Subject: Re: [PATCH 1/5] net: Modify all rtnetlink methods to only work in the initial namespace
Posted by ebiederm on Wed, 10 Oct 2007 19:37:04 GMT
View Forum Message <> Reply to Message

"Denis V. Lunev" <den@sw.ru> writes:

> Eric W. Biederman wrote:
>> Before I can enable rtnetlink to work in all network namespaces
>> I need to be certain that something won't break. So this
>> patch deliberately disables all of the rtnletlink methods in everything
>> except the initial network namespace. After the methods have been
>> audited this extra check can be disabled.
>>
> [...]
>> static int br_dump_ifinfo(struct sk_buff *skb, struct netlink_callback *cb)
>> {
>> + struct net *net = skb->sk->sk_net;
>> struct net_device *dev;
>> int idx;
>>
>
> I've read some code today greping 'init_net.loopback_dev' and found
> interesting non-trivial for me issue.
>
> Network namespace is extracted from the packet in two different ways in
> TCP. This is a socket for outgoing path and a device for incoming.
> Though, there are some places called uniformly both from incoming and
> outgoing path.
>
> Typical example is netfilters. They are called uniformly all around the
> code. The prototype is the following:
>
> static unsigned int reject6_target(struct sk_buff **pskb,
> const struct net_device *in,
> const struct net_device *out,
> unsigned int hooknum,
> const struct xt_target *target,
> const void *targinfo);
>
> So, we are bound to the following options:
> - perform additional non-uniform hacks around to place 'struct net' into
> other and other structures like xt_target
> - add 7th parameter here and over

> - introduce an skb_net field in the 'struct sk_buff' making all code
> uniform, at least when we have an skb

No. That bloats a sk_buff, changes the semantics of moving a skb
around, and decreases performance (because we have to maintain the
field on a fast path).

There will not be a skb_net field.

The entire concept of skb_net is a maintenance disaster.

> I think that this is not the last place with such a parameter list and
> we should make a decision at this point when the code in not mainline yet.

Certainly that is what I have a proof of concept tree for. So we can
see how these things look before we merge them.

> As far as I understand, netfilters are not touched by the Eric and we
> can face some non-trivial problems there.

No. In my proof of concept tree I should have working per network
namespace netfilter code. My intention was to just do enough to see
what the impact would be so most of the netfilter code (in my tree)
insists on running in the initial network namespace. But there are
a few pieces that are fully converted. Please take a look.

> So, if my point about uniformity is valid, this patchset looks wrong and
> should be re-worked :(

This patchset does need to get rebased on top of net-2.6.25 when it
opens and hopefully your patchset to remove the unnecessary work in
rtnl_unlock, and to really process netlink requests in process
context. I see a need for the more fundamental change you seem to
be advocating.

Differentiating between the incoming and the outgoing code paths is
something we already do permission checking, for locking, for
sleeping, etc. Modifying the code requires reading and understanding
it in context. That is the nature of code.

This does make large patches going across the entire networking stack
making something a network namespace parameter difficult, but it
should not cause any problem for maintenance or other work on
the code. As shown by the fact that even outside the tree rebasing my
network namespace patches has not been all that difficult.

So no I don't think uniformity, or beauty or elegance is what we
are after right now. Trying to hard in that direction ultimately
obfuscates the code.

What we want is something that is simple, straight forward, and

doesn't require you to be an expert in network namespaces to
understand the code or the patches.

In the particular case of the netfilter hooks we don't have a
network namespace parameter laying around before we call NF_HOOK,
and the idiom "net = (in?in:out)->nd_net" seems perfectly accurate
so it seems reasonable to me to derive the network namespace that
way in generic code. Although thinking about this. We know which
hooks we are being called from so we may in fact actually know
if which of in or out must be valid when we get to the netfilter
hook.

Eric
