Subject: netns : close all sockets at unshare ?
Posted by Daniel Lezcano on Tue, 02 Oct 2007 21:45:24 GMT
View Forum Message <> Reply to Message

Hi,

I was looking at some cornercases and trying to figure out what happens
if someone does:

1 - fd = socket(...)
2 - unshare(CLONE_NEWNET)
3 - bind(fd, ...) / listen(fd, ...)

There is here an interaction between two namespaces.
Trying to catch all these little tricky paths everywhere with the
network namespace is painful, perhaps we should consider a more radical
solution.

Shall we close all fd sockets when doing an unshare ? like a
close-on-exec behavior ?

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: netns : close all sockets at unshare ?
Posted by ebiederm on Tue, 02 Oct 2007 22:38:48 GMT
View Forum Message <> Reply to Message

Daniel Lezcano <dlezcano@fr.ibm.com> writes:

> Hi,
>
> I was looking at some cornercases and trying to figure out what happens if
> someone does:
>
> 1 - fd = socket(...)
> 2 - unshare(CLONE_NEWNET)
> 3 - bind(fd, ...) / listen(fd, ...)
>

> There is here an interaction between two namespaces.
> Trying to catch all these little tricky paths everywhere with the network
> namespace is painful, perhaps we should consider a more radical solution.

Huh?

socket() puts the namespace on struct sock.
bind/listen etc just look at that namespace.

Unless I'm blind it is simple and it works now.

> Shall we close all fd sockets when doing an unshare ? like a close-on-exec
> behavior ?

I think adopting that policy would dramatically reduce the usefulness
of network namespaces.

Making the mix and match cases gives the implementation much more flexibility
and it doesn't appear that hard right now.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: netns : close all sockets at unshare ?
Posted by Daniel Lezcano on Wed, 03 Oct 2007 08:40:49 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>
>> Hi,
>>
>> I was looking at some cornercases and trying to figure out what happens if
>> someone does:
>>
>> 1 - fd = socket(...)
>> 2 - unshare(CLONE_NEWNET)
>> 3 - bind(fd, ...) / listen(fd, ...)
>>
>> There is here an interaction between two namespaces.
>> Trying to catch all these little tricky paths everywhere with the network
>> namespace is painful, perhaps we should consider a more radical solution.
>
> Huh?

>
> socket() puts the namespace on struct sock.
> bind/listen etc just look at that namespace.
>
> Unless I'm blind it is simple and it works now.

Yes, it will work.

Do we want to be inside a network namespace and to use a socket
belonging to another network namespace ? If yes, then my remark is
irrelevant.

>> Shall we close all fd sockets when doing an unshare ? like a close-on-exec
>> behavior ?
>
> I think adopting that policy would dramatically reduce the usefulness
> of network namespaces.
>
> Making the mix and match cases gives the implementation much more flexibility
> and it doesn't appear that hard right now.

I am curious, why such functionality is useful ?

Subject: Re: netns : close all sockets at unshare ?
Posted by ebiederm on Wed, 03 Oct 2007 16:59:07 GMT
View Forum Message <> Reply to Message

Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>
> Yes, it will work.
>
> Do we want to be inside a network namespace and to use a socket belonging to
> another network namespace ? If yes, then my remark is irrelevant.

Yes we do.

>>> Shall we close all fd sockets when doing an unshare ? like a close-on-exec
>>> behavior ?
>>
>> I think adopting that policy would dramatically reduce the usefulness
>> of network namespaces.

>>
>> Making the mix and match cases gives the implementation much more flexibility
>> and it doesn't appear that hard right now.
>
> I am curious, why such functionality is useful ?

There are several reasons. Partly it is the principle of building
general purpose tools that can be used in a flexible way.

The biggest practical use I can see is that a control program outside
of a network namespace can configure and setup someone else's network
stack, perhaps preventing the need to enter someone else's container.

Another use is having a socket in an original network namespace for
doing a stdin/stdout style connections.

