Subject: [PATCH 12/16] net: Support multiple network namespaces with netlink
Posted by ebiederm on Sat, 08 Sep 2007 21:28:27 GMT
View Forum Message <> Reply to Message

Each netlink socket will live in exactly one network namespace,
this includes the controlling kernel sockets.

This patch updates all of the existing netlink protocols
to only support the initial network namespace. Request
by clients in other namespaces will get -ECONREFUSED.
As they would if the kernel did not have the support for
that netlink protocol compiled in.

As each netlink protocol is updated to be multiple network
namespace safe it can register multiple kernel sockets
to acquire a presence in the rest of the network namespaces.

The implementation in af_netlink is a simple filter implementation
at hash table insertion and hash table look up time.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
drivers/connector/connector.c | 2 +-
drivers/scsi/scsi_netlink.c | 2 +-
drivers/scsi/scsi_transport_iscsi.c | 2 +-
fs/ecryptfs/netlink.c | 2 +-
include/linux/netlink.h | 6 ++-
kernel/audit.c | 4 +-
lib/kobject_uevent.c | 5 +-
net/bridge/netfilter/ebt_ulog.c | 5 +-
net/core/rtnetlink.c | 4 +-
net/decnet/netfilter/dn_rtmsg.c | 3 +-
net/ipv4/fib_frontend.c | 4 +-
net/ipv4/inet_diag.c | 4 +-
net/ipv4/netfilter/ip_queue.c | 6 +-
net/ipv4/netfilter/ipt_ULOG.c | 3 +-
net/ipv6/netfilter/ip6_queue.c | 6 +-
net/netfilter/nfnetlink.c | 2 +-
net/netfilter/nfnetlink_log.c | 3 +-
net/netfilter/nfnetlink_queue.c | 3 +-
net/netlink/af_netlink.c | 106 ++++++++++++++++++++++++++++++++---------
net/netlink/genetlink.c | 4 +-
net/xfrm/xfrm_user.c | 2 +-
security/selinux/netlink.c | 5 +-
22 files changed, 122 insertions(+), 61 deletions(-)

diff --git a/drivers/connector/connector.c b/drivers/connector/connector.c
index a7b9e9b..5690709 100644

```
--- a/drivers/connector/connector.c
+++ b/drivers/connector/connector.c
@@ -446,7 +446,7 @@ static int __devinit cn_init(void)
  dev->id.idx = cn_idx;
  dev->id.val = cn_val;

- dev->nls = netlink_kernel_create(NETLINK_CONNECTOR,
+ dev->nls = netlink_kernel_create(&init_net, NETLINK_CONNECTOR,
       CN_NETLINK_USERS + 0xf,
       dev->input, NULL, THIS_MODULE);
  if (!dev->nls)
diff --git a/drivers/scsi/scsi_netlink.c b/drivers/scsi/scsi_netlink.c
index 4bf9aa5..163acf6 100644
--- a/drivers/scsi/scsi_netlink.c
+++ b/drivers/scsi/scsi_netlink.c
@@ -167,7 +167,7 @@ scsi_netlink_init(void)
   return;
  }

- scsi_nl_sock = netlink_kernel_create(NETLINK_SCSITRANSPORT,
+ scsi_nl_sock = netlink_kernel_create(&init_net, NETLINK_SCSITRANSPORT,
     SCSI_NL_GRP_CNT, scsi_nl_rcv, NULL,
     THIS_MODULE);
  if (!scsi_nl_sock) {
diff --git a/drivers/scsi/scsi_transport_iscsi.c b/drivers/scsi/scsi_transport_iscsi.c
index 34c1860..4916f01 100644
--- a/drivers/scsi/scsi_transport_iscsi.c
+++ b/drivers/scsi/scsi_transport_iscsi.c
@@ -1523,7 +1523,7 @@ static __init int iscsi_transport_init(void)
  if (err)
   goto unregister_conn_class;

- nls = netlink_kernel_create(NETLINK_ISCSI, 1, iscsi_if_rx, NULL,
+ nls = netlink_kernel_create(&init_net, NETLINK_ISCSI, 1, iscsi_if_rx, NULL,
    THIS_MODULE);
  if (!nls) {
