Subject: Re: namespace support requires network modules to say "GPL"
Posted by ebiederm on Tue, 04 Dec 2007 18:03:01 GMT
View Forum Message <> Reply to Message

Daniel Lezcano <daniel.lezcano@free.fr> writes:

> Ben Greear wrote:

>>> Once things are largely complete it makes sense to argue with out of
>>> tree module authors that because they don't have network namespace
>>> support in their modules, their modules are broken.
>> Does this imply that every module that accesses the network code *must* become
>> GPL simply because it must interact with namespace logic that is exported as
>> GPL only symbols?
>
> That's right, with init_net's EXPORT_SYMBOL_GPL and dev_get_xx, we enforce
> people to be GPL whatever they didn't asked to have the namespaces in their
> code.
>
> Eric, why can we simply change EXPORT_SYMBOL_GPL to EXPORT_SYMBOL for init_net ?

Hmm. I need to think this one through.

EXPORT_SYMBOL_GPL acts as a strong hint, and a hindrance to using
symbols in a non-GPL'd module. Not exactly an enforcement mechanism.

...

The current pattern is to first change the code to only work in the
initial network namespace. Which can usually be done with a few
trivial lines of code that utilize init_net.

Then the pattern is to move the globals (or at least a pointer to
them) into struct net, and utilize register_pernet_subsys to ensure
those variables are properly initialized and cleaned up after.

However there also seem to be simpler cases like Ben's bridge module,
that don't appear to have any global state.

Ben I don't have a clue how your user space interface works. My gut
feel is that you can likely use sk->sk_net (if your configuration is
through a socket), or failing that current->nsproxy->net_ns. To get
the network namespace to look up "eth0" and "eth1".

This however still begs the question how do we want to handle this
so there is a minimum of pain.

Since using register_pernet_subsys implies you need your own member

in struct net. I am inclined to leave that with the GPL hint on
the EXPORT as you need to be really tight with the system to use that.

...

Currently I don't know if the _GPL hint on the export of init_net buys
us anything except trouble so I am almost inclined to do something
there.

....

What really disturbs me is that as I look at this I see that we have
historically at least done a very haphazard job of maintaining our
kernel/userspace ABIs while making a commitment to maintain them
forever. Especially if as it seems that some would see that
commitment extending beyond the code that is ever potentially
mergable with the kernel.

....

Currently the only angle that I can see that makes sense to me in the
argument for change of how we are currently doing things is that by
adding a parameter to new existing functions I make it very difficult
for code with network namespace support to have one version that works
on both old and new kernels as we can not define the new API on the
old hardware.

I can see some technical merit in making that case better.

.....

My thinking on the namespaces have been that their interfaces are new
core kernel interfaces that have not existed on any other kernel. And
as such any code that needed to use those interfaces was:
a) definitely a derived work of the kernel.
b) was a core part of the kernel, and we don't even want normal
   day to day drivers using those interfaces much less weird
   random code outside of the kernel.

The above is why I habitually place a _GPL hint on my exports
of namespace related functions and data. To strongly suggest
to module authors that they are getting into hot water if
they use these interfaces and don't merge their code.

So far I really don't see anything to challenge my understanding above
but I am human and as such my heuristics for analysis and understanding
are not guaranteed to give me the right answer.

....

I don't want this to be a stupid political fight about GPL stuff.
Generally I am with Alan in not seeing any basis for distributing
non-GPL code that works in the kernel. Although I see Linus' point
that a legal case may be made that certain modules are not a
derivative work of the kernel.

...

I am confused. I don't see a path forward that feels right.
So I am going to sit and think about this some more, before I do anything.

Eric

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