Subject: [NETLINK] Don't attach callback to a going-away netlink socket
Posted by xemul on Wed, 18 Apr 2007 08:12:18 GMT
View Forum Message <> Reply to Message

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.

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 it 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.

Signed-off-by: Denis Lunev <den@openvz.org>
Signed-off-by: Kirill Korotaev <dev@openvz.org>
Signed-off-by: Pavel Emelianov <xemul@openvz.org>
Acked-by: Patrick McHardy <kaber@trash.net>

```
--- 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);
```