   err = -ENOBUFS;
diff --git a/fs/ecryptfs/netlink.c b/fs/ecryptfs/netlink.c
index fe91863..056519c 100644
--- a/fs/ecryptfs/netlink.c
+++ b/fs/ecryptfs/netlink.c
@@ -227,7 +227,7 @@ int ecryptfs_init_netlink(void)
 {
  int rc;

- ecryptfs_nl_sock = netlink_kernel_create(NETLINK_ECRYPTFS, 0,
+ ecryptfs_nl_sock = netlink_kernel_create(&init_net, NETLINK_ECRYPTFS, 0,
        ecryptfs_receive_nl_message,
```

```
        NULL, THIS_MODULE);
  if (!ecryptfs_nl_sock) {
diff --git a/include/linux/netlink.h b/include/linux/netlink.h
index 83d8239..d2843ae 100644
--- a/include/linux/netlink.h
+++ b/include/linux/netlink.h
@@ -27,6 +27,8 @@

 #define MAX_LINKS 32

+struct net;
+
 struct sockaddr_nl
 {
  sa_family_t nl_family; /* AF_NETLINK */
@@ -157,7 +159,8 @@ struct netlink_skb_parms
 #define NETLINK_CREDS(skb) (&NETLINK_CB((skb)).creds)


-extern struct sock *netlink_kernel_create(int unit, unsigned int groups,
+extern struct sock *netlink_kernel_create(struct net *net,
+       int unit,unsigned int groups,
        void (*input)(struct sock *sk, int len),
        struct mutex *cb_mutex,
        struct module *module);
@@ -206,6 +209,7 @@ struct netlink_callback

 struct netlink_notify
 {
+ struct net *net;
  int pid;
  int protocol;
 };
diff --git a/kernel/audit.c b/kernel/audit.c
index eb0f916..f3c390f 100644
--- a/kernel/audit.c
+++ b/kernel/audit.c
@@ -876,8 +876,8 @@ static int __init audit_init(void)

  printk(KERN_INFO "audit: initializing netlink socket (%s)\n",
         audit_default ? "enabled" : "disabled");
- audit_sock = netlink_kernel_create(NETLINK_AUDIT, 0, audit_receive,
-        NULL, THIS_MODULE);
+ audit_sock = netlink_kernel_create(&init_net, NETLINK_AUDIT, 0,
+       audit_receive, NULL, THIS_MODULE);
  if (!audit_sock)
   audit_panic("cannot initialize netlink socket");
  else
```

```
diff --git a/lib/kobject_uevent.c b/lib/kobject_uevent.c
index df02814..e06a8dc 100644
--- a/lib/kobject_uevent.c
+++ b/lib/kobject_uevent.c
@@ -280,9 +280,8 @@ EXPORT_SYMBOL_GPL(add_uevent_var);
 #if defined(CONFIG_NET)
 static int __init kobject_uevent_init(void)
 {
- uevent_sock = netlink_kernel_create(NETLINK_KOBJECT_UEVENT, 1, NULL,
-         NULL, THIS_MODULE);
-
+ uevent_sock = netlink_kernel_create(&init_net, NETLINK_KOBJECT_UEVENT,
+         1, NULL, NULL, THIS_MODULE);
  if (!uevent_sock) {
   printk(KERN_ERR
          "kobject_uevent: unable to create netlink socket!\n");
diff --git a/net/bridge/netfilter/ebt_ulog.c b/net/bridge/netfilter/ebt_ulog.c
index 204c968..e7cfd30 100644
--- a/net/bridge/netfilter/ebt_ulog.c
+++ b/net/bridge/netfilter/ebt_ulog.c
@@ -300,8 +300,9 @@ static int __init ebt_ulog_init(void)
   spin_lock_init(&ulog_buffers[i].lock);
  }

- ebtulognl = netlink_kernel_create(NETLINK_NFLOG, EBT_ULOG_MAXNLGROUPS,
-       NULL, NULL, THIS_MODULE);
+ ebtulognl = netlink_kernel_create(&init_net, NETLINK_NFLOG,
+       EBT_ULOG_MAXNLGROUPS, NULL, NULL,
+       THIS_MODULE);
  if (!ebtulognl)
   ret = -ENOMEM;
  else if ((ret = ebt_register_watcher(&ulog)))
diff --git a/net/core/rtnetlink.c b/net/core/rtnetlink.c
index 4185950..416768d 100644
--- a/net/core/rtnetlink.c
