Subject: [PATCH RFC 23/31] net: Modify all rtnetlink methods to only work in the initial namespace
Posted by ebiederm on Thu, 25 Jan 2007 19:00:25 GMT
View Forum Message <> Reply to Message

From: Eric W. Biederman <ebiederm@xmission.com> - unquoted

Before I can enable rtnetlink to work in all network namespaces
I need to be certain that something won't break. So this
patch deliberately disables all of the methods and when they
are audited this extra check can be disabled.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
net/bridge/br_netlink.c | 9 +++++++++
net/core/fib_rules.c | 7 +++++++
net/core/neighbour.c | 18 ++++++++++++++++++
net/core/rtnetlink.c | 13 +++++++++++++
net/decnet/dn_dev.c | 12 ++++++++++++
net/decnet/dn_fib.c | 8 ++++++++
net/decnet/dn_route.c | 8 ++++++++
net/decnet/dn_rules.c | 5 +++++
net/decnet/dn_table.c | 4 ++++
net/ipv4/devinet.c | 12 ++++++++++++
net/ipv4/fib_frontend.c | 12 ++++++++++++
net/ipv4/fib_rules.c | 5 +++++
net/ipv6/addrconf.c | 31 +++++++++++++++++++++++++++++++
net/ipv6/fib6_rules.c | 5 +++++
net/ipv6/ip6_fib.c | 4 ++++
net/ipv6/route.c | 12 ++++++++++++
net/sched/act_api.c | 8 ++++++++
net/sched/cls_api.c | 8 ++++++++
net/sched/sch_api.c | 20 ++++++++++++++++++++
19 files changed, 201 insertions(+), 0 deletions(-)

diff --git a/net/bridge/br_netlink.c b/net/bridge/br_netlink.c
index 119b97d..85165a1 100644
--- a/net/bridge/br_netlink.c
+++ b/net/bridge/br_netlink.c
@@ -14,6 +14,7 @@
#include <linux/rtnetlink.h>
#include <net/netlink.h>
#include <net/net_namespace.h>
+#include <net/sock.h>
#include "br_private.h"

static inline size_t br_nlmsg_size(void)
@@ -104,9 +105,13 @@ errout:

```
 */
static int br_dump_ifinfo(struct sk_buff *skb, struct netlink_callback *cb)
{
+ net_t net = skb->sk->sk_net;
  struct net_device *dev;
  int idx;

+ if (!net_eq(net, init_net()))
+  return 0;
+
  read_lock(&per_net(dev_base_lock, init_net()));
  for (dev = per_net(dev_base, init_net()), idx = 0; dev; dev = dev->next) {
   /* not a bridge port */
@@ -133,12 +138,16 @@ skip:
 */
static int br_rtm_setlink(struct sk_buff *skb,  struct nlmsghdr *nlh, void *arg)
{
+ net_t net = skb->sk->sk_net;
  struct ifinfomsg *ifm;
  struct nlattr *protinfo;
  struct net_device *dev;
  struct net_bridge_port *p;
  u8 new_state;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  if (nlmsg_len(nlh) < sizeof(*ifm))
   return -EINVAL;

diff --git a/net/core/fib_rules.c b/net/core/fib_rules.c
index 2fa2708..00b4148 100644
--- a/net/core/fib_rules.c
+++ b/net/core/fib_rules.c
@@ -163,6 +163,9 @@ int fib_nl_newrule(struct sk_buff *skb, struct nlmsghdr* nlh, void *arg)
  struct nlattr *tb[FRA_MAX+1];
  int err = -EINVAL;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  if (nlh->nlmsg_len < nlmsg_msg_size(sizeof(*frh)))
   goto errout;

@@ -244,12 +247,16 @@ errout:

int fib_nl_delrule(struct sk_buff *skb, struct nlmsghdr* nlh, void *arg)
{
```

```
+ net_t net = skb->sk->sk_net;
  struct fib_rule_hdr *frh = nlmsg_data(nlh);
  struct fib_rules_ops *ops = NULL;
  struct fib_rule *rule;
  struct nlattr *tb[FRA_MAX+1];
  int err = -EINVAL;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  if (nlh->nlmsg_len < nlmsg_msg_size(sizeof(*frh)))
   goto errout;

diff --git a/net/core/neighbour.c b/net/core/neighbour.c
index f5d4f92..d89c6fe 100644
--- a/net/core/neighbour.c
+++ b/net/core/neighbour.c
@@ -1445,6 +1445,9 @@ int neigh_delete(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
  struct net_device *dev = NULL;
  int err = -EINVAL;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  if (nlmsg_len(nlh) < sizeof(*ndm))
   goto out;

@@ -1511,6 +1514,9 @@ int neigh_add(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
  struct net_device *dev = NULL;
  int err;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  err = nlmsg_parse(nlh, sizeof(*ndm), tb, NDA_MAX, NULL);
  if (err < 0)
   goto out;
@@ -1783,11 +1789,15 @@ static struct nla_policy nl_ntbl_parm_policy[NDTPA_MAX+1]
__read_mostly = {

 int neightbl_set(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct neigh_table *tbl;
  struct ndtmsg *ndtmsg;
  struct nlattr *tb[NDTA_MAX+1];
  int err;
```

```
+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  err = nlmsg_parse(nlh, sizeof(*ndtmsg), tb, NDTA_MAX,
      nl_neightbl_policy);
  if (err < 0)
@@ -1907,11 +1917,15 @@ errout:

 int neightbl_dump_info(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  int family, tidx, nidx = 0;
  int tbl_skip = cb->args[0];
  int neigh_skip = cb->args[1];
  struct neigh_table *tbl;

+ if (!net_eq(net, init_net()))
+  return 0;
+
  family = ((struct rtgenmsg *) nlmsg_data(cb->nlh))->rtgen_family;

  read_lock(&neigh_tbl_lock);
@@ -2030,9 +2044,13 @@ out:

 int neigh_dump_info(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  struct neigh_table *tbl;
  int t, family, s_t;

+ if (!net_eq(net, init_net()))
+  return 0;
+
  read_lock(&neigh_tbl_lock);
  family = ((struct rtgenmsg *) nlmsg_data(cb->nlh))->rtgen_family;
  s_t = cb->args[0];
diff --git a/net/core/rtnetlink.c b/net/core/rtnetlink.c
index 5ac07a0..9be586c 100644
--- a/net/core/rtnetlink.c
+++ b/net/core/rtnetlink.c
@@ -395,6 +395,9 @@ static int rtnl_dump_ifinfo(struct sk_buff *skb, struct netlink_callback *cb)
  int s_idx = cb->args[0];
  struct net_device *dev;

+ if (!net_eq(net, init_net()))
+  return 0;
+
  read_lock(&per_net(dev_base_lock, net));
```

```
  for (dev=per_net(dev_base, net), idx=0; dev; dev = dev->next, idx++) {
   if (idx < s_idx)
@@ -429,6 +432,9 @@ static int rtnl_setlink(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
  struct nlattr *tb[IFLA_MAX+1];
  char ifname[IFNAMSIZ];

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  err = nlmsg_parse(nlh, sizeof(*ifm), tb, IFLA_MAX, ifla_policy);
  if (err < 0)
   goto errout;
@@ -602,6 +608,9 @@ static int rtnl_getlink(struct sk_buff *skb, struct nlmsghdr* nlh, void *arg)
  int iw_buf_len = 0;
  int err;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  err = nlmsg_parse(nlh, sizeof(*ifm), tb, IFLA_MAX, ifla_policy);
  if (err < 0)
   return err;
@@ -650,9 +659,13 @@ errout:

 static int rtnl_dump_all(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  int idx;
  int s_idx = cb->family;

+ if (!net_eq(net, init_net()))
+  return 0;
+
  if (s_idx == 0)
   s_idx = 1;
  for (idx=1; idx<NPROTO; idx++) {
diff --git a/net/decnet/dn_dev.c b/net/decnet/dn_dev.c
index c83c8d1..a09275b 100644
--- a/net/decnet/dn_dev.c
+++ b/net/decnet/dn_dev.c
@@ -648,12 +648,16 @@ static struct nla_policy dn_ifa_policy[IFA_MAX+1] __read_mostly = {

 static int dn_nl_deladdr(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct nlattr *tb[IFA_MAX+1];
  struct dn_dev *dn_db;
  struct ifaddrmsg *ifm;
```

```
  struct dn_ifaddr *ifa, **ifap;
  int err = -EADDRNOTAVAIL;

