Subject: [PATCH 0/4 net-2.6.25] Proper netlink kernel sockets disposal.
Posted by den on Fri, 18 Jan 2008 12:51:47 GMT
View Forum Message <> Reply to Message

Alexey Dobriyan found, that virtualized netlink kernel sockets (fibl &
rtnl) are leaked during namespace start/stop loop.

Leaking fix (simple and obvious) reveals that netlink kernel socket
disposal leads to OOPSes:
- nl_table[protocol]->listeners is double freed
- sometimes during namespace stop netlink_sock_destruct
  BUG_TRAP(!atomic_read(&sk->sk_rmem_alloc)); is hit

This set address all these issues.

Signed-off-by: Denis V. Lunev <den@openvz.org>
Tested-by: Alexey Dobriayn <adobriyan@openvz.org>

Subject: [PATCH] [NETNS 1/4 net-2.6.25] Double free in netlink_release.
Posted by den on Fri, 18 Jan 2008 12:53:13 GMT
View Forum Message <> Reply to Message

Netlink protocol table is global for all namespaces. Some netlink protocols
have been virtualized, i.e. they have per/namespace netlink socket. This
difference can easily lead to double free if more than 1 namespace is
started. Count the number of kernel netlink sockets to track that this
table is not used any more.

Signed-off-by: Denis V. Lunev <den@openvz.org>
Tested-by: Alexey Dobriyan <adobriyan@openvz.org>
---
 net/netlink/af_netlink.c | 10 +++++++---
 1 files changed, 7 insertions(+), 3 deletions(-)

diff --git a/net/netlink/af_netlink.c b/net/netlink/af_netlink.c
index 21f9e30..29fef55 100644
--- a/net/netlink/af_netlink.c
+++ b/net/netlink/af_netlink.c
@@ -498,9 +498,12 @@ static int netlink_release(struct socket *sock)

  netlink_table_grab();
  if (netlink_is_kernel(sk)) {
- kfree(nl_table[sk->sk_protocol].listeners);
- nl_table[sk->sk_protocol].module = NULL;
- nl_table[sk->sk_protocol].registered = 0;
+ BUG_ON(nl_table[sk->sk_protocol].registered == 0);
+ if (--nl_table[sk->sk_protocol].registered == 0) {

```
+   kfree(nl_table[sk->sk_protocol].listeners);
+   nl_table[sk->sk_protocol].module = NULL;
+   nl_table[sk->sk_protocol].registered = 0;
+  }
  } else if (nlk->subscriptions)
   netlink_update_listeners(sk);
  netlink_table_ungrab();
@@ -1389,6 +1392,7 @@ netlink_kernel_create(struct net *net, int unit, unsigned int groups,
   nl_table[unit].registered = 1;
  } else {
   kfree(listeners);
+  nl_table[unit].registered++;
  }
  netlink_table_ungrab();

--
1.5.3.rc5
```

Subject: [PATCH] [NETNS 2/4 net-2.6.25] Memory leak on network namespace stop.

Posted by den on Fri, 18 Jan 2008 12:53:14 GMT

View Forum Message <> Reply to Message

Network namespace allocates 2 kernel netlink sockets, fibnl & rtnl. These sockets should be disposed properly, i.e. by sock_release. Plain sock_put is not enough.

Signed-off-by: Denis V. Lunev <den@openvz.org>
Tested-by: Alexey Dobriyan <adobriyan@openvz.org>

```
---
 net/core/rtnetlink.c    |    2 +-
 net/ipv4/fib_frontend.c |    2 +-
 2 files changed, 2 insertions(+), 2 deletions(-)

diff --git a/net/core/rtnetlink.c b/net/core/rtnetlink.c
index 4a07e83..2c1f665 100644
--- a/net/core/rtnetlink.c
+++ b/net/core/rtnetlink.c
@@ -1381,7 +1381,7 @@ static void rtnetlink_net_exit(struct net *net)
     * free.
     */
   sk->sk_net = get_net(&init_net);
-  sock_put(sk);
+  sock_release(net->rtnl->sk_socket);
   net->rtnl = NULL;
  }
 }
```

```
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index 8ddcd3f..4e5216e 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -881,7 +881,7 @@ static void nl_fib_lookup_exit(struct net *net)
   * initial network namespace. So the socket will  be safe to free.
   */
  net->ipv4.fibnl->sk_net = get_net(&init_net);
- sock_put(net->ipv4.fibnl);
+ sock_release(net->ipv4.fibnl->sk_socket);
 }

 static void fib_disable_ip(struct net_device *dev, int force)
--
1.5.3.rc5
```