+++ b/net/core/rtnetlink.c
@@ -1327,8 +1327,8 @@ void __init rtnetlink_init(void)
  if (!rta_buf)
   panic("rtnetlink_init: cannot allocate rta_buf\n");

- rtnl = netlink_kernel_create(NETLINK_ROUTE, RTNLGRP_MAX, rtnetlink_rcv,
-         &rtnl_mutex, THIS_MODULE);
+ rtnl = netlink_kernel_create(&init_net, NETLINK_ROUTE, RTNLGRP_MAX,
+         rtnetlink_rcv, &rtnl_mutex, THIS_MODULE);
  if (rtnl == NULL)
   panic("rtnetlink_init: cannot initialize rtnetlink\n");
  netlink_set_nonroot(NETLINK_ROUTE, NL_NONROOT_RECV);
diff --git a/net/decnet/netfilter/dn_rtmsg.c b/net/decnet/netfilter/dn_rtmsg.c
```

```
index 6962346..ebb38fe 100644
--- a/net/decnet/netfilter/dn_rtmsg.c
+++ b/net/decnet/netfilter/dn_rtmsg.c
@@ -137,7 +137,8 @@ static int __init dn_rtmsg_init(void)
 {
  int rv = 0;

- dnrmg = netlink_kernel_create(NETLINK_DNRTMSG, DNRNG_NLGRP_MAX,
+ dnrmg = netlink_kernel_create(&init_net,
+          NETLINK_DNRTMSG, DNRNG_NLGRP_MAX,
           dnrmg_receive_user_sk, NULL, THIS_MODULE);
  if (dnrmg == NULL) {
   printk(KERN_ERR "dn_rtmsg: Cannot create netlink socket");
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index cefb55e..140bf7a 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -816,8 +816,8 @@ static void nl_fib_input(struct sock *sk, int len)

 static void nl_fib_lookup_init(void)
 {
-      netlink_kernel_create(NETLINK_FIB_LOOKUP, 0, nl_fib_input, NULL,
-       THIS_MODULE);
+ netlink_kernel_create(&init_net, NETLINK_FIB_LOOKUP, 0, nl_fib_input,
+    NULL, THIS_MODULE);
 }

 static void fib_disable_ip(struct net_device *dev, int force)
diff --git a/net/ipv4/inet_diag.c b/net/ipv4/inet_diag.c
index 06922da..169026d 100644
--- a/net/ipv4/inet_diag.c
+++ b/net/ipv4/inet_diag.c
@@ -893,8 +893,8 @@ static int __init inet_diag_init(void)
  if (!inet_diag_table)
   goto out;

- idiagnl = netlink_kernel_create(NETLINK_INET_DIAG, 0, inet_diag_rcv,
-     NULL, THIS_MODULE);
+ idiagnl = netlink_kernel_create(&init_net, NETLINK_INET_DIAG, 0,
+     inet_diag_rcv, NULL, THIS_MODULE);
  if (idiagnl == NULL)
   goto out_free_table;
  err = 0;
diff --git a/net/ipv4/netfilter/ip_queue.c b/net/ipv4/netfilter/ip_queue.c
index d918560..82fda92 100644
--- a/net/ipv4/netfilter/ip_queue.c
+++ b/net/ipv4/netfilter/ip_queue.c
@@ -579,7 +579,7 @@ ipq_rcv_nl_event(struct notifier_block *this,
```

```
  if (event == NETLINK_URELEASE &&
      n->protocol == NETLINK_FIREWALL && n->pid) {
    write_lock_bh(&queue_lock);
-   if (n->pid == peer_pid)
+   if ((n->net == &init_net) && (n->pid == peer_pid))
      __ipq_reset();
    write_unlock_bh(&queue_lock);
   }
@@ -671,8 +671,8 @@ static int __init ip_queue_init(void)
   struct proc_dir_entry *proc;

   netlink_register_notifier(&ipq_nl_notifier);
- ipqnl = netlink_kernel_create(NETLINK_FIREWALL, 0, ipq_rcv_sk,
-                               NULL, THIS_MODULE);
+ ipqnl = netlink_kernel_create(&init_net, NETLINK_FIREWALL, 0,
+                               ipq_rcv_sk, NULL, THIS_MODULE);
   if (ipqnl == NULL) {
     printk(KERN_ERR "ip_queue: failed to create netlink socket\n");
     goto cleanup_netlink_notifier;