The planetlab folks are actually actively using this functionality
already, and there was a thread several months ago about how this
functionality was important and how they were using it.

This also preserves normal unix file descriptor passing semantics.

A final reason for it is that it removes the need for a lot of
brittle special cases when network namespaces are mixed in something
other then a 1-1 correspondence with other namespaces. Like the one
you were concerned with in unshare. Handling this case means
everything just works.

So it may be a touch harder to implement but because we don't add
special rules it is much easier to review.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: netns : close all sockets at unshare ?
Posted by Daniel Lezcano on Wed, 03 Oct 2007 19:33:42 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>> Yes, it will work.
>>
>> Do we want to be inside a network namespace and to use a socket belonging to
>> another network namespace ? If yes, then my remark is irrelevant.

>
> Yes we do.
>
>>>> Shall we close all fd sockets when doing an unshare ? like a close-on-exec
>>>> behavior ?
>>> I think adopting that policy would dramatically reduce the usefulness
>>> of network namespaces.
>>>
>>> Making the mix and match cases gives the implementation much more flexibility
>>> and it doesn't appear that hard right now.
>> I am curious, why such functionality is useful ?
>
> There are several reasons. Partly it is the principle of building
> general purpose tools that can be used in a flexible way.
>
> The biggest practical use I can see is that a control program outside
> of a network namespace can configure and setup someone else's network
> stack, perhaps preventing the need to enter someone else's container.
>
> Another use is having a socket in an original network namespace for
> doing a stdin/stdout style connections.
>
> The planetlab folks are actually actively using this functionality
> already, and there was a thread several months ago about how this
> functionality was important and how they were using it.
>
> This also preserves normal unix file descriptor passing semantics.
>
> A final reason for it is that it removes the need for a lot of
> brittle special cases when network namespaces are mixed in something
> other then a 1-1 correspondence with other namespaces. Like the one
> you were concerned with in unshare. Handling this case means
> everything just works.
>
> So it may be a touch harder to implement but because we don't add
> special rules it is much easier to review.

Very interesting. Thank you for taking the time to answer.

 -- Daniel

Eric W. Biederman wrote:
> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>> Yes, it will work.
>>
>> Do we want to be inside a network namespace and to use a socket belonging to
>> another network namespace ? If yes, then my remark is irrelevant.
>
> Yes we do.
>
>>>> Shall we close all fd sockets when doing an unshare ? like a close-on-exec
>>>> behavior ?
>>> I think adopting that policy would dramatically reduce the usefulness
>>> of network namespaces.
>>>
>>> Making the mix and match cases gives the implementation much more flexibility
>>> and it doesn't appear that hard right now.
>> I am curious, why such functionality is useful ?
>
> There are several reasons. Partly it is the principle of building
> general purpose tools that can be used in a flexible way.
>
> The biggest practical use I can see is that a control program outside
> of a network namespace can configure and setup someone else's network
> stack, perhaps preventing the need to enter someone else's container.
>
> Another use is having a socket in an original network namespace for
> doing a stdin/stdout style connections.

this is also required in the HPC world. batch managers and MPI frameworks
often redirect stdios on the socket which was used to remotely execute
the job. Now, if we introduce a container around the job to be able
to migrate it, we are in the same scenario.

that socket is in the original network namespace and the fds are from
processes living in a container.

C.

> The planetlab folks are actually actively using this functionality
> already, and there was a thread several months ago about how this
> functionality was important and how they were using it.
>
> This also preserves normal unix file descriptor passing semantics.
>
> A final reason for it is that it removes the need for a lot of

> brittle special cases when network namespaces are mixed in something
> other then a 1-1 correspondence with other namespaces. Like the one
> you were concerned with in unshare. Handling this case means
> everything just works.
>
> So it may be a touch harder to implement but because we don't add
> special rules it is much easier to review.
>
> Eric
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers
>

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers