Subject: [patch 14/20] [Network namespace] Switch the l3 namespace using the destination address. Does not wo
Posted by Daniel Lezcano on Sun, 10 Dec 2006 21:58:31 GMT
View Forum Message <> Reply to Message

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>

---

 include/linux/net_namespace.h | 7 +++++++
 net/core/net_namespace.c | 28 ++++++++++++++++++++++++++++
 net/ipv4/ip_input.c | 21 +++++++++++++++++++-
 3 files changed, 55 insertions(+), 1 deletion(-)

Index: 2.6.19-rc6-mm2/net/ipv4/ip_input.c
===
--- 2.6.19-rc6-mm2.orig/net/ipv4/ip_input.c
+++ 2.6.19-rc6-mm2/net/ipv4/ip_input.c
@@ -374,6 +374,11 @@ int ip_rcv(struct sk_buff *skb, struct n
 {
  struct iphdr *iph;
  u32 len;
+ int err;
+#ifdef CONFIG_NET_NS
+ struct net_namespace *net_ns = current_net_ns;
+ struct net_namespace *dst_net_ns = NULL;
+#endif

  /* When the interface is in promisc. mode, drop all the crap
   * that it receives, do not try to analyse it.
@@ -393,6 +398,11 @@ int ip_rcv(struct sk_buff *skb, struct n

  iph = skb->nh.iph;

+#ifdef CONFIG_NET_NS
+ dst_net_ns = net_ns_find_from_dest_addr(iph->daddr);
+ if (dst_net_ns && net_ns != dst_net_ns)
+ push_net_ns(dst_net_ns, net_ns);
+#endif
  /*
   * RFC1122: 3.1.2.2 MUST silently discard any IP frame that fails the checksum.
   *
@@ -431,10 +441,19 @@ int ip_rcv(struct sk_buff *skb, struct n
  /* Remove any debris in the socket control block */
  memset(IPCB(skb), 0, sizeof(struct inet_skb_parm));

- return NF_HOOK(PF_INET, NF_IP_PRE_ROUTING, skb, dev, NULL,
+ err = NF_HOOK(PF_INET, NF_IP_PRE_ROUTING, skb, dev, NULL,

```
         ip_rcv_finish);
+#ifdef CONFIG_NET_NS
+ if (dst_net_ns && net_ns != dst_net_ns)
+  pop_net_ns(net_ns);
+#endif
+ return err;

 inhdr_error:
+#ifdef CONFIG_NET_NS
+ if (dst_net_ns && net_ns != dst_net_ns)
+  pop_net_ns(net_ns);
+#endif
  IP_INC_STATS_BH(IPSTATS_MIB_INHDRERRORS);
 drop:
         kfree_skb(skb);
Index: 2.6.19-rc6-mm2/include/linux/net_namespace.h
===================================================================
--- 2.6.19-rc6-mm2.orig/include/linux/net_namespace.h
+++ 2.6.19-rc6-mm2/include/linux/net_namespace.h
@@ -94,6 +94,8 @@ extern int net_ns_check_bind(int addr_ty
 extern __be32 net_ns_select_source_address(const struct net_device *dev,
         u32 dst, int scope);

+extern struct net_namespace *net_ns_find_from_dest_addr(u32 daddr);
+
 #define SELECT_SRC_ADDR net_ns_select_source_address

 #else /* CONFIG_NET_NS */
@@ -158,6 +160,11 @@ static inline __be32 net_ns_select_sourc
  return 0;
 }

+static inline struct net_namespace *net_ns_find_from_dest_addr(u32 daddr)
+{
+ return current_net_ns;
+}
+
 #define SELECT_SRC_ADDR inet_select_addr

 #endif /* !CONFIG_NET_NS */
Index: 2.6.19-rc6-mm2/net/core/net_namespace.c
===================================================================
--- 2.6.19-rc6-mm2.orig/net/core/net_namespace.c
+++ 2.6.19-rc6-mm2/net/core/net_namespace.c
@@ -375,4 +375,32 @@ out_unlock_both:
 out:
  return addr;
 }
```

```
+
+struct net_namespace *net_ns_find_from_dest_addr(u32 daddr)
+{
+ struct net_namespace *net_ns = NULL;
+ struct net_device *dev;
+ struct in_device *in_dev;
+
+ if (LOOPBACK(daddr))
+  return current_net_ns;
+
+ read_lock(&dev_base_lock);
+ rcu_read_lock();
+ for (dev = dev_base; dev; dev = dev->next) {
+  if ((in_dev = __in_dev_get_rcu(dev)) == NULL)
+   continue;
+  for_ifa(in_dev) {
+   if (ifa->ifa_local == daddr) {
+    net_ns = ifa->ifa_net_ns;
+    goto out_unlock_both;
+   }
+  } endfor_ifa(in_dev);
+ }
+out_unlock_both:
+ read_unlock(&dev_base_lock);
+ rcu_read_unlock();
+
+ return net_ns;
+}
 #endif /* CONFIG_NET_NS */

--
```