diff --git a/net/ipv4/netfilter/ipt_ULOG.c b/net/ipv4/netfilter/ipt_ULOG.c
index 6ca43e4..c636d6d 100644
--- a/net/ipv4/netfilter/ipt_ULOG.c
+++ b/net/ipv4/netfilter/ipt_ULOG.c
@@ -409,7 +409,8 @@ static int __init ipt_ulog_init(void)
   for (i = 0; i < ULOG_MAXNLGROUPS; i++)
     setup_timer(&ulog_buffers[i].timer, ulog_timer, i);

- nflognl = netlink_kernel_create(NETLINK_NFLOG, ULOG_MAXNLGROUPS, NULL,
+ nflognl = netlink_kernel_create(&init_net,
+                                 NETLINK_NFLOG, ULOG_MAXNLGROUPS, NULL,
                                  NULL, THIS_MODULE);
   if (!nflognl)
     return -ENOMEM;
diff --git a/net/ipv6/netfilter/ip6_queue.c b/net/ipv6/netfilter/ip6_queue.c
index 64536a3..2f5a524 100644
--- a/net/ipv6/netfilter/ip6_queue.c
+++ b/net/ipv6/netfilter/ip6_queue.c
@@ -569,7 +569,7 @@ ipq_rcv_nl_event(struct notifier_block *this,
   if (event == NETLINK_URELEASE &&
       n->protocol == NETLINK_IP6_FW && n->pid) {
     write_lock_bh(&queue_lock);
-   if (n->pid == peer_pid)
+   if ((n->net == &init_net) && (n->pid == peer_pid))
      __ipq_reset();
     write_unlock_bh(&queue_lock);
   }
@@ -661,8 +661,8 @@ static int __init ip6_queue_init(void)
   struct proc_dir_entry *proc;
```

```
  netlink_register_notifier(&ipq_nl_notifier);
- ipqnl = netlink_kernel_create(NETLINK_IP6_FW, 0, ipq_rcv_sk, NULL,
-          THIS_MODULE);
+ ipqnl = netlink_kernel_create(&init_net, NETLINK_IP6_FW, 0, ipq_rcv_sk,
+      NULL, THIS_MODULE);
  if (ipqnl == NULL) {
   printk(KERN_ERR "ip6_queue: failed to create netlink socket\n");
   goto cleanup_netlink_notifier;
diff --git a/net/netfilter/nfnetlink.c b/net/netfilter/nfnetlink.c
index 8797e69..fa974e8 100644
--- a/net/netfilter/nfnetlink.c
+++ b/net/netfilter/nfnetlink.c
@@ -264,7 +264,7 @@ static int __init nfnetlink_init(void)
 {
  printk("Netfilter messages via NETLINK v%s.\n", nfversion);

- nfnl = netlink_kernel_create(NETLINK_NETFILTER, NFNLGRP_MAX,
+ nfnl = netlink_kernel_create(&init_net, NETLINK_NETFILTER, NFNLGRP_MAX,
          nfnetlink_rcv, NULL, THIS_MODULE);
  if (!nfnl) {
   printk(KERN_ERR "cannot initialize nfnetlink!\n");
diff --git a/net/netfilter/nfnetlink_log.c b/net/netfilter/nfnetlink_log.c
index e185a5b..994fff0 100644
--- a/net/netfilter/nfnetlink_log.c
+++ b/net/netfilter/nfnetlink_log.c
@@ -705,7 +705,8 @@ nfulnl_rcv_nl_event(struct notifier_block *this,

    hlist_for_each_entry_safe(inst, tmp, t2, head, hlist) {
     UDEBUG("node = %p\n", inst);
-    if (n->pid == inst->peer_pid)
+    if ((n->net == &init_net) &&
+        (n->pid == inst->peer_pid))
      __instance_destroy(inst);
    }
   }
diff --git a/net/netfilter/nfnetlink_queue.c b/net/netfilter/nfnetlink_queue.c
index 5a8e8ff..c97369f 100644
--- a/net/netfilter/nfnetlink_queue.c
+++ b/net/netfilter/nfnetlink_queue.c
@@ -765,7 +765,8 @@ nfqnl_rcv_nl_event(struct notifier_block *this,
    struct hlist_head *head = &instance_table[i];

    hlist_for_each_entry_safe(inst, tmp, t2, head, hlist) {
-    if (n->pid == inst->peer_pid)
+    if ((n->net == &init_net) &&
+        (n->pid == inst->peer_pid))
      __instance_destroy(inst);
```

```
   }
  }