+ if (!net_eq(net, init_net()))
+  goto errout;
+
  err = nlmsg_parse(nlh, sizeof(*ifm), tb, IFA_MAX, dn_ifa_policy);
  if (err < 0)
   goto errout;
@@ -680,6 +684,7 @@ errout:

 static int dn_nl_newaddr(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct nlattr *tb[IFA_MAX+1];
  struct net_device *dev;
  struct dn_dev *dn_db;
@@ -687,6 +692,9 @@ static int dn_nl_newaddr(struct sk_buff *skb, struct nlmsghdr *nlh, void
*arg)
  struct dn_ifaddr *ifa;
  int err;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  err = nlmsg_parse(nlh, sizeof(*ifm), tb, IFA_MAX, dn_ifa_policy);
  if (err < 0)
   return err;
@@ -788,11 +796,15 @@ errout:

 static int dn_nl_dump_ifaddr(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  int idx, dn_idx = 0, skip_ndevs, skip_naddr;
  struct net_device *dev;
  struct dn_dev *dn_db;
  struct dn_ifaddr *ifa;

+ if (!net_eq(net, init_net()))
+  return 0;
+
  skip_ndevs = cb->args[0];
  skip_naddr = cb->args[1];

diff --git a/net/decnet/dn_fib.c b/net/decnet/dn_fib.c
index cc2ab1f..832e1b4 100644
--- a/net/decnet/dn_fib.c
+++ b/net/decnet/dn_fib.c
```

```
@@ -503,10 +503,14 @@ static int dn_fib_check_attr(struct rtmsg *r, struct rtattr **rta)

 int dn_fib_rtm_delroute(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct dn_fib_table *tb;
  struct rtattr **rta = arg;
  struct rtmsg *r = NLMSG_DATA(nlh);

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  if (dn_fib_check_attr(r, rta))
   return -EINVAL;

@@ -519,10 +523,14 @@ int dn_fib_rtm_delroute(struct sk_buff *skb, struct nlmsghdr *nlh, void
*arg)

 int dn_fib_rtm_newroute(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct dn_fib_table *tb;
  struct rtattr **rta = arg;
  struct rtmsg *r = NLMSG_DATA(nlh);

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  if (dn_fib_check_attr(r, rta))
   return -EINVAL;

diff --git a/net/decnet/dn_route.c b/net/decnet/dn_route.c
index 9669e50..d942ea0 100644
--- a/net/decnet/dn_route.c
+++ b/net/decnet/dn_route.c
@@ -1528,6 +1528,7 @@ rtattr_failure:
  */
 int dn_cache_getroute(struct sk_buff *in_skb, struct nlmsghdr *nlh, void *arg)
 {
+ net_t net = in_skb->sk->sk_net;
  struct rtattr **rta = arg;
  struct rtmsg *rtm = NLMSG_DATA(nlh);
  struct dn_route *rt = NULL;
@@ -1536,6 +1537,9 @@ int dn_cache_getroute(struct sk_buff *in_skb, struct nlmsghdr *nlh,
void *arg)
  struct sk_buff *skb;
  struct flowi fl;
```

```
+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  memset(&fl, 0, sizeof(fl));
  fl.proto = DNPROTO_NSP;

@@ -1613,10 +1617,14 @@ out_free:
  */
 int dn_cache_dump(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  struct dn_route *rt;
  int h, s_h;
  int idx, s_idx;

+ if (!net_eq(net, init_net()))
+  return 0;
+
  if (NLMSG_PAYLOAD(cb->nlh, 0) < sizeof(struct rtmsg))
   return -EINVAL;
  if (!((struct rtmsg *)NLMSG_DATA(cb->nlh))->rtm_flags&RTM_F_CLONED))
diff --git a/net/decnet/dn_rules.c b/net/decnet/dn_rules.c
index e32d0c3..84eec40 100644
--- a/net/decnet/dn_rules.c
+++ b/net/decnet/dn_rules.c
@@ -243,6 +243,11 @@ static u32 dn_fib_rule_default_pref(void)

 int dn_fib_dump_rules(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
+
+ if (!net_eq(net, init_net()))
+  return 0;
+
  return fib_rules_dump(skb, cb, AF_DECnet);
 }

diff --git a/net/decnet/dn_table.c b/net/decnet/dn_table.c
index 13b2421..3ff151c 100644
--- a/net/decnet/dn_table.c
+++ b/net/decnet/dn_table.c
