Subject: [patch 00/12] net namespace : L3 namespace - introduction
Posted by Daniel Lezcano on Fri, 19 Jan 2007 15:47:14 GMT
View Forum Message <> Reply to Message

This patchset provide a network isolation similar at what
Linux-Vserver provides. It is based on the L2 namespaces and relies on
the mechanisms provided by the namespace. This L3 namespaces does not
aim to bring full virtualization for the network, it provides an IP
isolation which can be reused for Linux-Vserver, jailed application or
application containers.

A L3 namespace are always L2 s' childs and they can not create more
network namespaces, furthermore, they lose their NET_ADMIN
capability. They share their parent's network ressources. From the
parent namespace, IP addresses are created and assigned to the
different L3 childs. From this point, L3 namespaces can use their
assigned IP address and all computed broadcast addresses.

Because the L3 namespace relies on the L2 virtualization mechanisms,
it is possible to have several L3 namespaces listening on
INADDR_ANY:port without conflict, that's allow to run several server
without modifying the network configuration.

The loopback is a shared device between all L3 namespaces. To ensure
the 127.0.0.1 address isolation, the sender store its namespace into
the packet, so when the packet arrives, the destination namespace is
already set, because "source" == "destination". By this way, it is
easy to disable the loopback isolation and let the application to talk
with application outside of the namespace via the 127.0.0.1 because we
consider them trusted (like portmap).

The ifconfig / ip commands will only show IP addresses assigned to the
L3 namespace. When a L3 namespace dies, the assigned IP address is
released to its parent.

At the IP level, when a packet arrives, the L3 network namespace
destination is retrieved from the destination address.

At the bind time, the address is checked against the assigned IP
address.

--

Subject: [patch 01/12] net namespace : initialize init process to level 2
Posted by Daniel Lezcano on Fri, 19 Jan 2007 15:47:15 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

Initialize the init's network namespace to level 2

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

---
 net/core/net_namespace.c | 1 +
 1 file changed, 1 insertion(+)

Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
===
--- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
+++ 2.6.20-rc4-mm1/net/core/net_namespace.c
@@ -21,6 +21,7 @@
  .dev_tail_p = &init_net_ns.dev_base_p,
  .loopback_dev_p = NULL,
  .pcpu_lstats_p = NULL,
+ .level   = NET_NS_LEVEL2,
 };

 #ifdef CONFIG_NET_NS

--

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

Subject: [patch 02/12] net namespace : store L2 parent namespace
Posted by Daniel Lezcano on Fri, 19 Jan 2007 15:47:16 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

All L3 namespaces are the final nodes of the L2 namespaces
tree. Because their share some ressources coming from the L2
namespace. The L2 parent namespace should be stored into the L3 child
when it is created.

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

```
 include/linux/net_namespace.h |    1 +
 net/core/net_namespace.c      |   11 ++++++++++
 2 files changed, 12 insertions(+)

Index: 2.6.20-rc4-mm1/include/linux/net_namespace.h
===================================================================
--- 2.6.20-rc4-mm1.orig/include/linux/net_namespace.h
+++ 2.6.20-rc4-mm1/include/linux/net_namespace.h
@@ -27,6 +27,7 @@
 #define NET_NS_LEVEL2  1
 #define NET_NS_LEVEL3  2
  unsigned int  level;
+ struct net_namespace    *parent;
 };

 extern struct net_namespace init_net_ns;
Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
+++ 2.6.20-rc4-mm1/net/core/net_namespace.c
@@ -22,6 +22,7 @@
  .loopback_dev_p = NULL,
  .pcpu_lstats_p = NULL,
  .level          = NET_NS_LEVEL2,
+ .parent         = NULL,
 };

 #ifdef CONFIG_NET_NS
@@ -62,6 +63,12 @@
   if (ip_fib_struct_init())
    goto out_fib4;
  }
+
+ if (level == NET_NS_LEVEL3) {
+  get_net_ns(old_ns);
+  ns->parent = old_ns;
+ }
+
  ns->level = level;
  if (loopback_init())
   goto out_loopback;
@@ -126,8 +133,12 @@
     ns, atomic_read(&ns->kref.refcount));
   return;
  }
+
  if (ns->level == NET_NS_LEVEL2)
   ip_fib_struct_cleanup(ns);
```

```
+ if (ns->level == NET_NS_LEVEL3)
+  put_net_ns(ns->parent);
+
  printk(KERN_DEBUG "NET_NS: net namespace %p destroyed\n", ns);
  kfree(ns);
 }
```

--

_____

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

Subject: [patch 03/12] net namespace : share network ressources L2 with L3

Posted by Daniel Lezcano on Fri, 19 Jan 2007 15:47:17 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

L3 namespace will use routes and devices belonging to its parent, so
the old network namespace structure is copied when allocating a new
one. By this way, hash value, dev list, routes are accessible from the
L3 namespaces. In case of L2 namespace, these values are overwritten
by the newly allocated values.

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

```
---
 include/linux/net_namespace.h |   14 +++++++++++++
 net/core/dev.c                |    4 ++--
 net/core/net_namespace.c      |   33 ++++++++++++++++++++++++++++++---------
 3 files changed, 34 insertions(+), 17 deletions(-)

Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
+++ 2.6.20-rc4-mm1/net/core/net_namespace.c
@@ -37,7 +37,7 @@
  * Return ERR_PTR on error, new ns otherwise
  */
 static struct net_namespace *clone_net_ns(unsigned int level,
-      struct net_namespace *old_ns)
+       struct net_namespace *old_ns)
 {
  struct net_namespace *ns;

@@ -45,23 +45,26 @@
```

```
  if (current_net_ns->level == NET_NS_LEVEL3)
   return ERR_PTR(-EPERM);

- ns = kzalloc(sizeof(struct net_namespace), GFP_KERNEL);
+ ns = kmemdup(old_ns, sizeof(struct net_namespace), GFP_KERNEL);
  if (!ns)
   return NULL;

  kref_init(&ns->kref);
- ns->dev_base_p = NULL;
- ns->dev_tail_p = &ns->dev_base_p;
- ns->hash = net_random();
-
  if ((push_net_ns(ns)) != old_ns)
+
   BUG();
  if (level ==  NET_NS_LEVEL2) {
+  ns->dev_base_p = NULL;
+  ns->dev_tail_p = &ns->dev_base_p;
+  ns->hash = net_random();
+
 #ifdef CONFIG_IP_MULTIPLE_TABLES
   INIT_LIST_HEAD(&ns->fib_rules_ops_list);
 #endif
   if (ip_fib_struct_init())
    goto out_fib4;
+  if (loopback_init())
+   goto out_loopback;
  }

  if (level == NET_NS_LEVEL3) {
@@ -70,8 +73,6 @@
  }

  ns->level = level;
- if (loopback_init())
-  goto out_loopback;
  pop_net_ns(old_ns);
  printk(KERN_DEBUG "NET_NS: created new netcontext %p, level %u, "
   "for %s (pid=%d)\n", ns, (ns->level == NET_NS_LEVEL2) ?
@@ -127,15 +128,17 @@
  struct net_namespace *ns;

  ns = container_of(kref, struct net_namespace, kref);
- unregister_netdev(ns->loopback_dev_p);
- if (ns->dev_base_p != NULL) {
-  printk("NET_NS: BUG: namespace %p has devices! ref %d\n",
-    ns, atomic_read(&ns->kref.refcount));
```

```
-  return;
- }

- if (ns->level == NET_NS_LEVEL2)
+ if (ns->level == NET_NS_LEVEL2) {
   ip_fib_struct_cleanup(ns);
+  unregister_netdev(ns->loopback_dev_p);
+  if (ns->dev_base_p != NULL) {
+   printk("NET_NS: BUG: namespace %p has devices! ref %d\n",
+          ns, atomic_read(&ns->kref.refcount));
+   return;
+  }
+ }
+
  if (ns->level == NET_NS_LEVEL3)
   put_net_ns(ns->parent);

Index: 2.6.20-rc4-mm1/include/linux/net_namespace.h
===================================================================
--- 2.6.20-rc4-mm1.orig/include/linux/net_namespace.h
+++ 2.6.20-rc4-mm1/include/linux/net_namespace.h
@@ -56,6 +56,15 @@
 DECLARE_PER_CPU(struct net_namespace *, exec_net_ns);
 #define current_net_ns  (__get_cpu_var(exec_net_ns))

+static inline struct net_namespace *net_ns_l2(void)
+{
+ struct net_namespace *net_ns = current_net_ns;
+
+ if (net_ns->level == NET_NS_LEVEL3)
+  return net_ns->parent;
+ return net_ns;
+}
+
 static inline void init_current_net_ns(int cpu)
 {
  get_net_ns(&init_net_ns);
@@ -110,6 +119,11 @@

 #define current_net_ns   NULL

+static inline struct net_namespace *net_ns_l2(void)
+{
+ return NULL;
+}
+
 static inline void init_current_net_ns(int cpu)
 {
```

```
 }
Index: 2.6.20-rc4-mm1/net/core/dev.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/core/dev.c
+++ 2.6.20-rc4-mm1/net/core/dev.c
@@ -485,7 +485,7 @@
 struct net_device *__dev_get_by_name(const char *name)
 {
  struct hlist_node *p;
- struct net_namespace *ns = current_net_ns;
+ struct net_namespace *ns = net_ns_l2();

  hlist_for_each(p, dev_name_hash(name, ns)) {
   struct net_device *dev
@@ -768,7 +768,7 @@
  if (!err) {
   hlist_del(&dev->name_hlist);
   hlist_add_head(&dev->name_hlist, dev_name_hash(dev->name,
-       current_net_ns));
+              net_ns_l2()));
   raw_notifier_call_chain(&netdev_chain,
     NETDEV_CHANGENAME, dev);
  }

--

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers
```

Subject: [patch 04/12] net namespace : isolate the inet device.
Posted by Daniel Lezcano on Fri, 19 Jan 2007 15:47:18 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

ip and ifconfig commands will not show ip addr
not belonging to the current network namespace.