diff --git a/net/netlink/af_netlink.c b/net/netlink/af_netlink.c
index 406a493..6726957 100644
--- a/net/netlink/af_netlink.c
+++ b/net/netlink/af_netlink.c
@@ -211,7 +211,7 @@ netlink_unlock_table(void)
   wake_up(&nl_table_wait);
 }

-static __inline__ struct sock *netlink_lookup(int protocol, u32 pid)
+static __inline__ struct sock *netlink_lookup(struct net *net, int protocol, u32 pid)
 {
  struct nl_pid_hash *hash = &nl_table[protocol].hash;
  struct hlist_head *head;
@@ -221,7 +221,7 @@ static __inline__ struct sock *netlink_lookup(int protocol, u32 pid)
  read_lock(&nl_table_lock);
  head = nl_pid_hashfn(hash, pid);
  sk_for_each(sk, node, head) {
-  if (nlk_sk(sk)->pid == pid) {
+  if ((sk->sk_net == net) && (nlk_sk(sk)->pid == pid)) {
    sock_hold(sk);
    goto found;
   }
@@ -328,7 +328,7 @@ netlink_update_listeners(struct sock *sk)
   * makes sure updates are visible before bind or setsockopt return. */
 }

-static int netlink_insert(struct sock *sk, u32 pid)
+static int netlink_insert(struct sock *sk, struct net *net, u32 pid)
 {
  struct nl_pid_hash *hash = &nl_table[sk->sk_protocol].hash;
  struct hlist_head *head;
@@ -341,7 +341,7 @@ static int netlink_insert(struct sock *sk, u32 pid)
  head = nl_pid_hashfn(hash, pid);
  len = 0;
  sk_for_each(osk, node, head) {
-  if (nlk_sk(osk)->pid == pid)
+  if ((osk->sk_net == net) && (nlk_sk(osk)->pid == pid))
    break;
   len++;
  }
@@ -419,9 +419,6 @@ static int netlink_create(struct net *net, struct socket *sock, int protocol)
  struct netlink_sock *nlk;
  int err = 0;

- if (net != &init_net)
-  return -EAFNOSUPPORT;
```

```
-
  sock->state = SS_UNCONNECTED;

  if (sock->type != SOCK_RAW && sock->type != SOCK_DGRAM)
@@ -481,6 +478,7 @@ static int netlink_release(struct socket *sock)

  if (nlk->pid && !nlk->subscriptions) {
   struct netlink_notify n = {
+      .net = sk->sk_net,
       .protocol = sk->sk_protocol,
       .pid = nlk->pid,
       };
@@ -509,6 +507,7 @@ static int netlink_release(struct socket *sock)
 static int netlink_autobind(struct socket *sock)
 {
  struct sock *sk = sock->sk;
+ struct net *net = sk->sk_net;
  struct nl_pid_hash *hash = &nl_table[sk->sk_protocol].hash;
  struct hlist_head *head;
  struct sock *osk;
@@ -522,6 +521,8 @@ retry:
  netlink_table_grab();
  head = nl_pid_hashfn(hash, pid);
  sk_for_each(osk, node, head) {
+  if ((osk->sk_net != net))
+   continue;
   if (nlk_sk(osk)->pid == pid) {
    /* Bind collision, search negative pid values. */
    pid = rover--;
@@ -533,7 +534,7 @@ retry:
  }
  netlink_table_ungrab();

- err = netlink_insert(sk, pid);
+ err = netlink_insert(sk, net, pid);
  if (err == -EADDRINUSE)
   goto retry;

@@ -598,6 +599,7 @@ static int netlink_realloc_groups(struct sock *sk)
 static int netlink_bind(struct socket *sock, struct sockaddr *addr, int addr_len)
 {
  struct sock *sk = sock->sk;
+ struct net *net = sk->sk_net;
  struct netlink_sock *nlk = nlk_sk(sk);
  struct sockaddr_nl *nladdr = (struct sockaddr_nl *)addr;
  int err;
@@ -619,7 +621,7 @@ static int netlink_bind(struct socket *sock, struct sockaddr *addr, int
addr_len
```

```
    return -EINVAL;
  } else {
    err = nladdr->nl_pid ?
-   netlink_insert(sk, nladdr->nl_pid) :
+   netlink_insert(sk, net, nladdr->nl_pid) :
     netlink_autobind(sock);
    if (err)
     return err;
@@ -703,10 +705,12 @@ static void netlink_overrun(struct sock *sk)
 static struct sock *netlink_getsockbypid(struct sock *ssk, u32 pid)
 {
  int protocol = ssk->sk_protocol;
+ struct net *net;
  struct sock *sock;
  struct netlink_sock *nlk;

- sock = netlink_lookup(protocol, pid);
+ net = ssk->sk_net;
+ sock = netlink_lookup(net, protocol, pid);
  if (!sock)
   return ERR_PTR(-ECONNREFUSED);

@@ -887,6 +891,7 @@ static __inline__ int netlink_broadcast_deliver(struct sock *sk, struct
sk_buff

 struct netlink_broadcast_data {
  struct sock *exclude_sk;
+ struct net *net;
  u32 pid;
  u32 group;
  int failure;
@@ -909,6 +914,9 @@ static inline int do_one_broadcast(struct sock *sk,
      !test_bit(p->group - 1, nlk->groups))
   goto out;

+ if ((sk->sk_net != p->net))
+  goto out;
+
  if (p->failure) {
   netlink_overrun(sk);
   goto out;
@@ -947,6 +955,7 @@ out:
 int netlink_broadcast(struct sock *ssk, struct sk_buff *skb, u32 pid,
        u32 group, gfp_t allocation)
 {
+ struct net *net = ssk->sk_net;
  struct netlink_broadcast_data info;
  struct hlist_node *node;
```

```
  struct sock *sk;
@@ -954,6 +963,7 @@ int netlink_broadcast(struct sock *ssk, struct sk_buff *skb, u32 pid,
  skb = netlink_trim(skb, allocation);

  info.exclude_sk = ssk;
+ info.net = net;
  info.pid = pid;
  info.group = group;
  info.failure = 0;
@@ -1002,6 +1012,9 @@ static inline int do_one_set_err(struct sock *sk,
  if (sk == p->exclude_sk)
   goto out;

+ if (sk->sk_net != p->exclude_sk->sk_net)
+  goto out;
+
  if (nlk->pid == p->pid || p->group - 1 >= nlk->ngroups ||
      !test_bit(p->group - 1, nlk->groups))
   goto out;
@@ -1304,7 +1317,7 @@ static void netlink_data_ready(struct sock *sk, int len)
  */

 struct sock *
-netlink_kernel_create(int unit, unsigned int groups,
+netlink_kernel_create(struct net *net, int unit, unsigned int groups,
         void (*input)(struct sock *sk, int len),
         struct mutex *cb_mutex, struct module *module)
 {
@@ -1321,7 +1334,7 @@ netlink_kernel_create(int unit, unsigned int groups,
  if (sock_create_lite(PF_NETLINK, SOCK_DGRAM, unit, &sock))
   return NULL;

- if (__netlink_create(&init_net, sock, cb_mutex, unit) < 0)
+ if (__netlink_create(net, sock, cb_mutex, unit) < 0)
   goto out_sock_release;

  if (groups < 32)
@@ -1336,18 +1349,20 @@ netlink_kernel_create(int unit, unsigned int groups,
  if (input)
   nlk_sk(sk)->data_ready = input;

- if (netlink_insert(sk, 0))
+ if (netlink_insert(sk, net, 0))
   goto out_sock_release;

  nlk = nlk_sk(sk);
  nlk->flags |= NETLINK_KERNEL_SOCKET;
```

```
  netlink_table_grab();
- nl_table[unit].groups = groups;
- nl_table[unit].listeners = listeners;
- nl_table[unit].cb_mutex = cb_mutex;
- nl_table[unit].module = module;
- nl_table[unit].registered = 1;
