
Subject: [PATCH RFC 14/31] net: Support multiple network namespaces with netlink

Posted by [ebiederm](#) on Thu, 25 Jan 2007 19:00:16 GMT

[View Forum Message](#) <> [Reply to Message](#)

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

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/scsi/scsi_netlink.c      | 2 +-
drivers/scsi/scsi_transport_iscsi.c | 2 +-
include/linux/netlink.h         | 3 +-
kernel/audit.c                  | 4 +-
lib/kobject_uevent.c            | 4 +-
net/bridge/netfilter/ebt_ulog.c | 5 +-
net/core/rtnetlink.c            | 4 +-
net/decnnet/netfilter/dn_rtmmsg.c | 3 +-
net/ipv4/fib_frontend.c          | 3 +-
net/ipv4/inet_diag.c            | 4 +-
net/ipv4/netfilter/ip_queue.c    | 6 +-
net/ipv4/netfilter/ipt_ULOG.c    | 4 +-
net/ipv6/netfilter/ip6_queue.c   | 4 +-
net/netfilter/nfnetlink.c        | 2 +-
net/netfilter/nfnetlink_log.c    | 3 +-
net/netfilter/nfnetlink_queue.c  | 3 +-
net/netlink/af_netlink.c         | 104 ++++++-----
net/netlink/genetlink.c          | 4 +-
net/xfrm/xfrm_user.c             | 2 +-
19 files changed, 112 insertions(+), 54 deletions(-)
```

```
diff --git a/drivers/scsi/scsi_netlink.c b/drivers/scsi/scsi_netlink.c
index 1b59b27..02c2c1e 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, THIS_MODULE);
if (!scsi_nl_sock) {
    printk(KERN_ERR "%s: register of recieve handler failed\n",
diff --git a/drivers/scsi/scsi_transport_iscsi.c b/drivers/scsi/scsi_transport_iscsi.c
index 9c22f13..1ad22c2 100644
--- a/drivers/scsi/scsi_transport_iscsi.c
+++ b/drivers/scsi/scsi_transport_iscsi.c
@@ -1435,7 +1435,7 @@ static __init int iscsi_transport_init(void)
    if (err)
        goto unregister_conn_class;

- nls = netlink_kernel_create(NETLINK_ISCSI, 1, iscsi_if_rx,
+ nls = netlink_kernel_create(init_net(), NETLINK_ISCSI, 1, iscsi_if_rx,
    THIS_MODULE);
if (!nls) {
    err = -ENOBUFFS;
diff --git a/include/linux/netlink.h b/include/linux/netlink.h
index b3b9b60..9dacd00 100644
--- a/include/linux/netlink.h
+++ b/include/linux/netlink.h
@@ -151,7 +151,7 @@ struct netlink_skb_parms
#define NETLINK_CREDS(skb) (&NETLINK_CB((skb)).creds)

-extern struct sock *netlink_kernel_create(int unit, unsigned int groups, void (*input)(struct sock
*sk, int len), struct module *module);
+extern struct sock *netlink_kernel_create(net_t net, int unit, unsigned int groups, void
(*input)(struct sock *sk, int len), struct module *module);
extern void netlink_ack(struct sk_buff *in_skb, struct nlmsg_hdr *nlh, int err);
extern int netlink_has_listeners(struct sock *sk, unsigned int group);
extern int netlink_unicast(struct sock *ssk, struct sk_buff *skb, __u32 pid, int nonblock);
@@ -188,6 +188,7 @@ struct netlink_callback

struct netlink_notify
{
+ net_t net;
    int pid;
    int protocol;
};
diff --git a/kernel/audit.c b/kernel/audit.c

```

index d9b690a..b0c5c61 100644

--- a/kernel/audit.c

+++ b/kernel/audit.c

@@ -696,8 +696,8 @@ 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,
-   THIS_MODULE);
+ audit_sock = netlink_kernel_create(init_net(), NETLINK_AUDIT, 0,
+   audit_receive, 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 84272ed..9a5d4ca 100644

--- a/lib/kobject_uevent.c

+++ b/lib/kobject_uevent.c

@@ -292,8 +292,8 @@ 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,
-   THIS_MODULE);
+ uevent_sock = netlink_kernel_create(init_net(), NETLINK_KOBJECT_UEVENT, 1,
+   NULL, THIS_MODULE);
```

```
if (!uevent_sock) {
    printk(KERN_ERR
```

