Subject: [PATCH] Fix inet_diag.ko register vs rcv race
Posted by Pavel Emelianov on Tue, 27 Nov 2007 13:09:43 GMT
View Forum Message <> Reply to Message

The following race is possible when one cpu unregisters the handler
while other one is trying to receive a message and call this one:

```
CPU1:                                   CPU2:
inet_diag_rcv()                           inet_diag_unregister()
  mutex_lock(&inet_diag_mutex);
  netlink_rcv_skb(skb, &inet_diag_rcv_msg);
    if (inet_diag_table[nlh->nlmsg_type] ==
                        NULL) /* false handler is still registered */
    ...
    netlink_dump_start(idiagnl, skb, nlh,
                       inet_diag_dump, NULL);
          cb = kzalloc(sizeof(*cb), GFP_KERNEL);
                /* sleep here freeing memory
                 * or preempt
                 * or sleep later on nlk->cb_mutex
                 */
                                          spin_lock(&inet_diag_register_lock);
                                          inet_diag_table[type] = NULL;
    ...                                   spin_unlock(&inet_diag_register_lock);
                                          synchronize_rcu();
                                          /* CPU1 is sleeping - RCU quiescent
                                           * state is passed
                                           */
                                          return;
    /* inet_diag_dump is finally called: */
    inet_diag_dump()
      handler = inet_diag_table[cb->nlh->nlmsg_type];
      BUG_ON(handler == NULL);
      /* OOPS! While we slept the unregister has set
       * handler to NULL :(
       */
```

Grep showed, that the register/unregister functions are called
from init/fini module callbacks for tcp_/dccp_diag, so it's OK
to use the inet_diag_mutex to synchronize manipulations with the
inet_diag_table and the access to it.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

---
diff --git a/net/ipv4/inet_diag.c b/net/ipv4/inet_diag.c
index b017073..5fe32d5 100644
--- a/net/ipv4/inet_diag.c

```
+++ b/net/ipv4/inet_diag.c
@@ -853,8 +853,6 @@ static void inet_diag_rcv(struct sk_buff *skb)
  mutex_unlock(&inet_diag_mutex);
 }

-static DEFINE_SPINLOCK(inet_diag_register_lock);
-
 int inet_diag_register(const struct inet_diag_handler *h)
 {
  const __u16 type = h->idiag_type;
@@ -863,13 +861,13 @@ int inet_diag_register(const struct inet_diag_handler *h)
  if (type >= INET_DIAG_GETSOCK_MAX)
   goto out;

- spin_lock(&inet_diag_register_lock);
+ mutex_lock(&inet_diag_mutex);
  err = -EEXIST;
  if (inet_diag_table[type] == NULL) {
   inet_diag_table[type] = h;
   err = 0;
  }
- spin_unlock(&inet_diag_register_lock);
+ mutex_unlock(&inet_diag_mutex);
 out:
  return err;
 }
@@ -882,11 +880,9 @@ void inet_diag_unregister(const struct inet_diag_handler *h)
  if (type >= INET_DIAG_GETSOCK_MAX)
   return;

- spin_lock(&inet_diag_register_lock);
+ mutex_lock(&inet_diag_mutex);
  inet_diag_table[type] = NULL;
- spin_unlock(&inet_diag_register_lock);
-
- synchronize_rcu();
+ mutex_unlock(&inet_diag_mutex);
 }
 EXPORT_SYMBOL_GPL(inet_diag_unregister);
```

Subject: Re: [PATCH] Fix inet_diag.ko register vs rcv race
Posted by Herbert Xu on Thu, 29 Nov 2007 12:37:34 GMT
View Forum Message <> Reply to Message

On Tue, Nov 27, 2007 at 04:09:43PM +0300, Pavel Emelyanov wrote:
> The following race is possible when one cpu unregisters the handler
> while other one is trying to receive a message and call this one:

Good catch! But I think we need a bit more to close this fully.

Dumps can resume asynchronously which means that they won't be
holding inet_diag_mutex. We can fix that pretty easily by
giving that as our cb_mutex.

So could you add that to your patch and resubmit?

Arnaldo, synchronize_rcu() doesn't work on its own. Whoever accesses
the object that it's supposed to protect has to use the correct RCU
primitives for this to work.

Synchronisation is like tango, it always takes two to make it work :)

Thanks,
--
Visit Openswan at http://www.openswan.org/
Email: Herbert Xu ~{PmV>HI~} <herbert@gondor.apana.org.au>
Home Page: http://gondor.apana.org.au/~herbert/
PGP Key: http://gondor.apana.org.au/~herbert/pubkey.txt

Subject: Re: [PATCH] Fix inet_diag.ko register vs rcv race
Posted by Arnaldo Carvalho de M on Thu, 29 Nov 2007 12:47:17 GMT
View Forum Message <> Reply to Message

Em Thu, Nov 29, 2007 at 11:37:34PM +1100, Herbert Xu escreveu:
> On Tue, Nov 27, 2007 at 04:09:43PM +0300, Pavel Emelyanov wrote:
> > The following race is possible when one cpu unregisters the handler
> > while other one is trying to receive a message and call this one:
>
> Good catch! But I think we need a bit more to close this fully.
>
> Dumps can resume asynchronously which means that they won't be
> holding inet_diag_mutex. We can fix that pretty easily by
> giving that as our cb_mutex.
>
> So could you add that to your patch and resubmit?
>
> Arnaldo, synchronize_rcu() doesn't work on its own. Whoever accesses
> the object that it's supposed to protect has to use the correct RCU
> primitives for this to work.
>
> Synchronisation is like tango, it always takes two to make it work :)

Agreed, I didn't checked that when refactoring inet_diag, leaving this
as it was before I put my hands on it :-)

- Arnaldo