+ if (!nl_table[unit].registered) {
+  nl_table[unit].groups = groups;
+  nl_table[unit].listeners = listeners;
+  nl_table[unit].cb_mutex = cb_mutex;
+  nl_table[unit].module = module;
+  nl_table[unit].registered = 1;
+ }
  netlink_table_ungrab();

  return sk;
@@ -1513,7 +1528,7 @@ int netlink_dump_start(struct sock *ssk, struct sk_buff *skb,
  atomic_inc(&skb->users);
  cb->skb = skb;

- sk = netlink_lookup(ssk->sk_protocol, NETLINK_CB(skb).pid);
+ sk = netlink_lookup(ssk->sk_net, ssk->sk_protocol, NETLINK_CB(skb).pid);
  if (sk == NULL) {
   netlink_destroy_callback(cb);
   return -ECONNREFUSED;
@@ -1555,7 +1570,8 @@ void netlink_ack(struct sk_buff *in_skb, struct nlmsghdr *nlh, int err)
  if (!skb) {
   struct sock *sk;

-  sk = netlink_lookup(in_skb->sk->sk_protocol,
+  sk = netlink_lookup(in_skb->sk->sk_net,
+        in_skb->sk->sk_protocol,
         NETLINK_CB(in_skb).pid);
   if (sk) {
    sk->sk_err = ENOBUFS;
@@ -1706,6 +1722,7 @@ int nlmsg_notify(struct sock *sk, struct sk_buff *skb, u32 pid,

 #ifdef CONFIG_PROC_FS
 struct nl_seq_iter {
+ struct net *net;
  int link;
  int hash_idx;
 };
@@ -1723,6 +1740,8 @@ static struct sock *netlink_seq_socket_idx(struct seq_file *seq, loff_t
pos)

   for (j = 0; j <= hash->mask; j++) {
    sk_for_each(s, node, &hash->table[j]) {
```

```
+    if (iter->net != s->sk_net)
+     continue;
     if (off == pos) {
      iter->link = i;
      iter->hash_idx = j;
@@ -1752,11 +1771,14 @@ static void *netlink_seq_next(struct seq_file *seq, void *v, loff_t
*pos)
  if (v == SEQ_START_TOKEN)
   return netlink_seq_socket_idx(seq, 0);

- s = sk_next(v);
+ iter = seq->private;
+ s = v;
+ do {
+  s = sk_next(s);
+ } while (s && (iter->net != s->sk_net));
  if (s)
   return s;

- iter = seq->private;
  i = iter->link;
  j = iter->hash_idx + 1;

@@ -1765,6 +1787,8 @@ static void *netlink_seq_next(struct seq_file *seq, void *v, loff_t *pos)

   for (; j <= hash->mask; j++) {
    s = sk_head(&hash->table[j]);
+   while (s && (iter->net != s->sk_net))
+    s = sk_next(s);
    if (s) {
     iter->link = i;
     iter->hash_idx = j;
@@ -1835,15 +1859,24 @@ static int netlink_seq_open(struct inode *inode, struct file *file)

  seq = file->private_data;
  seq->private = iter;
+ iter->net = get_net(PROC_NET(inode));
  return 0;
 }

+static int netlink_seq_release(struct inode *inode, struct file *file)
+{
+ struct seq_file *seq = file->private_data;
+ struct nl_seq_iter *iter = seq->private;
+ put_net(iter->net);
+ return seq_release_private(inode, file);
+}
+
```

```
 static const struct file_operations netlink_seq_fops = {
  .owner  = THIS_MODULE,
  .open  = netlink_seq_open,
  .read  = seq_read,
  .llseek  = seq_lseek,
- .release = seq_release_private,
+ .release = netlink_seq_release,
 };

 #endif
@@ -1885,6 +1918,27 @@ static struct net_proto_family netlink_family_ops = {
  .owner = THIS_MODULE, /* for consistency 8) */