diff --git a/net/bridge/netfilter/ebt_ulog.c b/net/bridge/netfilter/ebt_ulog.c

index c1af68b..abf2be7 100644

--- a/net/bridge/netfilter/ebt_ulog.c

+++ b/net/bridge/netfilter/ebt_ulog.c

@@ -301,8 +301,9 @@ static int __init ebt_ulog_init(void)

```
    spin_lock_init(&ulog_buffers[i].lock);
}
```

```
- ebtulognl = netlink_kernel_create(NETLINK_NFLOG, EBT_ULOG_MAXNLGROUPS,
-   NULL, THIS_MODULE);
```

```
+ ebtulognl = netlink_kernel_create(init_net(), NETLINK_NFLOG,
+   EBT_ULOG_MAXNLGROUPS, 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 7841e89..8f3dda8 100644

--- a/net/core/rtnetlink.c

```

+++ b/net/core/rtnetlink.c
@@ -870,8 +870,8 @@ 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,
-                               THIS_MODULE);
+ rtnl = netlink_kernel_create(init_net(), NETLINK_ROUTE, RTNLGRP_MAX,
+                               rtnetlink_rcv, 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 8b99bd3..14089ed 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;

- dnrng = netlink_kernel_create(NETLINK_DNRTMSG, DNRNG_NLGRP_MAX,
+ dnrng = netlink_kernel_create(init_net(),
+                               NETLINK_DNRTMSG, DNRNG_NLGRP_MAX,
+                               dnrng_receive_user_sk, THIS_MODULE);
    if (dnrng == 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 049c370..d1859ff 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -817,7 +817,8 @@ 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, THIS_MODULE);
+ netlink_kernel_create(init_net(), NETLINK_FIB_LOOKUP, 0, nl_fib_input,
+ 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 77761ac..bdf3064 100644
--- a/net/ipv4/inet_diag.c
+++ b/net/ipv4/inet_diag.c
@@ -913,8 +913,8 @@ static int __init inet_diag_init(void)
    if (!inet_diag_table)
        goto out;

```

```

- idiagnl = netlink_kernel_create(NETLINK_INET_DIAG, 0, inet_diag_rcv,
-   THIS_MODULE);
+ idiagnl = netlink_kernel_create(init_net(), NETLINK_INET_DIAG, 0,
+   inet_diag_rcv, 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 8650a57..d1c42b5 100644
--- a/net/ipv4/netfilter/ip_queue.c
+++ b/net/ipv4/netfilter/ip_queue.c
@@ -589,7 +589,7 @@ 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 (net_eq(n->net, init_net()) && (n->pid == peer_pid))
        __ipq_reset();
        write_unlock_bh(&queue_lock);
    }
@@ -681,8 +681,8 @@ 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,
-   THIS_MODULE);
+   ipqnl = netlink_kernel_create(init_net(), NETLINK_FIREWALL, 0,
+   ipq_rcv_sk, 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 dbd3478..8071d15 100644
--- a/net/ipv4/netfilter/ipt_ULOG.c
+++ b/net/ipv4/netfilter/ipt_ULOG.c
@@ -395,8 +395,8 @@ static int __init ipt_ulog_init(void)
    ulog_buffers[i].timer.data = i;
}

-   nflognl = netlink_kernel_create(NETLINK_NFLOG, ULOG_MAXNLGROUPS, NULL,
-   THIS_MODULE);
+   nflognl = netlink_kernel_create(init_net(), NETLINK_NFLOG,
+   ULOG_MAXNLGROUPS, NULL, THIS_MODULE);
if (!nflognl)
    return -ENOMEM;

diff --git a/net/ipv6/netfilter/ip6_queue.c b/net/ipv6/netfilter/ip6_queue.c
index f6e108c..02589b2 100644

```

```

--- a/net/ipv6/netfilter/ip6_queue.c
+++ b/net/ipv6/netfilter/ip6_queue.c
@@ -579,7 +579,7 @@ 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 (net_eq(n->net, init_net()) && (n->pid == peer_pid))
        __ipq_reset();
        write_unlock_bh(&queue_lock);
    }
@@ -671,7 +671,7 @@ 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,
+   ipqnl = netlink_kernel_create(init_net(), NETLINK_IP6_FW, 0, ipq_rcv_sk,
                                THIS_MODULE);
    if (ipqnl == NULL) {
        printk(KERN_ERR "ip6_queue: failed to create netlink socket\n");
diff --git a/net/netfilter/nfnetlink.c b/net/netfilter/nfnetlink.c
index 52fdfa2..180353f 100644
--- a/net/netfilter/nfnetlink.c
+++ b/net/netfilter/nfnetlink.c
@@ -356,7 +356,7 @@ 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, 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 d1505dd..1bdf9af 100644
--- a/net/netfilter/nfnetlink_log.c
+++ b/net/netfilter/nfnetlink_log.c
@@ -741,7 +741,8 @@ nfnl_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 (net_eq(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 59bf595..8e0c289 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 (net_eq(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 7433e71..37b6c28 100644
--- a/net/netlink/af_netlink.c
+++ b/net/netlink/af_netlink.c
@@ -200,7 +200,7 @@ 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(net_t net, int protocol, u32 pid)
{
    struct nl_pid_hash *hash = &nl_table[protocol].hash;
    struct hlist_head *head;
@@ -210,7 +210,7 @@ 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 (net_eq(sk->sk_net, net) && (nlk_sk(sk)->pid == pid)) {
        sock_hold(sk);
        goto found;
    }
}
@@ -315,7 +315,7 @@ 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, net_t net, u32 pid)
{
    struct nl_pid_hash *hash = &nl_table[sk->sk_protocol].hash;
    struct hlist_head *head;
@@ -328,7 +328,7 @@ 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 (net_eq(osk->sk_net, net) && (nlk_sk(osk)->pid == pid))
    break;
    len++;
}
@@ -400,9 +400,6 @@ static int netlink_create(net_t net, struct socket *sock, int protocol)
    unsigned int groups;
    int err = 0;

- if (!net_eq(net, init_net()))
- return -EAFNOSUPPORT;
-
    sock->state = SS_UNCONNECTED;

    if (sock->type != SOCK_RAW && sock->type != SOCK_DGRAM)
@@ -469,6 +466,7 @@ 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,
    };
@@ -497,6 +495,7 @@ static int netlink_release(struct socket *sock)
static int netlink_autobind(struct socket *sock)
{
    struct sock *sk = sock->sk;
+ net_t net = sk->sk_net;
    struct nl_pid_hash *hash = &nl_table[sk->sk_protocol].hash;
    struct hlist_head *head;
    struct sock *osk;
@@ -510,6 +509,8 @@ retry:
    netlink_table_grab();
    head = nl_pid_hashfn(hash, pid);
    sk_for_each(osk, node, head) {
+ if (!net_eq(osk->sk_net, net))
+ continue;
        if (nlk_sk(osk)->pid == pid) {
            /* Bind collision, search negative pid values. */
            pid = rover--;
@@ -521,7 +522,7 @@ retry:
    }
    netlink_table_ungrab();

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

```

```

@@ -575,6 +576,7 @@ static int netlink_alloc_groups(struct sock *sk)
static int netlink_bind(struct socket *sock, struct sockaddr *addr, int addr_len)
{
    struct sock *sk = sock->sk;
+ net_t net = sk->sk_net;
    struct netlink_sock *nlk = nlk_sk(sk);
    struct sockaddr_nl *nladdr = (struct sockaddr_nl *)addr;
    int err;
@@ -598,7 +600,7 @@ 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;
@@ -682,10 +684,12 @@ static void netlink_overrun(struct sock *sk)
static struct sock *netlink_getsockbypid(struct sock *ssk, u32 pid)
{
    int protocol = ssk->sk_protocol;
+ net_t 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);

@@ -858,6 +862,7 @@ static __inline__ int netlink_broadcast_deliver(struct sock *sk, struct
sk_buff

struct netlink_broadcast_data {
    struct sock *exclude_sk;
+ net_t net;
    u32 pid;
    u32 group;
    int failure;
@@ -880,6 +885,9 @@ static inline int do_one_broadcast(struct sock *sk,
!test_bit(p->group - 1, nlk->groups))
    goto out;

+ if (!net_eq(sk->sk_net, p->net))
+ goto out;

```

```

+
if (p->failure) {
    netlink_overrun(sk);
    goto out;
@@ -918,6 +926,7 @@ out:
int netlink_broadcast(struct sock *ssk, struct sk_buff *skb, u32 pid,
    u32 group, gfp_t allocation)
{
+ net_t net = ssk->sk_net;
    struct netlink_broadcast_data info;
    struct hlist_node *node;
    struct sock *sk;
@@ -925,6 +934,7 @@ 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;
@@ -973,6 +983,9 @@ static inline int do_one_set_err(struct sock *sk,
    if (sk == p->exclude_sk)
        goto out;

+ if (!net_eq(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;
@@ -1267,7 +1280,7 @@ static void netlink_data_ready(struct sock *sk, int len)
    */

    struct sock *
-netlink_kernel_create(int unit, unsigned int groups,
+netlink_kernel_create(net_t net, int unit, unsigned int groups,
    void (*input)(struct sock *sk, int len),
    struct module *module)
{
@@ -1284,7 +1297,7 @@ 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, unit) < 0)
+ if (__netlink_create(net, sock, unit) < 0)
    goto out_sock_release;

    if (groups < 32)

```

```

@@ -1299,17 +1312,19 @@ 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].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].module = module;
+     nl_table[unit].registered = 1;
+ }
    netlink_table_ungrab();

    return sk;
@@ -1410,7 +1425,7 @@ 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;
@@ -1447,7 +1462,8 @@ void netlink_ack(struct sk_buff *in_skb, struct nlmsg_hdr *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;
@@ -1585,6 +1601,7 @@ int nlmsg_notify(struct sock *sk, struct sk_buff *skb, u32 pid,

#ifdef CONFIG_PROC_FS
struct nl_seq_iter {

```

```

+ net_t net;
+ int link;
+ int hash_idx;
+ };
@@ -1602,6 +1619,8 @@ 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 (!net_eq(iter->net, s->sk_net))
+             continue;
            if (off == pos) {
                iter->link = i;
                iter->hash_idx = j;
@@ -1630,12 +1649,15 @@ 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 && !net_eq(iter->net, s->sk_net));
    if (s)
        return s;

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

@@ -1644,6 +1666,8 @@ 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 && !net_eq(iter->net, s->sk_net))
+             s = sk_next(s);
        if (s) {
            iter->link = i;
            iter->hash_idx = j;
@@ -1714,15 +1738,24 @@ 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 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
@@ -1764,6 +1797,27 @@ static struct net_proto_family netlink_family_ops = {
    .owner = THIS_MODULE, /* for consistency 8) */
};

+static int netlink_net_init(net_t 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(net_t 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;
@@ -1809,9 +1863,7 @@ 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
+ register_pernet_subsys(&netlink_net_ops);
    /* The netlink device handler may be needed early. */
    rtnetlink_init();
out:
diff --git a/net/netlink/genetlink.c b/net/netlink/genetlink.c
index 548e4e6..d6717e8 100644
--- a/net/netlink/genetlink.c
+++ b/net/netlink/genetlink.c
@@ -585,8 +585,8 @@ static int __init genl_init(void)
    goto errout_register;

    netlink_set_nonroot(NETLINK_GENERIC, NL_NONROOT_RECV);
- genl_sock = netlink_kernel_create(NETLINK_GENERIC, GENL_MAX_ID,
-    genl_rcv, THIS_MODULE);
+ genl_sock = netlink_kernel_create(init_net(), NETLINK_GENERIC,
+    GENL_MAX_ID, genl_rcv, 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 82f36d3..55affa7 100644
--- a/net/xfrm/xfrm_user.c
+++ b/net/xfrm/xfrm_user.c
@@ -2293,7 +2293,7 @@ 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, THIS_MODULE);
    if (nlsk == NULL)
        return -ENOMEM;
--
1.4.4.1.g278f

```

Containers mailing list
Containers@lists.osdl.org
<https://lists.osdl.org/mailman/listinfo/containers>