Subject: [PATCH] [NETNS 3/4 net-2.6.25] Consolidate kernel netlink socket destruction.

Posted by den on Fri, 18 Jan 2008 12:53:15 GMT

View Forum Message <> Reply to Message

Create a specific helper for netlink kernel socket disposal. This just
let the code look better and provides a ground for proper disposal inside
a namespace.

Signed-off-by: Denis V. Lunev <den@openvz.org>
Tested-by: Alexey Dobriyan <adobriyan@openvz.org>

```
---
 drivers/connector/connector.c       |    9 +++------
 drivers/scsi/scsi_netlink.c         |    2 +-
 drivers/scsi/scsi_transport_iscsi.c |    2 +-
 fs/ecryptfs/netlink.c               |    3 +--
 include/linux/netlink.h             |    1 +
 net/bridge/netfilter/ebt_ulog.c     |    4 ++--
 net/core/rtnetlink.c                |    2 +-
 net/decnet/netfilter/dn_rtmsg.c     |    4 ++--
 net/ipv4/fib_frontend.c             |    2 +-
 net/ipv4/inet_diag.c                |    2 +-
 net/ipv4/netfilter/ip_queue.c       |    4 ++--
 net/ipv4/netfilter/ipt_ULOG.c       |    4 ++--
 net/ipv6/netfilter/ip6_queue.c      |    4 ++--
 net/netfilter/nfnetlink.c           |    2 +-
 net/netlink/af_netlink.c            |   11 ++++++++++
 net/xfrm/xfrm_user.c                |    2 +-
 16 files changed, 33 insertions(+), 25 deletions(-)

diff --git a/drivers/connector/connector.c b/drivers/connector/connector.c
```

```
index 37976dc..fea2d3e 100644
--- a/drivers/connector/connector.c
+++ b/drivers/connector/connector.c
@@ -420,8 +420,7 @@ static int __devinit cn_init(void)

  dev->cbdev = cn_queue_alloc_dev("cqueue", dev->nls);
  if (!dev->cbdev) {
-  if (dev->nls->sk_socket)
-   sock_release(dev->nls->sk_socket);
+  netlink_kernel_release(dev->nls);
   return -EINVAL;
  }

@@ -431,8 +430,7 @@ static int __devinit cn_init(void)
  if (err) {
   cn_already_initialized = 0;
   cn_queue_free_dev(dev->cbdev);
-  if (dev->nls->sk_socket)
-   sock_release(dev->nls->sk_socket);
+  netlink_kernel_release(dev->nls);
   return -EINVAL;
  }

@@ -447,8 +445,7 @@ static void __devexit cn_fini(void)

  cn_del_callback(&dev->id);
  cn_queue_free_dev(dev->cbdev);
- if (dev->nls->sk_socket)
-  sock_release(dev->nls->sk_socket);
+ netlink_kernel_release(dev->nls);
 }

 subsys_initcall(cn_init);
diff --git a/drivers/scsi/scsi_netlink.c b/drivers/scsi/scsi_netlink.c
index 40579ed..fe48c24 100644
--- a/drivers/scsi/scsi_netlink.c
+++ b/drivers/scsi/scsi_netlink.c
@@ -169,7 +169,7 @@ void
 scsi_netlink_exit(void)
 {
  if (scsi_nl_sock) {
-  sock_release(scsi_nl_sock->sk_socket);
+  netlink_kernel_release(scsi_nl_sock);
   netlink_unregister_notifier(&scsi_netlink_notifier);
  }

diff --git a/drivers/scsi/scsi_transport_iscsi.c b/drivers/scsi/scsi_transport_iscsi.c
index 5428d15..9e463a6 100644
```

```
--- a/drivers/scsi/scsi_transport_iscsi.c
+++ b/drivers/scsi/scsi_transport_iscsi.c
@@ -1533,7 +1533,7 @@ unregister_transport_class:

 static void __exit iscsi_transport_exit(void)
 {
- sock_release(nls->sk_socket);
+ netlink_kernel_release(nls);
  transport_class_unregister(&iscsi_connection_class);
  transport_class_unregister(&iscsi_session_class);
  transport_class_unregister(&iscsi_host_class);
diff --git a/fs/ecryptfs/netlink.c b/fs/ecryptfs/netlink.c
index 9aa3451..f638a69 100644
--- a/fs/ecryptfs/netlink.c
+++ b/fs/ecryptfs/netlink.c
@@ -237,7 +237,6 @@ out:
  */
 void ecryptfs_release_netlink(void)
 {
- if (ecryptfs_nl_sock && ecryptfs_nl_sock->sk_socket)
-  sock_release(ecryptfs_nl_sock->sk_socket);
+ netlink_kernel_release(ecryptfs_nl_sock);
  ecryptfs_nl_sock = NULL;
 }
diff --git a/include/linux/netlink.h b/include/linux/netlink.h
index 2aee0f5..bd13b6f 100644
--- a/include/linux/netlink.h
+++ b/include/linux/netlink.h
@@ -178,6 +178,7 @@ extern struct sock *netlink_kernel_create(struct net *net,
        void (*input)(struct sk_buff *skb),
        struct mutex *cb_mutex,
        struct module *module);
+extern void netlink_kernel_release(struct sock *sk);
 extern int netlink_change_ngroups(struct sock *sk, unsigned int groups);
 extern void netlink_clear_multicast_users(struct sock *sk, unsigned int group);
 extern void netlink_ack(struct sk_buff *in_skb, struct nlmsghdr *nlh, int err);
diff --git a/net/bridge/netfilter/ebt_ulog.c b/net/bridge/netfilter/ebt_ulog.c
index b73ba28..8e7b00b 100644
--- a/net/bridge/netfilter/ebt_ulog.c
+++ b/net/bridge/netfilter/ebt_ulog.c
@@ -307,7 +307,7 @@ static int __init ebt_ulog_init(void)
  if (!ebtulognl)
   ret = -ENOMEM;
  else if ((ret = ebt_register_watcher(&ulog)))
-  sock_release(ebtulognl->sk_socket);
+  netlink_kernel_release(ebtulognl);

  if (ret == 0)
```

```
   nf_log_register(PF_BRIDGE, &ebt_ulog_logger);
@@ -333,7 +333,7 @@ static void __exit ebt_ulog_fini(void)
   }
   spin_unlock_bh(&ub->lock);
  }
- sock_release(ebtulognl->sk_socket);
+ netlink_kernel_release(ebtulognl);
 }

 module_init(ebt_ulog_init);
diff --git a/net/core/rtnetlink.c b/net/core/rtnetlink.c
index 2c1f665..2ef9480 100644
--- a/net/core/rtnetlink.c
+++ b/net/core/rtnetlink.c
@@ -1381,7 +1381,7 @@ static void rtnetlink_net_exit(struct net *net)
    * free.
    */
   sk->sk_net = get_net(&init_net);
-  sock_release(net->rtnl->sk_socket);
+  netlink_kernel_release(net->rtnl);
   net->rtnl = NULL;
  }
 }
diff --git a/net/decnet/netfilter/dn_rtmsg.c b/net/decnet/netfilter/dn_rtmsg.c
index 96375f2..6d2bd32 100644
--- a/net/decnet/netfilter/dn_rtmsg.c
+++ b/net/decnet/netfilter/dn_rtmsg.c
@@ -137,7 +137,7 @@ static int __init dn_rtmsg_init(void)

  rv = nf_register_hook(&dnrmg_ops);
  if (rv) {
-  sock_release(dnrmg->sk_socket);
+  netlink_kernel_release(dnrmg);
  }

  return rv;
@@ -146,7 +146,7 @@ static int __init dn_rtmsg_init(void)
 static void __exit dn_rtmsg_fini(void)
 {
  nf_unregister_hook(&dnrmg_ops);
- sock_release(dnrmg->sk_socket);
+ netlink_kernel_release(dnrmg);
 }


diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index 4e5216e..e787d21 100644
--- a/net/ipv4/fib_frontend.c
```

```
+++ b/net/ipv4/fib_frontend.c
@@ -881,7 +881,7 @@ static void nl_fib_lookup_exit(struct net *net)
   * initial network namespace. So the socket will  be safe to free.
   */
  net->ipv4.fibnl->sk_net = get_net(&init_net);
- sock_release(net->ipv4.fibnl->sk_socket);
+ netlink_kernel_release(net->ipv4.fibnl);
 }

 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 e468e7a..605ed2c 100644
--- a/net/ipv4/inet_diag.c
+++ b/net/ipv4/inet_diag.c
@@ -935,7 +935,7 @@ out_free_table:

 static void __exit inet_diag_exit(void)
 {
- sock_release(idiagnl->sk_socket);
+ netlink_kernel_release(idiagnl);
  kfree(inet_diag_table);
 }

diff --git a/net/ipv4/netfilter/ip_queue.c b/net/ipv4/netfilter/ip_queue.c
index 7361315..5109839 100644
--- a/net/ipv4/netfilter/ip_queue.c
+++ b/net/ipv4/netfilter/ip_queue.c
@@ -605,7 +605,7 @@ cleanup_sysctl:
  unregister_netdevice_notifier(&ipq_dev_notifier);
  proc_net_remove(&init_net, IPQ_PROC_FS_NAME);
 cleanup_ipqnl:
- sock_release(ipqnl->sk_socket);
+ netlink_kernel_release(ipqnl);
  mutex_lock(&ipqnl_mutex);
  mutex_unlock(&ipqnl_mutex);

@@ -624,7 +624,7 @@ static void __exit ip_queue_fini(void)
  unregister_netdevice_notifier(&ipq_dev_notifier);
  proc_net_remove(&init_net, IPQ_PROC_FS_NAME);

- sock_release(ipqnl->sk_socket);
+ netlink_kernel_release(ipqnl);
  mutex_lock(&ipqnl_mutex);
  mutex_unlock(&ipqnl_mutex);

diff --git a/net/ipv4/netfilter/ipt_ULOG.c b/net/ipv4/netfilter/ipt_ULOG.c
index fa24efa..b192756 100644
--- a/net/ipv4/netfilter/ipt_ULOG.c
```

```
+++ b/net/ipv4/netfilter/ipt_ULOG.c
@@ -415,7 +415,7 @@ static int __init ulog_tg_init(void)

  ret = xt_register_target(&ulog_tg_reg);
  if (ret < 0) {
-  sock_release(nflognl->sk_socket);
+  netlink_kernel_release(nflognl);
   return ret;
  }
  if (nflog)
@@ -434,7 +434,7 @@ static void __exit ulog_tg_exit(void)
  if (nflog)
   nf_log_unregister(&ipt_ulog_logger);
  xt_unregister_target(&ulog_tg_reg);
- sock_release(nflognl->sk_socket);
+ netlink_kernel_release(nflognl);

  /* remove pending timers and free allocated skb's */
  for (i = 0; i < ULOG_MAXNLGROUPS; i++) {
diff --git a/net/ipv6/netfilter/ip6_queue.c b/net/ipv6/netfilter/ip6_queue.c
index a20db0b..56b4ea6 100644
--- a/net/ipv6/netfilter/ip6_queue.c
+++ b/net/ipv6/netfilter/ip6_queue.c
@@ -609,7 +609,7 @@ cleanup_sysctl:
  proc_net_remove(&init_net, IPQ_PROC_FS_NAME);

 cleanup_ipqnl:
- sock_release(ipqnl->sk_socket);
+ netlink_kernel_release(ipqnl);
  mutex_lock(&ipqnl_mutex);
  mutex_unlock(&ipqnl_mutex);

@@ -628,7 +628,7 @@ static void __exit ip6_queue_fini(void)
  unregister_netdevice_notifier(&ipq_dev_notifier);
  proc_net_remove(&init_net, IPQ_PROC_FS_NAME);

- sock_release(ipqnl->sk_socket);
+ netlink_kernel_release(ipqnl);
  mutex_lock(&ipqnl_mutex);
  mutex_unlock(&ipqnl_mutex);

diff --git a/net/netfilter/nfnetlink.c b/net/netfilter/nfnetlink.c
index 2128542..b75c9c4 100644
--- a/net/netfilter/nfnetlink.c
+++ b/net/netfilter/nfnetlink.c
@@ -179,7 +179,7 @@ static void nfnetlink_rcv(struct sk_buff *skb)
 static void __exit nfnetlink_exit(void)
 {
```

```
  printk("Removing netfilter NETLINK layer.\n");
- sock_release(nfnl->sk_socket);
+ netlink_kernel_release(nfnl);
  return;
 }

diff --git a/net/netlink/af_netlink.c b/net/netlink/af_netlink.c
index 29fef55..626a582 100644
--- a/net/netlink/af_netlink.c
+++ b/net/netlink/af_netlink.c
@@ -1405,6 +1405,17 @@ out_sock_release:
 }
 EXPORT_SYMBOL(netlink_kernel_create);

+
+void
+netlink_kernel_release(struct sock *sk)
+{
+ if (sk == NULL || sk->sk_socket == NULL)
+  return;
+ sock_release(sk->sk_socket);
+}
+EXPORT_SYMBOL(netlink_kernel_release);
+
+
 /**
  * netlink_change_ngroups - change number of multicast groups
  *
diff --git a/net/xfrm/xfrm_user.c b/net/xfrm/xfrm_user.c
index 35fc16a..e0ccdf2 100644
--- a/net/xfrm/xfrm_user.c
+++ b/net/xfrm/xfrm_user.c
@@ -2420,7 +2420,7 @@ static void __exit xfrm_user_exit(void)
  xfrm_unregister_km(&netlink_mgr);
  rcu_assign_pointer(xfrm_nl, NULL);
  synchronize_rcu();
- sock_release(nlsk->sk_socket);
+ netlink_kernel_release(nlsk);
 }

 module_init(xfrm_user_init);
--
1.5.3.rc5
```

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

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);
- }
+ 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
```

Subject: Re: [PATCH 0/4 net-2.6.25] Proper netlink kernel sockets disposal.
Posted by davem on Sat, 19 Jan 2008 07:55:43 GMT
View Forum Message <> Reply to Message

From: "Denis V. Lunev" <den@sw.ru>
Date: Fri, 18 Jan 2008 15:51:47 +0300

> Alexey Dobriyan found, that virtualized netlink kernel sockets (fibl &
> rtnl) are leaked during namespace start/stop loop.
>
> Leaking fix (simple and obvious) reveals that netlink kernel socket
> disposal leads to OOPSes:
> - nl_table[protocol]->listeners is double freed
> - sometimes during namespace stop netlink_sock_destruct
>   BUG_TRAP(!atomic_read(&sk->sk_rmem_alloc)); is hit
>
> This set address all these issues.
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>
> Tested-by: Alexey Dobriayn <adobriyan@openvz.org>

All 4 patches applied, thanks!