 };

+static int netlink_net_init(struct net *net)
+{
+#ifdef CONFIG_PROC_FS
+ if (!proc_net_fops_create(net, "netlink", 0, &netlink_seq_fops))
+  return -ENOMEM;
+#endif
+ return 0;
+}
+
+static void netlink_net_exit(struct net *net)
+{
+#ifdef CONFIG_PROC_FS
+ proc_net_remove(net, "netlink");
+#endif
+}
+
+static struct pernet_operations netlink_net_ops = {
+ .init = netlink_net_init,
+ .exit = netlink_net_exit,
+};
+
 static int __init netlink_proto_init(void)
 {
  struct sk_buff *dummy_skb;
@@ -1930,10 +1984,8 @@ static int __init netlink_proto_init(void)
  }

  sock_register(&netlink_family_ops);
-#ifdef CONFIG_PROC_FS
- proc_net_fops_create(&init_net, "netlink", 0, &netlink_seq_fops);
-#endif
- /* The netlink device handler may be needed early. */
+ register_pernet_subsys(&netlink_net_ops);
+ /* The netlink device handler may be needed early. */
```

```
  rtnetlink_init();
 out:
  return err;
diff --git a/net/netlink/genetlink.c b/net/netlink/genetlink.c
index 8c11ca4..af8fe26 100644
--- a/net/netlink/genetlink.c
+++ b/net/netlink/genetlink.c
@@ -782,8 +782,8 @@ static int __init genl_init(void)
  netlink_set_nonroot(NETLINK_GENERIC, NL_NONROOT_RECV);

  /* we'll bump the group number right afterwards */
- genl_sock = netlink_kernel_create(NETLINK_GENERIC, 0, genl_rcv,
-       NULL, THIS_MODULE);
+ genl_sock = netlink_kernel_create(&init_net, NETLINK_GENERIC, 0,
+       genl_rcv, NULL, THIS_MODULE);
  if (genl_sock == NULL)
   panic("GENL: Cannot initialize generic netlink\n");

diff --git a/net/xfrm/xfrm_user.c b/net/xfrm/xfrm_user.c
index 46076f5..2035d62 100644
--- a/net/xfrm/xfrm_user.c
+++ b/net/xfrm/xfrm_user.c
@@ -2391,7 +2391,7 @@ static int __init xfrm_user_init(void)

  printk(KERN_INFO "Initializing XFRM netlink socket\n");

- nlsk = netlink_kernel_create(NETLINK_XFRM, XFRMNLGRP_MAX,
+ nlsk = netlink_kernel_create(&init_net, NETLINK_XFRM, XFRMNLGRP_MAX,
          xfrm_netlink_rcv, NULL, THIS_MODULE);
  if (nlsk == NULL)
   return -ENOMEM;
diff --git a/security/selinux/netlink.c b/security/selinux/netlink.c
index f49046d..b59871d 100644
--- a/security/selinux/netlink.c
+++ b/security/selinux/netlink.c
@@ -17,6 +17,7 @@
 #include <linux/skbuff.h>
 #include <linux/netlink.h>
 #include <linux/selinux_netlink.h>
+#include <net/net_namespace.h>

 static struct sock *selnl;

@@ -104,8 +105,8 @@ void selnl_notify_policyload(u32 seqno)

 static int __init selnl_init(void)
 {
- selnl = netlink_kernel_create(NETLINK_SELINUX, SELNLGRP_MAX, NULL, NULL,
```

```
-                               THIS_MODULE);
+ selnl = netlink_kernel_create(&init_net, NETLINK_SELINUX,
+          SELNLGRP_MAX, NULL, NULL, THIS_MODULE);
  if (selnl == NULL)
   panic("SELinux:  Cannot create netlink socket.");
  netlink_set_nonroot(NETLINK_SELINUX, NL_NONROOT_RECV);
--
1.5.3.rc6.17.g1911
```

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers