks like sys_connectat () and sys_bindat () could be an organic
part on openat () and friends family.
But currently I don't have any usage example for them in hands. And the
main problem here, that this syscalls can be used only for unix sockets.

Subject: Re: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by bfields on Fri, 10 Aug 2012 19:11:50 GMT
View Forum Message <> Reply to Message

On Fri, Aug 10, 2012 at 07:26:28PM +0100, Alan Cox wrote:
> > On that whole subject...
> >
> > Do we need a Unix domain socket equivalent to openat()?
>
> I don't think so. The name is just a file system indexing trick, it's not
> really the socket proper. It's little more than "ascii string with
> permissions attached"

That's overstating the case. As I understand it the address is resolved
by a pathname lookup like any other--it can follow symlinks, is relative
to the current working directory and filesystem namespace, etc. So a
unix-domain socket equivalent to openat() would at least be
well-defined--whether it's needed or not, I don't know.

--b.

Subject: Re: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by Alan Cox on Fri, 10 Aug 2012 19:24:44 GMT
View Forum Message <> Reply to Message

On Fri, 10 Aug 2012 15:11:50 -0400
"J. Bruce Fields" <bfields@fieldses.org> wrote:

> On Fri, Aug 10, 2012 at 07:26:28PM +0100, Alan Cox wrote:
> > > On that whole subject...
> > >
> > > Do we need a Unix domain socket equivalent to openat()?
> >
> > I don't think so. The name is just a file system indexing trick, it's not
> > really the socket proper. It's little more than "ascii string with
> > permissions attached"
>
> That's overstating the case. As I understand it the address is resolved
> by a pathname lookup like any other--it can follow symlinks, is relative
> to the current working directory and filesystem namespace, etc.

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 !

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.

Alan

Subject: Re: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by hpa on Fri, 10 Aug 2012 23:09:08 GMT
View Forum Message <> Reply to Message

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().

 -hpa

Subject: Re: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by Pavel Emelyanov on Sat, 11 Aug 2012 06:23:22 GMT
View Forum Message <> Reply to Message

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.

> -hpa

Thanks,
Pavel

Subject: Re: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by Stanislav Kinsbursky on Sat, 11 Aug 2012 11:15:24 GMT
View Forum Message <> Reply to Message

> 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.

>
>> -hpa
> Thanks,
> Pavel
>

Subject: Re: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by bfields on Mon, 13 Aug 2012 16:47:30 GMT
View Forum Message <> Reply to Message

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?

--b.

Subject: Re: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by Stanislav Kinsbursky on Mon, 13 Aug 2012 17:39:53 GMT
View Forum Message <> Reply to Message

> 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?

>
> --b.

Subject: Re: [RFC PATCH 0/2] net: connect to UNIX sockets from specified root
Posted by bfields on Mon, 13 Aug 2012 18:24:31 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.

We currently set that up with __sock_create followed by kernel_connect.

The proposal seems to be to instead do an openat followed by a kernel_connect, and pass the path in the openat instead of the connect.

(Though in the kernel we won't be able to call openat, so we'll end up doing something like nfsd does (calling lookup_one_len() and dentry_open() by hand).)

Have I got all that right?

I don't know if that's better just calling into the unix socket code at connect time as your patch does. Maybe the answer depends on whether it's a priority to make this functionality available to userspace.

--b.

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