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

```
---
 include/linux/inetdevice.h |    1 +
 net/ipv4/devinet.c         |   22 ++++++++++++++++++++-
 2 files changed, 22 insertions(+), 1 deletion(-)

Index: 2.6.20-rc4-mm1/include/linux/inetdevice.h
```

```
===================================================================
--- 2.6.20-rc4-mm1.orig/include/linux/inetdevice.h
+++ 2.6.20-rc4-mm1/include/linux/inetdevice.h
@@ -99,6 +99,7 @@
  unsigned char  ifa_flags;
  unsigned char  ifa_prefixlen;
  char   ifa_label[IFNAMSIZ];
+ struct net_namespace    *ifa_net_ns;
 };

 extern int register_inetaddr_notifier(struct notifier_block *nb);
Index: 2.6.20-rc4-mm1/net/ipv4/devinet.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/ipv4/devinet.c
+++ 2.6.20-rc4-mm1/net/ipv4/devinet.c
@@ -53,6 +53,7 @@
 #include <linux/notifier.h>
 #include <linux/inetdevice.h>
 #include <linux/igmp.h>
+#include <linux/net_namespace.h>
 #ifdef CONFIG_SYSCTL
 #include <linux/sysctl.h>
 #endif
@@ -269,6 +270,7 @@

    if (!(ifa->ifa_flags & IFA_F_SECONDARY) ||
        ifa1->ifa_mask != ifa->ifa_mask ||
+       !net_ns_match(ifa->ifa_net_ns, ifa1->ifa_net_ns) ||
        !inet_ifa_match(ifa1->ifa_address, ifa)) {
     ifap1 = &ifa->ifa_next;
     prev_prom = ifa;
@@ -471,6 +473,9 @@

  for (ifap = &in_dev->ifa_list; (ifa = *ifap) != NULL;
       ifap = &ifa->ifa_next) {
+  if (!net_ns_match(ifa->ifa_net_ns, current_net_ns))
+   continue;
+
   if (tb[IFA_LOCAL] &&
       ifa->ifa_local != nla_get_be32(tb[IFA_LOCAL]))
    continue;
@@ -544,6 +549,7 @@
  ifa->ifa_flags = ifm->ifa_flags;
  ifa->ifa_scope = ifm->ifa_scope;
  ifa->ifa_dev = in_dev;
+ ifa->ifa_net_ns = current_net_ns;

  ifa->ifa_local = nla_get_be32(tb[IFA_LOCAL]);
```

```
  ifa->ifa_address = nla_get_be32(tb[IFA_ADDRESS]);
@@ -689,6 +695,8 @@
     for (ifap = &in_dev->ifa_list; (ifa = *ifap) != NULL;
          ifap = &ifa->ifa_next) {
      if (!strcmp(ifr.ifr_name, ifa->ifa_label) &&
+         net_ns_match(ifa->ifa_net_ns,
+        current_net_ns) &&
         sin_orig.sin_addr.s_addr ==
        ifa->ifa_address) {
       break; /* found */
@@ -701,11 +709,16 @@
   if (!ifa) {
     for (ifap = &in_dev->ifa_list; (ifa = *ifap) != NULL;
          ifap = &ifa->ifa_next)
-    if (!strcmp(ifr.ifr_name, ifa->ifa_label))
+     if (!strcmp(ifr.ifr_name, ifa->ifa_label) &&
+                                net_ns_match(ifa->ifa_net_ns,
+                                             current_net_ns))
       break;
   }
  }

+ if (ifa && !net_ns_match(ifa->ifa_net_ns, current_net_ns))
+  goto done;
+
  ret = -EADDRNOTAVAIL;
  if (!ifa && cmd != SIOCSIFADDR && cmd != SIOCSIFFLAGS)
   goto done;
@@ -749,6 +762,8 @@
    ret = -ENOBUFS;
    if ((ifa = inet_alloc_ifa()) == NULL)
     break;
+
+   ifa->ifa_net_ns = current_net_ns;
    if (colon)
     memcpy(ifa->ifa_label, ifr.ifr_name, IFNAMSIZ);
    else
@@ -853,6 +868,8 @@
   goto out;

  for (; ifa; ifa = ifa->ifa_next) {
+  if (!net_ns_match(ifa->ifa_net_ns, current_net_ns))
+   continue;
   if (!buf) {
    done += sizeof(ifr);
    continue;
@@ -1086,6 +1103,7 @@
     in_dev_hold(in_dev);
```

```
     ifa->ifa_dev = in_dev;
     ifa->ifa_scope = RT_SCOPE_HOST;
+    ifa->ifa_net_ns = current_net_ns;
     memcpy(ifa->ifa_label, dev->name, IFNAMSIZ);
     inet_insert_ifa(ifa);
    }
@@ -1198,6 +1216,8 @@

   for (ifa = in_dev->ifa_list, ip_idx = 0; ifa;
        ifa = ifa->ifa_next, ip_idx++) {
+   if (!net_ns_match(ifa->ifa_net_ns, current_net_ns))
+    continue;
    if (ip_idx < s_ip_idx)
     continue;
    if (inet_fill_ifaddr(skb, ifa, NETLINK_CB(cb->skb).pid,

--
```

Subject: [patch 05/12] net namespace : ioctl to push ifa to net namespace l3
Posted by Daniel Lezcano on Fri, 19 Jan 2007 15:47:19 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

New ioctl to "push" ifaddr to a container. Actually, the push is done
from the current namespace, so the right word is "pull". That will be
changed to move ifaddr from l2 network namespace to l3.

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

```
---
 include/linux/net_namespace.h |    7 ++
 include/linux/sockios.h       |    4 +
 net/core/net_namespace.c      |  118 ++++++++++++++++++++++++++++++++++++++++++-
 net/ipv4/af_inet.c            |    4 +
 4 files changed, 132 insertions(+), 1 deletion(-)

Index: 2.6.20-rc4-mm1/include/linux/sockios.h
===================================================================
--- 2.6.20-rc4-mm1.orig/include/linux/sockios.h
+++ 2.6.20-rc4-mm1/include/linux/sockios.h
@@ -122,6 +122,10 @@
 #define SIOCBRADDIF 0x89a2  /* add interface to bridge      */
```

```
 #define SIOCBRDELIF 0x89a3  /* remove interface from bridge */

+/* Container calls */
+#define SIOCNETNSPUSHIF  0x89b0         /* add ifaddr to namespace      */
+#define SIOCNETNSPULLIF  0x89b1         /* remove ifaddr to namespace   */
+
 /* Device private ioctl calls */

 /*
Index: 2.6.20-rc4-mm1/net/ipv4/af_inet.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/ipv4/af_inet.c
+++ 2.6.20-rc4-mm1/net/ipv4/af_inet.c
@@ -789,6 +789,10 @@
   case SIOCSIFFLAGS:
    err = devinet_ioctl(cmd, (void __user *)arg);
    break;
+         case SIOCNETNSPUSHIF:
+         case SIOCNETNSPULLIF:
+   err = net_ns_ioctl(cmd, (void __user *)arg);
+   break;
   default:
    if (sk->sk_prot->ioctl)
     err = sk->sk_prot->ioctl(sk, cmd, arg);
Index: 2.6.20-rc4-mm1/include/linux/net_namespace.h
===================================================================
--- 2.6.20-rc4-mm1.orig/include/linux/net_namespace.h
+++ 2.6.20-rc4-mm1/include/linux/net_namespace.h
@@ -91,6 +91,8 @@

 #define net_ns_hash(ns)   ((ns)->hash)

+extern int net_ns_ioctl(unsigned int cmd, void __user *arg);
+
 #else /* CONFIG_NET_NS */

 #define INIT_NET_NS(net_ns)
@@ -141,6 +143,11 @@

 #define net_ns_hash(ns)   (0)

+static inline int net_ns_ioctl(unsigned int cmd, void __user *arg)
+{
+ return -ENOSYS;
+}
+
 #endif /* !CONFIG_NET_NS */
```

```
 #endif /* _LINUX_NET_NAMESPACE_H */
Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
+++ 2.6.20-rc4-mm1/net/core/net_namespace.c
@@ -10,7 +10,9 @@
 #include <linux/nsproxy.h>
 #include <linux/net_namespace.h>
 #include <linux/net.h>
+#include <linux/in.h>
 #include <linux/netdevice.h>
+#include <linux/inetdevice.h>
 #include <net/ip_fib.h>

 struct net_namespace init_net_ns = {
@@ -123,6 +125,33 @@
  return err;
 }

+/*
+ * The function will move the ifaddr to the l2 network namespace
+ * parent.
+ * @net_ns: the related network namespace
+ */
+static void release_ifa_to_parent(const struct net_namespace* net_ns)
+{
+ struct net_device *dev;
+ struct in_device *in_dev;
+
+ read_lock(&dev_base_lock);
+ rcu_read_lock();
+ for (dev = dev_base; dev; dev = dev->next) {
+  in_dev = __in_dev_get_rcu(dev);
+  if (!in_dev)
+   continue;
+
+  for_ifa(in_dev) {
+   if (ifa->ifa_net_ns != net_ns)
+    continue;
+   ifa->ifa_net_ns = net_ns->parent;
+  } endfor_ifa(in_dev);
+ }
+ read_unlock(&dev_base_lock);
+ rcu_read_unlock();
+}
+
 void free_net_ns(struct kref *kref)
 {
```

```
  struct net_namespace *ns;
@@ -139,12 +168,99 @@
    }
   }

- if (ns->level == NET_NS_LEVEL3)
+ if (ns->level == NET_NS_LEVEL3) {
+  release_ifa_to_parent(ns);
    put_net_ns(ns->parent);
+ }

   printk(KERN_DEBUG "NET_NS: net namespace %p destroyed\n", ns);
   kfree(ns);
  }
  EXPORT_SYMBOL_GPL(free_net_ns);

+/*
+ * This function allows to assign an IP address from a l2 network
+ * namespace to one of his l3 child or to release from an l3 network
+ * namespace to his l2 network namespace parent.
+ * @cmd: a "push" / "pull" command
+ * @arg: an userspace buffer containing an ifreq structure
+ * Returns:
+ * - EPERM : if caller has no CAP_NET_ADMIN capabilities or the
+ *           current level of network namespace is not layer 2
+ * - EFAULT : if arg is an invalid buffer
+ * - EADDRNOTAVAIL : if the specified ifaddr does not exists
+ * - EINVAL : if cmd is unknown
+ * - zero on success
+ */
+int net_ns_ioctl(unsigned int cmd, void __user *arg)
+{
+ struct ifreq ifr;
+ struct sockaddr_in *sin = (struct sockaddr_in *)&ifr.ifr_addr;
+ struct net_namespace *net_ns = current_net_ns;
+ struct net_device *dev;
+ struct in_device *in_dev;
+ struct in_ifaddr **ifap = NULL;
+ struct in_ifaddr *ifa = NULL;
+ char *colon;
+ int err;
+
+ if (!capable(CAP_NET_ADMIN))
+  return -EPERM;
+
+ if (net_ns->level != NET_NS_LEVEL3)
+  return -EPERM;
+
```

```
+ if (copy_from_user(&ifr, arg, sizeof(struct ifreq)))
+  return -EFAULT;
+
+ ifr.ifr_name[IFNAMSIZ - 1] = 0;
+
+
+ colon = strchr(ifr.ifr_name, ':');
+ if (colon)
+  *colon = 0;
+
+ rtnl_lock();
+
+ err = -ENODEV;
+ dev = __dev_get_by_name(ifr.ifr_name);
+ if (!dev)
+  goto out;
+
+ if (colon)
+  *colon = ':';
+
+ err = -EADDRNOTAVAIL;
+ in_dev = __in_dev_get_rtnl(dev);
+ if (!in_dev)
+  goto out;
+
+ for (ifap = &in_dev->ifa_list; (ifa = *ifap) != NULL;
+      ifap = &ifa->ifa_next)
+  if (!strcmp(ifr.ifr_name, ifa->ifa_label) &&
+      sin->sin_addr.s_addr == ifa->ifa_local)
+   break;
+ if (!ifa)
+  goto out;
+
+ err = -EINVAL;
+ switch(cmd) {
+
+ case SIOCNETNSPUSHIF:
+  ifa->ifa_net_ns = net_ns;
+  break;
+
+ case SIOCNETNSPULLIF:
+  ifa->ifa_net_ns = net_ns->parent;
+  break;
+ default:
+  goto out;
+ }
+
+ err = 0;
```

```
+out:
+ rtnl_unlock();
+ return err;
+}
+
 #endif /* CONFIG_NET_NS */

--
```

Subject: [patch 06/12] net namespace : check bind address
Posted by Daniel Lezcano on Fri, 19 Jan 2007 15:47:20 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

Check the bind address is allowed. It must match ifaddr assigned to
the namespace and all derivative addresses.

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

```
---
 include/linux/net_namespace.h |    7 +++++
 net/core/net_namespace.c      |   54 ++++++++++++++++++++++++++++++++++++++++++
 net/ipv4/af_inet.c            |    2 +
 net/ipv4/raw.c                |    3 ++
 4 files changed, 66 insertions(+)

Index: 2.6.20-rc4-mm1/net/ipv4/af_inet.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/ipv4/af_inet.c
+++ 2.6.20-rc4-mm1/net/ipv4/af_inet.c
@@ -433,6 +433,8 @@
   *  is temporarily down)
   */
  err = -EADDRNOTAVAIL;
+ if (net_ns_check_bind(chk_addr_ret, addr->sin_addr.s_addr))
+  goto out;
  if (!sysctl_ip_nonlocal_bind &&
      !inet->freebind &&
      addr->sin_addr.s_addr != INADDR_ANY &&
Index: 2.6.20-rc4-mm1/net/ipv4/raw.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/ipv4/raw.c
```

```
+++ 2.6.20-rc4-mm1/net/ipv4/raw.c
@@ -559,7 +559,10 @@
  if (sk->sk_state != TCP_CLOSE || addr_len < sizeof(struct sockaddr_in))
   goto out;
  chk_addr_ret = inet_addr_type(addr->sin_addr.s_addr);
+
  ret = -EADDRNOTAVAIL;
+ if (net_ns_check_bind(chk_addr_ret, addr->sin_addr.s_addr))
+  goto out;
  if (addr->sin_addr.s_addr && chk_addr_ret != RTN_LOCAL &&
      chk_addr_ret != RTN_MULTICAST && chk_addr_ret != RTN_BROADCAST)
   goto out;
Index: 2.6.20-rc4-mm1/include/linux/net_namespace.h
===================================================================
--- 2.6.20-rc4-mm1.orig/include/linux/net_namespace.h
+++ 2.6.20-rc4-mm1/include/linux/net_namespace.h
@@ -93,6 +93,8 @@

 extern int net_ns_ioctl(unsigned int cmd, void __user *arg);

+extern int net_ns_check_bind(int addr_type, u32 addr);
+
 #else /* CONFIG_NET_NS */

 #define INIT_NET_NS(net_ns)
@@ -148,6 +150,11 @@
  return -ENOSYS;
 }

+static inline int net_ns_check_bind(int addr_type, u32 addr)
+{
+ return 0;
+}
+
 #endif /* !CONFIG_NET_NS */

 #endif /* _LINUX_NET_NAMESPACE_H */
Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
+++ 2.6.20-rc4-mm1/net/core/net_namespace.c
@@ -263,4 +263,58 @@
  return err;
 }

+/*
+ * This function check if the specified bind address is allowed.
+ * The bind is allowed if the address is:
```

```
+ * - 127.0.0.1
+ * - INADDR_ANY
+ * - INADDR_BROADCAST
+ * - a multicast address
+ * - the specified address match an ifaddr owned by the current
+ *   network namespace. That implies the local address and the
+ *   computed address from the netmask
+ * @addr_type : an addr type
+ * @addr : the requested bind address
+ * Returns: -EPERM on failure, 0 on success
+ */
+int net_ns_check_bind(int addr_type, u32 addr)
+{
+ int ret = -EPERM;
+       struct net_device *dev;
+       struct in_device *in_dev;
+ struct net_namespace *net_ns = current_net_ns;
+
+ if (LOOPBACK(addr) ||
+     MULTICAST(addr) ||
+     INADDR_ANY == addr ||
+     INADDR_BROADCAST == addr)
+  return 0;
+
+       read_lock(&dev_base_lock);
+       rcu_read_lock();
+       for (dev = dev_base; dev; dev = dev->next) {
+               in_dev = __in_dev_get_rcu(dev);
+               if (!in_dev)
+                       continue;
+
+               for_ifa(in_dev) {
+                       if (ifa->ifa_net_ns != net_ns)
+    continue;
+   if (addr == ifa->ifa_local ||
+       addr == ifa->ifa_broadcast ||
+       addr == (ifa->ifa_local & ifa->ifa_mask) ||
+       addr == ((ifa->ifa_address & ifa->ifa_mask)|
+         ~ifa->ifa_mask)) {
+    ret = 0;
+    goto out;
+   }
+               } endfor_ifa(in_dev);
+       }
+out:
+       read_unlock(&dev_base_lock);
+       rcu_read_unlock();
+
```

```
+ return ret;
+}
+
 #endif /* CONFIG_NET_NS */

--
```

Subject: [patch 07/12] net namespace: set source addresse
Posted by Daniel Lezcano on Fri, 19 Jan 2007 15:47:21 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

When no source address is specified, search from the dev list the
ifaddr allowed to be used as source address.

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

```
---
 include/linux/net_namespace.h |   14 ++++++++
 net/core/net_namespace.c      |   68 ++++++++++++++++++++++++++++++++++++++++++
 net/ipv4/route.c              |   28 ++++++++++------
 3 files changed, 100 insertions(+), 10 deletions(-)

Index: 2.6.20-rc4-mm1/net/ipv4/route.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/ipv4/route.c
+++ 2.6.20-rc4-mm1/net/ipv4/route.c
@@ -2475,17 +2475,17 @@

   if (LOCAL_MCAST(oldflp->fl4_dst) || oldflp->fl4_dst == htonl(0xFFFFFFFF)) {
    if (!fl.fl4_src)
-    fl.fl4_src = inet_select_addr(dev_out, 0,
-              RT_SCOPE_LINK);
+    fl.fl4_src = SELECT_SRC_ADDR(dev_out, 0,
+              RT_SCOPE_LINK);
    goto make_route;
   }
   if (!fl.fl4_src) {
    if (MULTICAST(oldflp->fl4_dst))
-    fl.fl4_src = inet_select_addr(dev_out, 0,
-              fl.fl4_scope);
+    fl.fl4_src = SELECT_SRC_ADDR(dev_out, 0,
```

```
+            fl.fl4_scope);
    else if (!oldflp->fl4_dst)
-    fl.fl4_src = inet_select_addr(dev_out, 0,
-            RT_SCOPE_HOST);
+    fl.fl4_src = SELECT_SRC_ADDR(dev_out, 0,
+            RT_SCOPE_HOST);
   }
  }

@@ -2525,8 +2525,8 @@
     */

    if (fl.fl4_src == 0)
-    fl.fl4_src = inet_select_addr(dev_out, 0,
-            RT_SCOPE_LINK);
+    fl.fl4_src = SELECT_SRC_ADDR(dev_out, 0,
+            RT_SCOPE_LINK);
    res.type = RTN_UNICAST;
    goto make_route;
   }
@@ -2539,7 +2539,13 @@

  if (res.type == RTN_LOCAL) {
   if (!fl.fl4_src)
+#ifdef CONFIG_NET_NS
+   fl.fl4_src = net_ns_select_source_address(dev_out,
+           fl.fl4_dst,
+           RT_SCOPE_LINK);
+#else
    fl.fl4_src = fl.fl4_dst;
+#endif
   if (dev_out)
    dev_put(dev_out);
   dev_out = &loopback_dev;
@@ -2561,8 +2567,10 @@
   fib_select_default(&fl, &res);

  if (!fl.fl4_src)
-  fl.fl4_src = FIB_RES_PREFSRC(res);
-
+  fl.fl4_src = res.fi->fib_prefsrc ? :
+   SELECT_SRC_ADDR(FIB_RES_DEV(res),
+     FIB_RES_GW(res),
+     res.scope);
  if (dev_out)
   dev_put(dev_out);
  dev_out = FIB_RES_DEV(res);
Index: 2.6.20-rc4-mm1/include/linux/net_namespace.h
```

```
===================================================================
--- 2.6.20-rc4-mm1.orig/include/linux/net_namespace.h
+++ 2.6.20-rc4-mm1/include/linux/net_namespace.h
@@ -5,6 +5,7 @@
 #include <linux/kref.h>
 #include <linux/nsproxy.h>
 #include <linux/errno.h>
+#include <linux/types.h>

 struct net_namespace {
  struct kref  kref;
@@ -95,6 +96,11 @@

 extern int net_ns_check_bind(int addr_type, u32 addr);

+extern __be32 net_ns_select_source_address(const struct net_device *dev,
+        u32 dst, int scope);
+
+#define SELECT_SRC_ADDR net_ns_select_source_address
+
 #else /* CONFIG_NET_NS */

 #define INIT_NET_NS(net_ns)
@@ -155,6 +161,14 @@
  return 0;
 }

+static inline __be32 net_ns_select_source_address(struct net_device *dev,
+        u32 dst, int scope)
+{
+ return 0;
+}
+
+#define SELECT_SRC_ADDR inet_select_addr
+
 #endif /* !CONFIG_NET_NS */

 #endif /* _LINUX_NET_NAMESPACE_H */
Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
+++ 2.6.20-rc4-mm1/net/core/net_namespace.c
@@ -317,4 +317,72 @@
  return ret;
 }

+/*
+ * This function choose the source address from the network device,
```

```
+ * destination and the scope. The function will browse the ifaddr
+ * owned by network namespace and choose the most adapted for the
+ * dst address and dev.
+ * @dev : the network device where the traffic will go
+ * @dst : the destination address
+ * @scope : the scope of the dst address
+ * Returns: a source address
+ */
+__be32 net_ns_select_source_address(const struct net_device *dev,
+        u32 dst, int scope)
+{
+ __be32 addr = 0;
+ struct in_device *in_dev;
+ struct net_namespace *net_ns = current_net_ns;
+
+ if (LOOPBACK(dst))
+     return htonl(INADDR_LOOPBACK);
+
+ if (!dev)
+  goto no_dev;
+
+ rcu_read_lock();
+ in_dev = __in_dev_get_rcu(dev);
+ if (!in_dev)
+  goto no_in_dev;
+
+ for_ifa(in_dev) {
+  if (ifa->ifa_scope > scope)
+   continue;
+  if (ifa->ifa_net_ns != net_ns)
+   continue;
+  if (!dst || inet_ifa_match(dst, ifa)) {
+   addr = ifa->ifa_local;
+   break;
+  }
+  if (!addr)
+   addr = ifa->ifa_local;
+ } endfor_ifa(in_dev);
+no_in_dev:
+ rcu_read_unlock();
+
+ if (addr)
+  goto out;
+
+no_dev:
+ 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_scope != RT_SCOPE_LINK &&
+       ifa->ifa_scope <= scope &&
+       ifa->ifa_net_ns == net_ns) {
+    addr = ifa->ifa_local;
+    goto out_unlock_both;
+   }
+  } endfor_ifa(in_dev);
+ }
+out_unlock_both:
+ read_unlock(&dev_base_lock);
+ rcu_read_unlock();
+out:
+ return addr;
+}
 #endif /* CONFIG_NET_NS */

--
```

Subject: [patch 08/12] net namespace : find namespace by addr
Posted by Daniel Lezcano on Fri, 19 Jan 2007 15:47:22 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

Switch to the the l3 namespace using the destination address.

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

```
---
 include/linux/net_namespace.h |    7 +++++++
 net/core/net_namespace.c      |   35 +++++++++++++++++++++++++++++++++++
 net/ipv4/ip_input.c           |   16 ++++++++++++++-
 3 files changed, 57 insertions(+), 1 deletion(-)

Index: 2.6.20-rc4-mm1/net/ipv4/ip_input.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/ipv4/ip_input.c
+++ 2.6.20-rc4-mm1/net/ipv4/ip_input.c
@@ -374,6 +374,9 @@
```

```
 {
  struct iphdr *iph;
  u32 len;
+ int err;
+ struct net_namespace *net_ns = current_net_ns;
+ struct net_namespace *dst_net_ns = NULL;

  /* When the interface is in promisc. mode, drop all the crap
   * that it receives, do not try to analyse it.
@@ -393,6 +396,9 @@

  iph = skb->nh.iph;

+ dst_net_ns = net_ns_find_from_dest_addr(iph->daddr);
+ if (dst_net_ns && !net_ns_match(net_ns, dst_net_ns))
+  push_net_ns(dst_net_ns);
  /*
   * RFC1122: 3.1.2.2 MUST silently discard any IP frame that fails the checksum.
   *
@@ -431,10 +437,18 @@
  /* 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);

+ if (dst_net_ns && !net_ns_match(net_ns, dst_net_ns))
+  pop_net_ns(net_ns);
+
+ return err;
+
 inhdr_error:
+ if (dst_net_ns && !net_ns_match(net_ns, dst_net_ns))
+  pop_net_ns(net_ns);
+
  IP_INC_STATS_BH(IPSTATS_MIB_INHDRERRORS);
 drop:
         kfree_skb(skb);
Index: 2.6.20-rc4-mm1/include/linux/net_namespace.h
===================================================================
--- 2.6.20-rc4-mm1.orig/include/linux/net_namespace.h
+++ 2.6.20-rc4-mm1/include/linux/net_namespace.h
@@ -99,6 +99,8 @@
 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 */
@@ -167,6 +169,11 @@
  return 0;
 }

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

 #endif /* !CONFIG_NET_NS */
Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
+++ 2.6.20-rc4-mm1/net/core/net_namespace.c
@@ -385,4 +385,39 @@
 out:
  return addr;
 }
+
+/*
+ * This function finds the network namespace destination deduced from
+ * the destination address. The network namespace is retrieved from
+ * the ifaddr owned by a network namespace
+ * @daddr  : destination
+ * Returns : the network namespace destination or NULL if not found
+ */
+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 */

--
```

Subject: [patch 09/12] net namespace : make loopback address always visible
Posted by Daniel Lezcano on Fri, 19 Jan 2007 15:47:23 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

Add a specific condition when doing inet interface listing
in order to see always the loopback address.

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

```
---
 include/linux/net_namespace.h |    9 +++++++++
 net/core/net_namespace.c      |   22 ++++++++++++++++++++++
 net/ipv4/devinet.c            |   12 +++++++-------
 3 files changed, 36 insertions(+), 7 deletions(-)

Index: 2.6.20-rc4-mm1/net/ipv4/devinet.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/ipv4/devinet.c
+++ 2.6.20-rc4-mm1/net/ipv4/devinet.c
@@ -695,8 +695,7 @@
    for (ifap = &in_dev->ifa_list; (ifa = *ifap) != NULL;
         ifap = &ifa->ifa_next) {
     if (!strcmp(ifr.ifr_name, ifa->ifa_label) &&
-        net_ns_match(ifa->ifa_net_ns,
-       current_net_ns) &&
+        net_ns_ifa_is_visible(ifa) &&
```

```
         sin_orig.sin_addr.s_addr ==
        ifa->ifa_address) {
       break; /* found */
@@ -710,13 +709,12 @@
     for (ifap = &in_dev->ifa_list; (ifa = *ifap) != NULL;
          ifap = &ifa->ifa_next)
      if (!strcmp(ifr.ifr_name, ifa->ifa_label) &&
-                                net_ns_match(ifa->ifa_net_ns,
-                                             current_net_ns))
+                                net_ns_ifa_is_visible(ifa))
       break;
    }
   }

- if (ifa && !net_ns_match(ifa->ifa_net_ns, current_net_ns))
+ if (ifa && !net_ns_ifa_is_visible(ifa))
    goto done;

   ret = -EADDRNOTAVAIL;
@@ -868,7 +866,7 @@
    goto out;

   for (; ifa; ifa = ifa->ifa_next) {
-   if (!net_ns_match(ifa->ifa_net_ns, current_net_ns))
+   if (!net_ns_ifa_is_visible(ifa))
     continue;
    if (!buf) {
     done += sizeof(ifr);
@@ -1216,7 +1214,7 @@

    for (ifa = in_dev->ifa_list, ip_idx = 0; ifa;
         ifa = ifa->ifa_next, ip_idx++) {
-    if (!net_ns_match(ifa->ifa_net_ns, current_net_ns))
+    if (!net_ns_ifa_is_visible(ifa))
      continue;
     if (ip_idx < s_ip_idx)
      continue;
Index: 2.6.20-rc4-mm1/include/linux/net_namespace.h
===================================================================
--- 2.6.20-rc4-mm1.orig/include/linux/net_namespace.h
+++ 2.6.20-rc4-mm1/include/linux/net_namespace.h
@@ -7,6 +7,8 @@
 #include <linux/errno.h>
 #include <linux/types.h>

+struct in_ifaddr;
+
 struct net_namespace {
```

```
  struct kref  kref;
  struct net_device *dev_base_p, **dev_tail_p;
@@ -101,6 +103,8 @@

 extern struct net_namespace *net_ns_find_from_dest_addr(u32 daddr);

+extern int net_ns_ifa_is_visible(const struct in_ifaddr *ifa);
+
 #define SELECT_SRC_ADDR net_ns_select_source_address

 #else /* CONFIG_NET_NS */
@@ -174,6 +178,11 @@
  return NULL;
 }

+static inline int net_ns_ifa_is_visible(const struct in_ifaddr *ifa)
+{
+ return 1;
+}
+
 #define SELECT_SRC_ADDR inet_select_addr

 #endif /* !CONFIG_NET_NS */
Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
+++ 2.6.20-rc4-mm1/net/core/net_namespace.c
@@ -420,4 +420,26 @@

  return net_ns;
 }
+
+/*
+ * This function checks if the ifaddr is visible from the
+ * current network namespace. This is true if the ifaddr is
+ * the loopback address or if the ifaddr is owned by the network
+ * namespace.
+ * @ifa : the ifaddr
+ * Returns : 1 if visible, 0 otherwise
+ */
+int net_ns_ifa_is_visible(const struct in_ifaddr *ifa)
+{
+  struct net_namespace *net_ns = current_net_ns;
+
+  if (LOOPBACK(ifa->ifa_local))
+   return 1;
+
+  if (net_ns_match(ifa->ifa_net_ns, net_ns))
```

```
+   return 1;
+
+  return 0;
+}
+
 #endif /* CONFIG_NET_NS */

--
```

_____

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

Subject: [patch 10/12] net namespace : add the loopback isolation
Posted by Daniel Lezcano on Fri, 19 Jan 2007 15:47:24 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

When a packet is outgoing, the namespace source is stored
into the skbuff. Because it is the loopback address, the
source == destination, so when the packet is incoming, it
has already the namespace destination set into the packet.

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

```
---
 include/linux/net_namespace.h |   13 ++++++++++--
 include/linux/skbuff.h        |    5 ++++-
 net/core/net_namespace.c      |   32 ++++++++++++++++++++++++++++++++-----
 net/ipv4/ip_input.c           |    2 +-
 net/ipv4/ip_output.c          |    1 +
 5 files changed, 44 insertions(+), 9 deletions(-)

Index: 2.6.20-rc4-mm1/include/linux/skbuff.h
===================================================================
--- 2.6.20-rc4-mm1.orig/include/linux/skbuff.h
+++ 2.6.20-rc4-mm1/include/linux/skbuff.h
@@ -225,6 +225,7 @@
  * @dma_cookie: a cookie to one of several possible DMA operations
  *  done by skb DMA functions
  * @secmark: security marking
+ *      @net_ns: namespace destination
  */

 struct sk_buff {
@@ -309,7 +310,9 @@
```

```
 #ifdef CONFIG_NETWORK_SECMARK
  __u32   secmark;
 #endif
-
+#ifdef CONFIG_NET_NS
+  struct net_namespace    *net_ns;
+#endif
  __u32   mark;

  /* These elements must be at the end, see alloc_skb() for details.  */
Index: 2.6.20-rc4-mm1/net/ipv4/ip_input.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/ipv4/ip_input.c
+++ 2.6.20-rc4-mm1/net/ipv4/ip_input.c
@@ -396,7 +396,7 @@

  iph = skb->nh.iph;

- dst_net_ns = net_ns_find_from_dest_addr(iph->daddr);
+ dst_net_ns = net_ns_find_from_dest_addr(skb);
  if (dst_net_ns && !net_ns_match(net_ns, dst_net_ns))
   push_net_ns(dst_net_ns);
  /*
Index: 2.6.20-rc4-mm1/net/ipv4/ip_output.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/ipv4/ip_output.c
+++ 2.6.20-rc4-mm1/net/ipv4/ip_output.c
@@ -272,6 +272,7 @@

  IP_INC_STATS(IPSTATS_MIB_OUTREQUESTS);

+ net_ns_tag_sk_buff(skb);
  skb->dev = dev;
  skb->protocol = htons(ETH_P_IP);

Index: 2.6.20-rc4-mm1/include/linux/net_namespace.h
===================================================================
--- 2.6.20-rc4-mm1.orig/include/linux/net_namespace.h
+++ 2.6.20-rc4-mm1/include/linux/net_namespace.h
@@ -8,6 +8,7 @@
 #include <linux/types.h>

 struct in_ifaddr;
+struct sk_buff;

 struct net_namespace {
  struct kref  kref;
@@ -101,10 +102,13 @@
```

```
 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);
+extern struct net_namespace
+*net_ns_find_from_dest_addr(const struct sk_buff *skb);

 extern int net_ns_ifa_is_visible(const struct in_ifaddr *ifa);

+extern void net_ns_tag_sk_buff(struct sk_buff *skb);
+
 #define SELECT_SRC_ADDR net_ns_select_source_address

 #else /* CONFIG_NET_NS */
@@ -173,7 +177,8 @@
  return 0;
 }

-static inline struct net_namespace *net_ns_find_from_dest_addr(u32 daddr)
+static inline struct net_namespace
+*net_ns_find_from_dest_addr(const struct sk_buff *skb)
 {
  return NULL;
 }
@@ -183,6 +188,10 @@
  return 1;
 }

+static inline void net_ns_tag_sk_buff(struct sk_buff *skb)
+{
+ ;
+}
 #define SELECT_SRC_ADDR inet_select_addr

 #endif /* !CONFIG_NET_NS */
Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
+++ 2.6.20-rc4-mm1/net/core/net_namespace.c
@@ -13,6 +13,9 @@
 #include <linux/in.h>
 #include <linux/netdevice.h>
 #include <linux/inetdevice.h>
+#include <linux/skbuff.h>
+#include <linux/ip.h>
+
 #include <net/ip_fib.h>
```

```
 struct net_namespace init_net_ns = {
@@ -389,18 +392,25 @@
 /*
  * This function finds the network namespace destination deduced from
  * the destination address. The network namespace is retrieved from
- * the ifaddr owned by a network namespace
- * @daddr  : destination
+ * the ifaddr owned by a network namespace. If the packet is for the
+ * loopback address so we assume the destination address is already filled
+ * by the sender which is the same as the receiver.
+ * @skb : the packet to be delivered
  * Returns : the network namespace destination or NULL if not found
  */
-struct net_namespace *net_ns_find_from_dest_addr(u32 daddr)
+struct net_namespace *net_ns_find_from_dest_addr(const struct sk_buff *skb)
 {
  struct net_namespace *net_ns = NULL;
  struct net_device *dev;
  struct in_device *in_dev;
+   struct iphdr *iph;
+   __be32 daddr;
+
+   iph = skb->nh.iph;
+   daddr = iph->daddr;

- if (LOOPBACK(daddr))
-  return current_net_ns;
+    if (LOOPBACK(daddr))
+    return skb->net_ns;

  read_lock(&dev_base_lock);
  rcu_read_lock();
@@ -442,4 +452,16 @@
   return 0;
 }

+/*
+ * This function allows to isolation the loopback. Because we are
+ * using the 127.0.0.1, we assume the source network namespace is
+ * the same as the destination network namespace. So setting the
+ * source, we have directly the destination when receiving the packet
+ * @skb : the packet to be tagged with the network namespace
+ */
+void net_ns_tag_sk_buff(struct sk_buff *skb)
+{
+ struct net_namespace *net_ns = current_net_ns;
+ skb->net_ns = net_ns;
+}
```

```
#endif /* CONFIG_NET_NS */

--
```

_____

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

Subject: [patch 11/12] net namespace : debugfs - add net_ns debugfs
Posted by Daniel Lezcano on Fri, 19 Jan 2007 15:47:25 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

For debug purpose only, this is not intended to be included.
Add /sys/kernel/debug/net_ns.

Creation of network namespace:

echo <level> > /sys/kernel/debug/net_ns/start

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

```
---
 fs/debugfs/Makefile |    2
 fs/debugfs/net_ns.c |  335 ++++++++++++++++++++++++++++++++++++++++++++++++++++
 net/Kconfig         |    4
 3 files changed, 340 insertions(+), 1 deletion(-)

Index: 2.6.20-rc4-mm1/fs/debugfs/Makefile
===================================================================
--- 2.6.20-rc4-mm1.orig/fs/debugfs/Makefile
+++ 2.6.20-rc4-mm1/fs/debugfs/Makefile
@@ -1,4 +1,4 @@
 debugfs-objs := inode.o file.o

 obj-$(CONFIG_DEBUG_FS) += debugfs.o
-
+obj-$(CONFIG_NET_NS_DEBUG) += net_ns.o
Index: 2.6.20-rc4-mm1/fs/debugfs/net_ns.c
===================================================================
--- /dev/null
+++ 2.6.20-rc4-mm1/fs/debugfs/net_ns.c
@@ -0,0 +1,335 @@
+/*
+ *  net_ns.c - adds a net_ns/ directory to debug NET namespaces
+ *
```

```
+ *  Author: Daniel Lezcano <dlezcano@fr.ibm.com>
+ *
+ *     This program is free software; you can redistribute it and/or
+ *     modify it under the terms of the GNU General Public License as
+ *     published by the Free Software Foundation, version 2 of the
+ *     License.
+ */
+
+#include <linux/module.h>
+#include <linux/kernel.h>
+#include <linux/pagemap.h>
+#include <linux/debugfs.h>
+#include <linux/sched.h>
+#include <linux/netdevice.h>
+#include <linux/inetdevice.h>
+#include <linux/syscalls.h>
+#include <linux/net_namespace.h>
+#include <linux/rtnetlink.h>
+
+static struct dentry *net_ns_dentry;
+static struct dentry *net_ns_dentry_dev;
+static struct dentry *net_ns_dentry_start;
+static struct dentry *net_ns_dentry_info;
+
+static ssize_t net_ns_dev_read_file(struct file *file, char __user *user_buf,
+        size_t count, loff_t *ppos)
+{
+ return 0;
+}
+
+static ssize_t net_ns_dev_write_file(struct file *file,
+         const char __user *user_buf,
+         size_t count, loff_t *ppos)
+{
+ return 0;
+}
+
+static int net_ns_dev_open_file(struct inode *inode, struct file *file)
+{
+ return 0;
+}
+
+static int net_ns_start_open_file(struct inode *inode, struct file *file)
+{
+ return 0;
+}
+
+static ssize_t net_ns_start_read_file(struct file *file, char __user *user_buf,
```

```
+          size_t count, loff_t *ppos)
+{
+ return 0;
+}
+
+static ssize_t net_ns_start_write_file(struct file *file,
+           const char __user *user_buf,
+           size_t count, loff_t *ppos)
+{
+ int err;
+ size_t len;
+ const char __user *p;
+ char c;
+ unsigned long flags;
+ struct net_namespace *net, *new_net;
+  struct nsproxy *new_nsproxy = NULL, *old_nsproxy = NULL;
+
+ if (current_net_ns != &init_net_ns)
+  return -EBUSY;
+
+ len = 0;
+ p = user_buf;
+ while (len < count) {
+  if (get_user(c, p++))
+   return -EFAULT;
+  if (c == 0 || c == '\n')
+   break;
+  len++;
+ }
+
+ if (len > 1)
+  return -EINVAL;
+
+ if (copy_from_user(&c, user_buf, sizeof(c)))
+  return -EFAULT;
+
+ if (c != '2' && c != '3')
+  return -EINVAL;
+
+ flags = (c=='2'?CLONE_NEWNET2:CLONE_NEWNET3);
+  err = unshare_net_ns(flags, &new_net);
+ if (err)
+  return err;
+
+  old_nsproxy = current->nsproxy;
+  new_nsproxy = dup_namespaces(old_nsproxy);
+
+  if (!new_nsproxy) {
```

```
+   put_net_ns(new_net);
+   task_unlock(current);
+   return -ENOMEM;
+  }
+
+  task_lock(current);
+
+  if (new_nsproxy) {
+   current->nsproxy = new_nsproxy;
+   new_nsproxy = old_nsproxy;
+  }
+
+  net = current->nsproxy->net_ns;
+  current->nsproxy->net_ns = new_net;
+ pop_net_ns(new_net);
+  new_net = net;
+
+  task_unlock(current);
+
+  put_nsproxy(new_nsproxy);
+  put_net_ns(new_net);
+
+ return count;
+}
+
+static int net_ns_info_open_file(struct inode *inode, struct file *file)
+{
+ return 0;
+}
+
+static ssize_t net_ns_info_read_file(struct file *file, char __user *user_buf,
+         size_t count, loff_t *ppos)
+{
+ const unsigned int length = 256;
+ size_t len;
+ char buff[length];
+ char *level;
+ struct net_namespace *net_ns = current_net_ns;
+ struct nsproxy *ns = current->nsproxy;
+
+ if (*ppos < 0)
+  return -EINVAL;
+ if (*ppos >= count)
+  return 0;
+ if (!count)
+  return 0;
+
+ switch (net_ns->level) {
```

```
+ case NET_NS_LEVEL2:
+  level = "layer 2";
+  break;
+ case NET_NS_LEVEL3:
+  level = "layer 3";
+  break;
+ default:
+  level = "unknown";
+  break;
+ }
+
+  sprintf(buff,
+   "nsproxy: %p\nnsproxy refcnt: %d\nnet_ns: %p\nnet_ns refcnt: %d\nlevel: %s\n",
+   ns,
+   atomic_read(&ns->count),
+   net_ns,
+   atomic_read(&net_ns->kref.refcount),
+   level);
+
+ len = strlen(buff);
+ if (len > count)
+  len = count;
+
+ if (copy_to_user(user_buf, buff, len))
+  return -EINVAL;
+
+ *ppos += count;
+
+ return count;
+}
+
+static ssize_t net_ns_info_write_file(struct file *file,
+         const char __user *user_buf,
+         size_t count, loff_t *ppos)
+{
+ struct net_namespace *net_ns = current_net_ns;
+ struct net_device *dev;
+ struct in_device *in_dev;
+ struct in_ifaddr **ifap = NULL;
+ struct in_ifaddr *ifa = NULL;
+ char *colon;
+ int err;
+
+ char buff[1024];
+ char *eth, *addr, *s;
+ __be32 address = 0;
+ __be32 p;
+
```

```
+ if (!capable(CAP_NET_ADMIN))
+  return -EPERM;
+
+ if (net_ns->level != NET_NS_LEVEL3)
+  return -EPERM;
+
+ if (count > sizeof(buff))
+  return -EINVAL;
+
+ if (copy_from_user(buff, user_buf, count))
+  return -EFAULT;
+
+ buff[count] = '\0';
+
+       eth = buff;
+       s = strchr(eth, ' ');
+       if (!s)
+               return -EINVAL;
+       *s = 0;
+
+       addr = s + 1;
+       s = strchr(addr, '.');
+       if (!s)
+               return -EINVAL;
+       *s = 0;
+       p = simple_strtoul(addr, NULL, 0);
+       ((char *)&address)[3] = p;
+       addr = s + 1;
+
+       s = strchr(addr, '.');
+       if (!s)
+               return -EINVAL;
+       *s = 0;
+       p = simple_strtoul(addr, NULL, 0);
+       ((char *)&address)[2] = p;
+       addr = s + 1;
+
+       s = strchr(addr, '.');
+       if (!s)
+               return -EINVAL;
+       *s = 0;
+       p = simple_strtoul(addr, NULL, 0);
+       ((char *)&address)[1] = p;
+       addr = s + 1;
+
+       p = simple_strtoul(addr, NULL, 0);
+       ((char *)&address)[0]  = p;
+
```

```
+ colon = strchr(eth, ':');
+ if (colon)
+  *colon = 0;
+
+       address = htonl(address);
+
+ rtnl_lock();
+
+ err = -ENODEV;
+ dev = __dev_get_by_name(eth);
+ if (!dev)
+  goto out;
+
+ if (colon)
+  *colon = ':';
+
+ err = -EADDRNOTAVAIL;
+ in_dev = __in_dev_get_rtnl(dev);
+ if (!in_dev)
+  goto out;
+
+ for (ifap = &in_dev->ifa_list; (ifa = *ifap) != NULL;
+      ifap = &ifa->ifa_next)
+  if (!strcmp(eth, ifa->ifa_label) &&
+      address == ifa->ifa_local)
+   break;
+ if (!ifa)
+  goto out;
+
+ ifa->ifa_net_ns = net_ns;
+
+ err = count;
+out:
+ rtnl_unlock();
+ return err;
+}
+
+
+static struct file_operations net_ns_dev_fops = {
+       .read =         net_ns_dev_read_file,
+       .write =        net_ns_dev_write_file,
+       .open =         net_ns_dev_open_file,
+};
+
+static struct file_operations net_ns_start_fops = {
+       .read =         net_ns_start_read_file,
+       .write =        net_ns_start_write_file,
+       .open =         net_ns_start_open_file,
```

```
+};
+
+static struct file_operations net_ns_info_fops = {
+       .read =         net_ns_info_read_file,
+       .write =        net_ns_info_write_file,
+       .open =         net_ns_info_open_file,
+};
+
+static int __init net_ns_init(void)
+{
+ net_ns_dentry = debugfs_create_dir("net_ns", NULL);
+
+ net_ns_dentry_dev = debugfs_create_file("dev", 0444,
+       net_ns_dentry,
+       NULL,
+       &net_ns_dev_fops);
+
+ net_ns_dentry_start = debugfs_create_file("start", 0666,
+        net_ns_dentry,
+        NULL,
+        &net_ns_start_fops);
+
+ net_ns_dentry_info = debugfs_create_file("info", 0444,
+       net_ns_dentry,
+       NULL,
+       &net_ns_info_fops);
+
+ return 0;
+}
+
+static void __exit net_ns_exit(void)
+{
+ debugfs_remove(net_ns_dentry_info);
+ debugfs_remove(net_ns_dentry_start);
+ debugfs_remove(net_ns_dentry_dev);
+ debugfs_remove(net_ns_dentry);
+}
+
+module_init(net_ns_init);
+module_exit(net_ns_exit);
+
+MODULE_DESCRIPTION("NET namespace debugfs");
+MODULE_AUTHOR("Daniel Lezcano <dlezcano@fr.ibm.com>");
+MODULE_LICENSE("GPL");
Index: 2.6.20-rc4-mm1/net/Kconfig
===================================================================
--- 2.6.20-rc4-mm1.orig/net/Kconfig
+++ 2.6.20-rc4-mm1/net/Kconfig
```

```
@@ -60,6 +60,10 @@

    Short answer: say Y.

+config NET_NS_DEBUG
+ bool "Debug fs for network namespace"
+ depends on DEBUG_FS && NET_NS
+
 if INET
 source "net/ipv4/Kconfig"
 source "net/ipv6/Kconfig"

--
```

_____

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

Subject: [patch 12/12] net namespace : Add broadcasting
Posted by Daniel Lezcano on Fri, 19 Jan 2007 15:47:26 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

Broadcast packets should be delivered to l2 and all l3 childs

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

```
---
 include/linux/net_namespace.h |   11 ++++++++++
 net/core/net_namespace.c      |   27 ++++++++++++++++++++++++++
 net/ipv4/udp.c                |    3 ++-
 3 files changed, 40 insertions(+), 1 deletion(-)

Index: 2.6.20-rc4-mm1/include/linux/net_namespace.h
===================================================================
--- 2.6.20-rc4-mm1.orig/include/linux/net_namespace.h
+++ 2.6.20-rc4-mm1/include/linux/net_namespace.h
@@ -9,6 +9,7 @@

 struct in_ifaddr;
 struct sk_buff;
+struct sock;

 struct net_namespace {
   struct kref  kref;
@@ -109,6 +110,9 @@
```

```
 extern void net_ns_tag_sk_buff(struct sk_buff *skb);

+extern int net_ns_sock_is_visible(const struct sock *sk,
+      const struct net_namespace *net_ns);
+
 #define SELECT_SRC_ADDR net_ns_select_source_address

 #else /* CONFIG_NET_NS */
@@ -192,6 +196,13 @@
 {
  ;
 }
+
+static inline int net_ns_sock_is_visible(const struct sock *sk,
+       const struct net_namespace *net_ns)
+{
+  return 1;
+}
+
 #define SELECT_SRC_ADDR inet_select_addr

 #endif /* !CONFIG_NET_NS */
Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
+++ 2.6.20-rc4-mm1/net/core/net_namespace.c
@@ -17,6 +17,7 @@
 #include <linux/ip.h>

 #include <net/ip_fib.h>
+#include <net/sock.h>

 struct net_namespace init_net_ns = {
  .kref = {
@@ -464,4 +465,30 @@
  struct net_namespace *net_ns = current_net_ns;
  skb->net_ns = net_ns;
 }
+
+/*
+ * This function checks if the socket is visible from the specified
+ * namespace. This is needed to ensure the broadcast and the multicast
+ * for multiple network namespace l2 and l3 to have the packets to be
+ * delivered. If we have a l3 namespace and its parent (l2 namespace)
+ * listening on a broadcast address, we should deliver the packet to
+ * both. That is done by the udp_v4_mcast_next function. But we should
+ * find a common point between sockets which are relatives to a
```

```
+ * namespace.  The common point is they have the same parent in case
+ * of l3 network namespace.
+ * @sk : the socket to be checked
+ * @net_ns : the receiving network namespace
+ * Returns: 1 if the socket is visible by the namespace, 0 otherwise.
+ */
+int net_ns_sock_is_visible(const struct sock *sk,
+      const struct net_namespace *net_ns)
+{
+ if (net_ns->level == NET_NS_LEVEL3)
+  net_ns = net_ns->parent;
+
+ if (sk->sk_net_ns->level == NET_NS_LEVEL3)
+  return sk->sk_net_ns->parent == net_ns;
+ else
+  return sk->sk_net_ns == net_ns;
+}
 #endif /* CONFIG_NET_NS */
Index: 2.6.20-rc4-mm1/net/ipv4/udp.c
===================================================================
--- 2.6.20-rc4-mm1.orig/net/ipv4/udp.c
+++ 2.6.20-rc4-mm1/net/ipv4/udp.c
@@ -309,9 +309,10 @@
       (inet->dport != rmt_port && inet->dport)  ||
       (inet->rcv_saddr && inet->rcv_saddr != loc_addr) ||
       ipv6_only_sock(s)     ||
-      !net_ns_match(sk->sk_net_ns, ns)   ||
       (s->sk_bound_dev_if && s->sk_bound_dev_if != dif))
    continue;
+  if (!net_ns_sock_is_visible(sk, ns))
+   continue;
   if (!ip_mc_sf_allow(s, loc_addr, rmt_addr, dif))
    continue;
   goto found;

--
```

Subject: Re: [patch 00/12] net namespace : L3 namespace - introduction
Posted by Herbert Poetzl on Sat, 20 Jan 2007 04:48:12 GMT

View Forum Message <> Reply to Message

On Fri, Jan 19, 2007 at 04:47:14PM +0100, dlezcano@fr.ibm.com wrote:
> This patchset provide a network isolation similar at what