@@ -459,12 +459,16 @@ static int dn_fib_table_dump(struct dn_fib_table *tb, struct sk_buff
*skb,

 int dn_fib_dump(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  unsigned int h, s_h;
```

```
  unsigned int e = 0, s_e;
  struct dn_fib_table *tb;
  struct hlist_node *node;
  int dumped = 0;

+ if (!net_eq(net, init_net()))
+  return 0;
+
  if (NLMSG_PAYLOAD(cb->nlh, 0) >= sizeof(struct rtmsg) &&
   ((struct rtmsg *)NLMSG_DATA(cb->nlh))->rtm_flags&RTM_F_CLONED)
    return dn_cache_dump(skb, cb);
diff --git a/net/ipv4/devinet.c b/net/ipv4/devinet.c
index b0d12ec..7769b1c 100644
--- a/net/ipv4/devinet.c
+++ b/net/ipv4/devinet.c
@@ -443,6 +443,7 @@ struct in_ifaddr *inet_ifa_byprefix(struct in_device *in_dev, __be32 prefix,

 static int inet_rtm_deladdr(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct nlattr *tb[IFA_MAX+1];
  struct in_device *in_dev;
  struct ifaddrmsg *ifm;
@@ -451,6 +452,9 @@ static int inet_rtm_deladdr(struct sk_buff *skb, struct nlmsghdr *nlh, void
*arg

  ASSERT_RTNL();

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  err = nlmsg_parse(nlh, sizeof(*ifm), tb, IFA_MAX, ifa_ipv4_policy);
  if (err < 0)
   goto errout;
@@ -562,10 +566,14 @@ errout:

 static int inet_rtm_newaddr(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct in_ifaddr *ifa;

  ASSERT_RTNL();

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  ifa = rtm_to_ifaddr(nlh);
  if (IS_ERR(ifa))
```

```
   return PTR_ERR(ifa);
@@ -1173,12 +1181,16 @@ nla_put_failure:

 static int inet_dump_ifaddr(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  int idx, ip_idx;
  struct net_device *dev;
  struct in_device *in_dev;
  struct in_ifaddr *ifa;
  int s_ip_idx, s_idx = cb->args[0];

+ if (!net_eq(net, init_net()))
+  return 0;
+
  s_ip_idx = ip_idx = cb->args[1];
  read_lock(&per_net(dev_base_lock, init_net()));
  for (dev = per_net(dev_base, init_net()), idx = 0; dev; dev = dev->next, idx++) {
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index 449f42d..0e48fb8 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -538,10 +538,14 @@ errout:

 int inet_rtm_delroute(struct sk_buff *skb, struct nlmsghdr* nlh, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct fib_config cfg;
  struct fib_table *tb;
  int err;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  err = rtm_to_fib_config(skb, nlh, &cfg);
  if (err < 0)
   goto errout;
@@ -559,10 +563,14 @@ errout:

 int inet_rtm_newroute(struct sk_buff *skb, struct nlmsghdr* nlh, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct fib_config cfg;
  struct fib_table *tb;
  int err;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
```

```
+
  err = rtm_to_fib_config(skb, nlh, &cfg);
  if (err < 0)
   goto errout;
@@ -580,12 +588,16 @@ errout:

 int inet_dump_fib(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  unsigned int h, s_h;
  unsigned int e = 0, s_e;
  struct fib_table *tb;
  struct hlist_node *node;
  int dumped = 0;

+ if (!net_eq(net, init_net()))
+  return 0;
+
  if (nlmsg_len(cb->nlh) >= sizeof(struct rtmsg) &&
      ((struct rtmsg *) nlmsg_data(cb->nlh))->rtm_flags & RTM_F_CLONED)
   return ip_rt_dump(skb, cb);
diff --git a/net/ipv4/fib_rules.c b/net/ipv4/fib_rules.c
index b837c33..f2c50e0 100644
--- a/net/ipv4/fib_rules.c
+++ b/net/ipv4/fib_rules.c
@@ -279,6 +279,11 @@ nla_put_failure:

 int fib4_rules_dump(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
