Subject: Re: [PATCH 1/5] net: Modify all rtnetlink methods to only work in the initial namespace
Posted by den on Thu, 11 Oct 2007 06:35:58 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
[..]
>> - 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.

I see that current patchset of RTNL code is attached. I'll start the
next piece of work next week after some reaction from people.

[..]
> 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.

I understand your position. But still have some points :)

First. A real-life usecase we have recently fixed in the OpenVZ. I have described it in the previous post, but (may be) not in great details.

UDP buffers management for outgoing traffic.
The packet carries over a destructor, which is called in skb_orphan in the real networking device. This destructor allows to queue more packets. There is no problem in the current implementation. But there is one after namespace introduction in the following configuration:
    [NS1] <-> [NS2] <-> [world]
There are two namespaces on the host. One of them is connected to the outside world via another and the packet follows usual forwarding path in NS2.

The problem: if we call skb_orphan in [NS1] outgoing device, there is a great possibility to have a really huge packet drop in [NS2] on queuing operation.

In OpenVZ we have added skb_orphan into receive path (tcp_v4_rcv, __udp4_lib_rcv etc) and removed one from our virtual devices. As unfortunate side effect we have packet in NS2 with a socket from NS1.

So, we should have an architectural solution for this from a beginning. It will be too late to hack around and change namespace lookup scheme.

I see that we should adopt a generic approach:
- each concrete packet belongs to a concrete namespace
- if function has a packet as a parameter, we should get namespace from packet
- if skb->dev is present we should get namespace as skb->dev->nd_net
- otherwise we should get skb->sk->sk_net. If skb->sk is NULL -> this is a bug

So, for the case of all netfilter calls we'll have a pskb->dev->nd_net defined correctly.

Regards,
 Den