Subject: [PATCH] [NETNS 4/4 net-2.6.25] Namespace stop vs 'ip r l' race.
Posted by den on Fri, 18 Jan 2008 12:53:16 GMT
View Forum Message <> Reply to Message

During network namespace stop process kernel side netlink sockets belonging
to a namespace should be closed. They should not prevent namespace to stop,
so they do not increment namespace usage counter. Though this counter will
be put during last sock_put.

The raplacement of the correct netns for init_ns solves the problem only
partial as socket to be stoped until proper stop is a valid netlink kernel
socket and can be looked up by the user processes. This is not a problem
until it resides in initial namespace (no processes inside this net), but
this is not true for init_net.

So, hold the referrence for a socket, remove it from lookup tables and only
after that change namespace and perform a last put.

Signed-off-by: Denis V. Lunev <den@openvz.org>
Tested-by: Alexey Dobriyan <adobriyan@openvz.org>
---
 net/core/rtnetlink.c | 15 ++----------
 net/ipv4/fib_frontend.c | 7 +-----
 net/netlink/af_netlink.c | 15 ++++++++++++++
 3 files changed, 18 insertions(+), 19 deletions(-)

diff --git a/net/core/rtnetlink.c b/net/core/rtnetlink.c
index 2ef9480..aafc34d 100644
--- a/net/core/rtnetlink.c
+++ b/net/core/rtnetlink.c
@@ -1365,25 +1365,14 @@ static int rtnetlink_net_init(struct net *net)
        rtnetlink_rcv, &rtnl_mutex, THIS_MODULE);
  if (!sk)
   return -ENOMEM;
-
- /* Don't hold an extra reference on the namespace */
- put_net(sk->sk_net);
  net->rtnl = sk;
  return 0;
 }

 static void rtnetlink_net_exit(struct net *net)
 {
- struct sock *sk = net->rtnl;
- if (sk) {
- /* At the last minute lie and say this is a socket for the
-   * initial network namespace. So the socket will be safe to
-   * free.

```
-   */
-  sk->sk_net = get_net(&init_net);
-  netlink_kernel_release(net->rtnl);
-  net->rtnl = NULL;
- }
+ netlink_kernel_release(net->rtnl);
+ net->rtnl = NULL;
 }

 static struct pernet_operations rtnetlink_net_ops = {
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index e787d21..62bd791 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -869,19 +869,14 @@ static int nl_fib_lookup_init(struct net *net)
        nl_fib_input, NULL, THIS_MODULE);
  if (sk == NULL)
   return -EAFNOSUPPORT;
- /* Don't hold an extra reference on the namespace */
- put_net(sk->sk_net);
  net->ipv4.fibnl = sk;
  return 0;
 }

 static void nl_fib_lookup_exit(struct net *net)
 {
- /* At the last minute lie and say this is a socket for the
-  * initial network namespace. So the socket will  be safe to free.
-  */
- net->ipv4.fibnl->sk_net = get_net(&init_net);
  netlink_kernel_release(net->ipv4.fibnl);
+ net->ipv4.fibnl = NULL;
 }

 static void fib_disable_ip(struct net_device *dev, int force)
diff --git a/net/netlink/af_netlink.c b/net/netlink/af_netlink.c
index 626a582..6b178e1 100644
--- a/net/netlink/af_netlink.c
+++ b/net/netlink/af_netlink.c
@@ -1396,6 +1396,9 @@ netlink_kernel_create(struct net *net, int unit, unsigned int groups,
  }
  netlink_table_ungrab();

+ /* Do not hold an extra referrence to a namespace as this socket is
+  * internal to a namespace and does not prevent it to stop. */
+ put_net(net);
  return sk;
```

```
 out_sock_release:
@@ -1411,7 +1414,19 @@ netlink_kernel_release(struct sock *sk)
 {
  if (sk == NULL || sk->sk_socket == NULL)
   return;
+
+ /*
+  * Last sock_put should drop referrence to sk->sk_net. It has already
+  * been dropped in netlink_kernel_create. Taking referrence to stopping
+  * namespace is not an option.
+  * Take referrence to a socket to remove it from netlink lookup table
+  * _alive_ and after that destroy it in the context of init_net.
+  */
+ sock_hold(sk);
  sock_release(sk->sk_socket);
+
+ sk->sk_net = get_net(&init_net);
+ sock_put(sk);
 }
 EXPORT_SYMBOL(netlink_kernel_release);

--
1.5.3.rc5
```