
Subject: [PATCH] Don't attach callback to a going-away netlink socket
Posted by [xemul](#) on Mon, 16 Apr 2007 11:34:52 GMT

[View Forum Message](#) <> [Reply to Message](#)

From: Denis Lunev <den@openvz.org>

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.

```
--- a/net/netlink/af_netlink.c 2004-10-25 12:12:23.000000000 +0400
+++ b/net/netlink/af_netlink.c 2004-10-28 16:26:12.000000000 +0400
@@ -255,6 +255,7 @@ static int netlink_release(struct socket
     return 0;

     netlink_remove(sk);
+ sock_orphan(sk);
     nlk = nlk_sk(sk);

     spin_lock(&nlk->cb_lock);
@@ -269,7 +270,6 @@ static int netlink_release(struct socket
 /* OK. Socket is unlinked, and, therefore,
    no new packets will arrive */

- sock_orphan(sk);
  sock->sk = NULL;
  wake_up_interruptible_all(&nlk->wait);

@@ -942,9 +942,9 @@ int netlink_dump_start(struct sock *ssk,
     return -ECONNREFUSED;
 }
 nlk = nlk_sk(sk);
- /* A dump is in progress... */
+ /* A dump or destruction is in progress... */
  spin_lock(&nlk->cb_lock);
- if (nlk->cb) {
+ if (nlk->cb || sock_flag(sk, SOCK_DEAD)) {
     spin_unlock(&nlk->cb_lock);
     netlink_destroy_callback(cb);
     sock_put(sk);
```

Subject: Re: [PATCH] Don't attach callback to a going-away netlink socket
Posted by [Patrick McHardy](#) on Mon, 16 Apr 2007 11:41:54 GMT

[View Forum Message](#) <> [Reply to Message](#)

Pavel Emelianov wrote:

> From: Denis Lunev <den@openvz.org>

>
> 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.

Can you describe the race in more detail please?

Subject: Re: [PATCH] Don't attach callback to a going-away netlink socket
Posted by [xemul](#) on Mon, 16 Apr 2007 11:53:17 GMT
[View Forum Message](#) <> [Reply to Message](#)

Patrick McHardy wrote:

> Pavel Emelianov wrote:
>> From: Denis Lunev <den@openvz.org>
>>
>> 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.
>
>
> Can you describe the race in more detail please?
>
>

Here it is:

```
CPU1:          CPU2
netlink_release():      netlink_dump_start():

                        sk = netlink_lookup(); /* OK */

netlink_remove();

spin_lock(&nlk->cb_lock);
if (nlk->cb) { /* false */
    ...
}
spin_unlock(&nlk->cb_lock);

                        spin_lock(&nlk->cb_lock);
                        if (nlk->cb) { /* false */
                                ...
                        }
                        nlk->cb = cb;
                        spin_unlock(&nlk->cb_lock);
                        ...
```

```
sock_orphan(sk);
/*
 * proceed with releasing
 * the socket
 */
```

The proposal is to make sock_orphan before detaching the callback in netlink_release() and to check for the sock to be SOCK_DEAD in netlink_dump_start() before setting a new callback.

Subject: Re: [PATCH] Don't attach callback to a going-away netlink socket
Posted by [James Morris](#) on Mon, 16 Apr 2007 12:37:30 GMT
[View Forum Message](#) <> [Reply to Message](#)

Please post networking patches to the networking developer list:

<http://vger.kernel.org/vger-lists.html#netdev>

- James

--

James Morris
<jmorris@namei.org>

Subject: Re: [PATCH] Don't attach callback to a going-away netlink socket
Posted by [Patrick McHardy](#) on Mon, 16 Apr 2007 12:55:08 GMT
[View Forum Message](#) <> [Reply to Message](#)

Pavel Emelianov wrote:

> Patrick McHardy wrote:

>

>>>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.

>>

>>

>>Can you describe the race in more detail please?

>>

> Here it is:

>

> [...]

> The proposal is to make sock_orphan before detaching the callback
> in netlink_release() and to check for the sock to be SOCK_DEAD in

> netlink_dump_start() before setting a new callback.

Thanks, good catch. Your patch also looks good.

Acked-by: Patrick McHardy <kaber@trash.net>
