Subject: Re: [PATCH 0/8] Cleanup/fix the sk_alloc() call
Posted by Arnaldo Carvalho de M on Wed, 31 Oct 2007 14:14:22 GMT
View Forum Message <> Reply to Message

Em Wed, Oct 31, 2007 at 05:32:20PM +0300, Pavel Emelyanov escreveu:
> Arnaldo Carvalho de Melo wrote:
> > Em Wed, Oct 31, 2007 at 04:40:01PM +0300, Pavel Emelyanov escreveu:
> >> The sk_alloc() function suffers from two problems:
> >> 1 (major). The error path is not clean in it - if the security
> >>   call fails, the net namespace is not put, if the try_module_get
> >>   fails additionally the security context is not released;
> >> 2 (minor). The zero_it argument is misleading, as it doesn't just
> >>   zeroes it, but performs some extra setup. Besides this argument
> >>   is used only in one place - in the sk_clone().
> >>
> >> So this set fixes these problems and performs some additional
> >> cleanup.
> >>
> >> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
> >
> > for the series:
> >
> > Acked-by: Arnaldo Carvalho de Melo <acme@redhat.com>
>
> Thanks a lot :)
>
> > Haven't tested, but it looks straightforward and conceptually sound,
> > thanks for improving the sk_prot infrastructure! :-)
>
> > Now we have just to make all the other protocols fill in the missing
> > sk->sk_prot-> methods (converting what is there now in socket->ops) so
> > that we can kill socket->ops and eliminate one level of indirection :-P
>
> Do I get your idea right, that having the 'struct sock->ops' field is not
> that good and the long-term TODO is to remove it (or smth similar)? Can you,
> please, pour some more light on this, because I'm not yet very common with
> the networking code, but I'm trying to learn it better by fixing obvious
> bugs and cleaning the code.

Start here:

```
const struct proto_ops inet_stream_ops = {
	.family		   = PF_INET,
	.owner		   = THIS_MODULE,
	.release	   = inet_release,
	.bind		   = inet_bind,
	.connect	   = inet_stream_connect,
	.socketpair	   = sock_no_socketpair,
        .accept         = inet_accept,
        .getname        = inet_getname,
        .poll           = tcp_poll,
        .ioctl          = inet_ioctl,
        .listen         = inet_listen,
        .shutdown       = inet_shutdown,
        .setsockopt     = sock_common_setsockopt,
        .getsockopt     = sock_common_getsockopt,
        .sendmsg        = tcp_sendmsg,
        .recvmsg        = sock_common_recvmsg,
        .mmap           = sock_no_mmap,
        .sendpage       = tcp_sendpage,
#ifdef CONFIG_COMPAT
        .compat_setsockopt = compat_sock_common_setsockopt,
        .compat_getsockopt = compat_sock_common_getsockopt,
#endif
};
```

Now look at all the "*_common_*" stuff, for instance:

```
int sock_common_recvmsg(struct kiocb *iocb, struct socket *sock,
                        struct msghdr *msg, size_t size, int flags)
{
        struct sock *sk = sock->sk;
        int addr_len = 0;
        int err;

        err = sk->sk_prot->recvmsg(iocb, sk, msg, size, flags & MSG_DONTWAIT,
                                   flags & ~MSG_DONTWAIT, &addr_len);
        if (err >= 0)
                msg->msg_namelen = addr_len;
        return err;
}
```

So if we made all protocols implement sk->sk_prot_recvmsg... got it?

And then look at the inet_* routines above, at least for LLC I was using
several unmodified.

Over the years the quality work is done on the mainstream protocols,
with the legacy ones lagging behind, so the more we share...

Anyway, look at my paper about it:

http://www.linuxsymposium.org/proceedings/reprints/Reprint-Melo-OLS2004.pdf

The DCCP paper also talks about this:

http://www.linuxinsight.com/files/ols2005/melo-reprint.pdf

- Arnaldo