+
+ if (!net_eq(net, init_net()))
+  return 0;
+
  return fib_rules_dump(skb, cb, AF_INET);
 }

diff --git a/net/ipv6/addrconf.c b/net/ipv6/addrconf.c
index 7afe698..83b7312 100644
--- a/net/ipv6/addrconf.c
+++ b/net/ipv6/addrconf.c
@@ -2951,11 +2951,15 @@ static struct nla_policy ifa_ipv6_policy[IFA_MAX+1] __read_mostly
= {
 static int
 inet6_rtm_deladdr(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct ifaddrmsg *ifm;
```

```
  struct nlattr *tb[IFA_MAX+1];
  struct in6_addr *pfx;
  int err;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  err = nlmsg_parse(nlh, sizeof(*ifm), tb, IFA_MAX, ifa_ipv6_policy);
  if (err < 0)
   return err;
@@ -3003,6 +3007,7 @@ static int inet6_addr_modify(struct inet6_ifaddr *ifp, u8 ifa_flags,
 static int
 inet6_rtm_newaddr(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct ifaddrmsg *ifm;
  struct nlattr *tb[IFA_MAX+1];
  struct in6_addr *pfx;
@@ -3012,6 +3017,9 @@ inet6_rtm_newaddr(struct sk_buff *skb, struct nlmsghdr *nlh, void
*arg)
  u8 ifa_flags;
  int err;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  err = nlmsg_parse(nlh, sizeof(*ifm), tb, IFA_MAX, ifa_ipv6_policy);
  if (err < 0)
   return err;
@@ -3278,26 +3286,42 @@ done:

 static int inet6_dump_ifaddr(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  enum addr_type_t type = UNICAST_ADDR;
+
+ if (!net_eq(net, init_net()))
+  return 0;
+
  return inet6_dump_addr(skb, cb, type);
 }

 static int inet6_dump_ifmcaddr(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  enum addr_type_t type = MULTICAST_ADDR;
+
+ if (!net_eq(net, init_net()))
```

```
+  return 0;
+
  return inet6_dump_addr(skb, cb, type);
 }


 static int inet6_dump_ifacaddr(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  enum addr_type_t type = ANYCAST_ADDR;
+
+ if (!net_eq(net, init_net()))
+  return 0;
+
  return inet6_dump_addr(skb, cb, type);
 }

 static int inet6_rtm_getaddr(struct sk_buff *in_skb, struct nlmsghdr* nlh,
        void *arg)
 {
+ net_t net = in_skb->sk->sk_net;
  struct ifaddrmsg *ifm;
  struct nlattr *tb[IFA_MAX+1];
  struct in6_addr *addr = NULL;
@@ -3306,6 +3330,9 @@ static int inet6_rtm_getaddr(struct sk_buff *in_skb, struct nlmsghdr*
nlh,
  struct sk_buff *skb;
  int err;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  err = nlmsg_parse(nlh, sizeof(*ifm), tb, IFA_MAX, ifa_ipv6_policy);
  if (err < 0)
   goto errout;
@@ -3472,11 +3499,15 @@ nla_put_failure:

 static int inet6_dump_ifinfo(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  int idx, err;
  int s_idx = cb->args[0];
  struct net_device *dev;
  struct inet6_dev *idev;

+ if (!net_eq(net, init_net()))
+  return 0;
+
```

```
  read_lock(&per_net(dev_base_lock, init_net()));
  for (dev=per_net(dev_base, init_net()), idx=0; dev; dev = dev->next, idx++) {
   if (idx < s_idx)
diff --git a/net/ipv6/fib6_rules.c b/net/ipv6/fib6_rules.c
index 0862809..80d6de6 100644
--- a/net/ipv6/fib6_rules.c
+++ b/net/ipv6/fib6_rules.c
@@ -223,6 +223,11 @@ nla_put_failure:

 int fib6_rules_dump(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
+
+ if (!net_eq(net, init_net()))
+  return 0;
+
  return fib_rules_dump(skb, cb, AF_INET6);
 }

diff --git a/net/ipv6/ip6_fib.c b/net/ipv6/ip6_fib.c
index 96d8310..97814ed 100644
--- a/net/ipv6/ip6_fib.c
+++ b/net/ipv6/ip6_fib.c
@@ -362,6 +362,7 @@ end:

 int inet6_dump_fib(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  unsigned int h, s_h;
  unsigned int e = 0, s_e;
  struct rt6_rtnl_dump_arg arg;
@@ -370,6 +371,9 @@ int inet6_dump_fib(struct sk_buff *skb, struct netlink_callback *cb)
  struct hlist_node *node;
  int res = 0;

+ if (!net_eq(net, init_net()))
+  return 0;
+
  s_h = cb->args[0];
  s_e = cb->args[1];

diff --git a/net/ipv6/route.c b/net/ipv6/route.c
index 4519006..02fd8ae 100644
--- a/net/ipv6/route.c
+++ b/net/ipv6/route.c
@@ -1985,9 +1985,13 @@ errout:

 int inet6_rtm_delroute(struct sk_buff *skb, struct nlmsghdr* nlh, void *arg)
```

```
 {
+ net_t net = skb->sk->sk_net;
  struct fib6_config cfg;
  int err;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  err = rtm_to_fib6_config(skb, nlh, &cfg);
  if (err < 0)
   return err;
@@ -1997,9 +2001,13 @@ int inet6_rtm_delroute(struct sk_buff *skb, struct nlmsghdr* nlh, void
*arg)

 int inet6_rtm_newroute(struct sk_buff *skb, struct nlmsghdr* nlh, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct fib6_config cfg;
  int err;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  err = rtm_to_fib6_config(skb, nlh, &cfg);
  if (err < 0)
   return err;
@@ -2132,6 +2140,7 @@ int rt6_dump_route(struct rt6_info *rt, void *p_arg)

 int inet6_rtm_getroute(struct sk_buff *in_skb, struct nlmsghdr* nlh, void *arg)
 {
+ net_t net = in_skb->sk->sk_net;
  struct nlattr *tb[RTA_MAX+1];
  struct rt6_info *rt;
  struct sk_buff *skb;
@@ -2139,6 +2148,9 @@ int inet6_rtm_getroute(struct sk_buff *in_skb, struct nlmsghdr* nlh,
void *arg)
  struct flowi fl;
  int err, iif = 0;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  err = nlmsg_parse(nlh, sizeof(*rtm), tb, RTA_MAX, rtm_ipv6_policy);
  if (err < 0)
   goto errout;
diff --git a/net/sched/act_api.c b/net/sched/act_api.c
index 835070e..18d8f68 100644
--- a/net/sched/act_api.c
```

```
+++ b/net/sched/act_api.c
@@ -942,10 +942,14 @@ done:

 static int tc_ctl_action(struct sk_buff *skb, struct nlmsghdr *n, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct rtattr **tca = arg;
  u32 pid = skb ? NETLINK_CB(skb).pid : 0;
  int ret = 0, ovr = 0;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  if (tca[TCA_ACT_TAB-1] == NULL) {
   printk("tc_ctl_action: received NO action attribs\n");
   return -EINVAL;
@@ -1015,6 +1019,7 @@ find_dump_kind(struct nlmsghdr *n)
 static int
 tc_dump_action(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  struct nlmsghdr *nlh;
  unsigned char *b = skb->tail;
  struct rtattr *x;
@@ -1024,6 +1029,9 @@ tc_dump_action(struct sk_buff *skb, struct netlink_callback *cb)
  struct tcamsg *t = (struct tcamsg *) NLMSG_DATA(cb->nlh);
  struct rtattr *kind = find_dump_kind(cb->nlh);

+ if (!net_eq(net, init_net()))
+  return 0;
+
  if (kind == NULL) {
   printk("tc_dump_action: action bad kind\n");
   return 0;
diff --git a/net/sched/cls_api.c b/net/sched/cls_api.c
index 19935f9..09a3ec8 100644
--- a/net/sched/cls_api.c
+++ b/net/sched/cls_api.c
@@ -129,6 +129,7 @@ static __inline__ u32 tcf_auto_prio(struct tcf_proto *tp)

 static int tc_ctl_tfilter(struct sk_buff *skb, struct nlmsghdr *n, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct rtattr **tca;
  struct tcmsg *t;
  u32 protocol;
@@ -145,6 +146,9 @@ static int tc_ctl_tfilter(struct sk_buff *skb, struct nlmsghdr *n, void *arg)
  unsigned long fh;
```

```
  int err;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
 replay:
  tca = arg;
  t = NLMSG_DATA(n);
@@ -385,6 +389,7 @@ static int tcf_node_dump(struct tcf_proto *tp, unsigned long n, struct
tcf_walke

 static int tc_dump_tfilter(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  int t;
  int s_t;
  struct net_device *dev;
@@ -395,6 +400,9 @@ static int tc_dump_tfilter(struct sk_buff *skb, struct netlink_callback *cb)
  struct Qdisc_class_ops *cops;
  struct tcf_dump_args arg;

+ if (!net_eq(net, init_net()))
+  return 0;
+
  if (cb->nlh->nlmsg_len < NLMSG_LENGTH(sizeof(*tcm)))
   return skb->len;
  if ((dev = dev_get_by_index(init_net(), tcm->tcm_ifindex)) == NULL)
diff --git a/net/sched/sch_api.c b/net/sched/sch_api.c
index 912e8e1..7e33f73 100644
--- a/net/sched/sch_api.c
+++ b/net/sched/sch_api.c
@@ -578,6 +578,7 @@ check_loop_fn(struct Qdisc *q, unsigned long cl, struct qdisc_walker *w)

 static int tc_get_qdisc(struct sk_buff *skb, struct nlmsghdr *n, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct tcmsg *tcm = NLMSG_DATA(n);
  struct rtattr **tca = arg;
  struct net_device *dev;
@@ -586,6 +587,9 @@ static int tc_get_qdisc(struct sk_buff *skb, struct nlmsghdr *n, void *arg)
  struct Qdisc *p = NULL;
  int err;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  if ((dev = __dev_get_by_index(init_net(), tcm->tcm_ifindex)) == NULL)
   return -ENODEV;
```

```
@@ -639,6 +643,7 @@ static int tc_get_qdisc(struct sk_buff *skb, struct nlmsghdr *n, void *arg)

 static int tc_modify_qdisc(struct sk_buff *skb, struct nlmsghdr *n, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct tcmsg *tcm;
  struct rtattr **tca;
  struct net_device *dev;
@@ -646,6 +651,9 @@ static int tc_modify_qdisc(struct sk_buff *skb, struct nlmsghdr *n, void
*arg)
  struct Qdisc *q, *p;
  int err;

+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
 replay:
  /* Reinit, just in case something touches this. */
  tcm = NLMSG_DATA(n);
@@ -851,11 +859,15 @@ err_out:

 static int tc_dump_qdisc(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  int idx, q_idx;
  int s_idx, s_q_idx;
  struct net_device *dev;
  struct Qdisc *q;

+ if (!net_eq(net, init_net()))
+  return 0;
+
  s_idx = cb->args[0];
  s_q_idx = q_idx = cb->args[1];
  read_lock(&per_net(dev_base_lock, init_net()));
@@ -900,6 +912,7 @@ done:

 static int tc_ctl_tclass(struct sk_buff *skb, struct nlmsghdr *n, void *arg)
 {
+ net_t net = skb->sk->sk_net;
  struct tcmsg *tcm = NLMSG_DATA(n);
  struct rtattr **tca = arg;
  struct net_device *dev;
@@ -912,6 +925,9 @@ static int tc_ctl_tclass(struct sk_buff *skb, struct nlmsghdr *n, void *arg)
  u32 qid = TC_H_MAJ(clid);
  int err;
```

```
+ if (!net_eq(net, init_net()))
+  return -EINVAL;
+
  if ((dev = __dev_get_by_index(init_net(), tcm->tcm_ifindex)) == NULL)
   return -ENODEV;

@@ -1086,6 +1102,7 @@ static int qdisc_class_dump(struct Qdisc *q, unsigned long cl, struct
qdisc_walk

 static int tc_dump_tclass(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ net_t net = skb->sk->sk_net;
  int t;
  int s_t;
  struct net_device *dev;
@@ -1093,6 +1110,9 @@ static int tc_dump_tclass(struct sk_buff *skb, struct netlink_callback
*cb)
  struct tcmsg *tcm = (struct tcmsg*)NLMSG_DATA(cb->nlh);
  struct qdisc_dump_args arg;

+ if (!net_eq(net, init_net()))
+  return 0;
+
  if (cb->nlh->nlmsg_len < NLMSG_LENGTH(sizeof(*tcm)))
   return 0;
  if ((dev = dev_get_by_index(init_net(), tcm->tcm_ifindex)) == NULL)
--
1.4.4.1.g278f
```