Subject: Re: [NETLINK] Don't attach callback to a going-away netlink socket
Posted by Evgeniy Polyakov on Wed, 18 Apr 2007 08:42:07 GMT

View Forum Message <> Reply to Message

On Wed, Apr 18, 2007 at 10:26:31AM +0200, Patrick McHardy (kaber@trash.net) wrote:
> Evgeniy Polyakov wrote:
> > On Wed, Apr 18, 2007 at 12:16:18PM +0400, Pavel Emelianov (xemul@sw.ru) wrote:
> >
> >>Sorry, I forgot to put netdev and David in Cc when I first sent it.
> >>
> >>There is a race between netlink_dump_start() and netlink_release()
> >>that can lead to the situation when a netlink socket with non-zero
> >>callback is freed.
> >
> >
> > Out of curiosity, why not to fix a netlink_dump_start() to remove
> > callback in error path, since in 'no-error' path it removes it in
> > netlink_dump().
>
>
> It already does (netlink_destroy_callback), but that doesn't help
> with this race though since without this patch we don't enter the
> error path.

I thought that with releasing a socket, which will have a callback
attached only results in a leak of the callback? In that case we can
just free it in dump() just like it is done in no-error path already.
Or do I miss something additional?

> > And, btw, can release method be called while socket is being used, I
> > thought about proper reference counters should prevent this, but not
> > 100% sure with RCU dereferencing of the descriptor.
>
>
> The problem is asynchronous processing of the dump request in the
> context of a different process. Process requests a dump, message
> is queued and process returns from sendmsg since some other process
> is already processing the queue. Then the process closes the socket,
> resulting in netlink_release being called. When the dump request
> is finally processed the race Pavel described might happen. This
> can only happen for netlink families that use mutex_try_lock for
> queue processing of course.

Doesn't it called from ->sk_data_ready() which is synchronous with
respect to sendmsg, not sure about conntrack though, but it looks so?

--
Evgeniy Polyakov