> Linux-Vserver provides. It is based on the L2 namespaces and relies on
> the mechanisms provided by the namespace. This L3 namespaces does not
> aim to bring full virtualization for the network, it provides an IP
> isolation which can be reused for Linux-Vserver, jailed application or
> application containers.
>
> A L3 namespace are always L2 s' childs and they can not create more
> network namespaces, furthermore, they lose their NET_ADMIN
> capability. They share their parent's network ressources. From the
> parent namespace, IP addresses are created and assigned to the
> different L3 childs. From this point, L3 namespaces can use their
> assigned IP address and all computed broadcast addresses.
  ~~~~~~~~~~~~~~~~~~~

okay, I conclude that this only handles a single address
for now. what are your plans to handle entire sets?

TIA,
Herbert

> Because the L3 namespace relies on the L2 virtualization mechanisms,
> it is possible to have several L3 namespaces listening on
> INADDR_ANY:port without conflict, that's allow to run several server
> without modifying the network configuration.
>
> The loopback is a shared device between all L3 namespaces. To ensure
> the 127.0.0.1 address isolation, the sender store its namespace into
> the packet, so when the packet arrives, the destination namespace is
> already set, because "source" == "destination". By this way, it is
> easy to disable the loopback isolation and let the application to talk
> with application outside of the namespace via the 127.0.0.1 because we
> consider them trusted (like portmap).
>
> The ifconfig / ip commands will only show IP addresses assigned to the
> L3 namespace. When a L3 namespace dies, the assigned IP address is
> released to its parent.
>
> At the IP level, when a packet arrives, the L3 network namespace
> destination is retrieved from the destination address.
>
> At the bind time, the address is checked against the assigned IP
> address.
>
> --
> ____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers
  ~~~~~~~~~~~~~~~~~~~

Subject: Re: [patch 05/12] net namespace : ioctl to push ifa to net namespace l3
Posted by Herbert Poetzl on Sat, 20 Jan 2007 04:52:34 GMT
View Forum Message <> Reply to Message

On Fri, Jan 19, 2007 at 04:47:19PM +0100, dlezcano@fr.ibm.com wrote:
> From: Daniel Lezcano <dlezcano@fr.ibm.com>
>
> New ioctl to "push" ifaddr to a container. Actually, the push is done
> from the current namespace, so the right word is "pull". That will be
> changed to move ifaddr from l2 network namespace to l3.
>
> Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
>
> ---
> include/linux/net_namespace.h | 7 ++
> include/linux/sockios.h | 4 +
> net/core/net_namespace.c | 118 ++-
> net/ipv4/af_inet.c | 4 +
> 4 files changed, 132 insertions(+), 1 deletion(-)
>
> Index: 2.6.20-rc4-mm1/include/linux/sockios.h
> ===
> --- 2.6.20-rc4-mm1.orig/include/linux/sockios.h
> +++ 2.6.20-rc4-mm1/include/linux/sockios.h
> @@ -122,6 +122,10 @@
> #define SIOCBRADDIF 0x89a2 /* add interface to bridge */
> #define SIOCBRDELIF 0x89a3 /* remove interface from bridge */
>
> +/* Container calls */
> +#define SIOCNETNSPUSHIF 0x89b0 /* add ifaddr to namespace */
> +#define SIOCNETNSPULLIF 0x89b1 /* remove ifaddr to namespace */
~~~ from
> +
> /* Device private ioctl calls */
>
> /*
> Index: 2.6.20-rc4-mm1/net/ipv4/af_inet.c
> ===================================================================
> --- 2.6.20-rc4-mm1.orig/net/ipv4/af_inet.c
> +++ 2.6.20-rc4-mm1/net/ipv4/af_inet.c
> @@ -789,6 +789,10 @@
> case SIOCSIFFLAGS:
~~~

```
>     err = devinet_ioctl(cmd, (void __user *)arg);
>     break;
> +         case SIOCNETNSPUSHIF:
> +         case SIOCNETNSPULLIF:
> +   err = net_ns_ioctl(cmd, (void __user *)arg);
> +   break;
>    default:
>     if (sk->sk_prot->ioctl)
>      err = sk->sk_prot->ioctl(sk, cmd, arg);
> Index: 2.6.20-rc4-mm1/include/linux/net_namespace.h
> ===================================================================
> --- 2.6.20-rc4-mm1.orig/include/linux/net_namespace.h
> +++ 2.6.20-rc4-mm1/include/linux/net_namespace.h
> @@ -91,6 +91,8 @@
>
>  #define net_ns_hash(ns)   ((ns)->hash)
>
> +extern int net_ns_ioctl(unsigned int cmd, void __user *arg);
> +
>  #else /* CONFIG_NET_NS */
>
>  #define INIT_NET_NS(net_ns)
> @@ -141,6 +143,11 @@
>
>  #define net_ns_hash(ns)   (0)
>
> +static inline int net_ns_ioctl(unsigned int cmd, void __user *arg)
> +{
> + return -ENOSYS;
> +}
> +
>  #endif /* !CONFIG_NET_NS */
>
>  #endif /* _LINUX_NET_NAMESPACE_H */
> Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
> ===================================================================
> --- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
> +++ 2.6.20-rc4-mm1/net/core/net_namespace.c
> @@ -10,7 +10,9 @@
>  #include <linux/nsproxy.h>
>  #include <linux/net_namespace.h>
>  #include <linux/net.h>
> +#include <linux/in.h>
>  #include <linux/netdevice.h>
> +#include <linux/inetdevice.h>
>  #include <net/ip_fib.h>
>
>  struct net_namespace init_net_ns = {
```

```
> @@ -123,6 +125,33 @@
>   return err;
>  }
>
> +/*
> + * The function will move the ifaddr to the l2 network namespace
> + * parent.
> + * @net_ns: the related network namespace
> + */
> +static void release_ifa_to_parent(const struct net_namespace* net_ns)
> +{
> + struct net_device *dev;
> + struct in_device *in_dev;
> +
> + read_lock(&dev_base_lock);
> + rcu_read_lock();
> + for (dev = dev_base; dev; dev = dev->next) {
> +  in_dev = __in_dev_get_rcu(dev);
> +  if (!in_dev)
> +   continue;
> +
> +  for_ifa(in_dev) {
> +   if (ifa->ifa_net_ns != net_ns)
> +    continue;
> +   ifa->ifa_net_ns = net_ns->parent;
> +  } endfor_ifa(in_dev);
> + }
> + read_unlock(&dev_base_lock);
> + rcu_read_unlock();
> +}
> +
>  void free_net_ns(struct kref *kref)
>  {
>   struct net_namespace *ns;
> @@ -139,12 +168,99 @@
>    }
>   }
>
> - if (ns->level == NET_NS_LEVEL3)
> + if (ns->level == NET_NS_LEVEL3) {
> +  release_ifa_to_parent(ns);
>    put_net_ns(ns->parent);
> + }
>
>   printk(KERN_DEBUG "NET_NS: net namespace %p destroyed\n", ns);
>   kfree(ns);
>  }
>  EXPORT_SYMBOL_GPL(free_net_ns);
```

>
> +/*
> + * This function allows to assign an IP address from a l2 network
> + * namespace to one of his l3 child or to release from an l3 network
> + * namespace to his l2 network namespace parent.

hmm, sounds like the address is moved between the
namespaces? does that mean that the 'parent' will
not see the 'isolated' ip anymore?

TIA,
Herbert

> + * @cmd: a "push" / "pull" command
> + * @arg: an userspace buffer containing an ifreq structure
> + * Returns:
> + * - EPERM : if caller has no CAP_NET_ADMIN capabilities or the
> + *   current level of network namespace is not layer 2
> + * - EFAULT : if arg is an invalid buffer
> + * - EADDRNOTAVAIL : if the specified ifaddr does not exists
> + * - EINVAL : if cmd is unknown
> + * - zero on success
> + */
> +int net_ns_ioctl(unsigned int cmd, void __user *arg)
> +{
> + struct ifreq ifr;
> + struct sockaddr_in *sin = (struct sockaddr_in *)&ifr.ifr_addr;
> + struct net_namespace *net_ns = current_net_ns;
> + struct net_device *dev;
> + struct in_device *in_dev;
> + struct in_ifaddr **ifap = NULL;
> + struct in_ifaddr *ifa = NULL;
> + char *colon;
> + int err;
> +
> + if (!capable(CAP_NET_ADMIN))
> + return -EPERM;
> +
> + if (net_ns->level != NET_NS_LEVEL3)
> + return -EPERM;
> +
> + if (copy_from_user(&ifr, arg, sizeof(struct ifreq)))
> + return -EFAULT;
> +
> + ifr.ifr_name[IFNAMSIZ - 1] = 0;
> +
> +
> + colon = strchr(ifr.ifr_name, ':');

```
> + if (colon)
> +  *colon = 0;
> +
> + rtnl_lock();
> +
> + err = -ENODEV;
> + dev = __dev_get_by_name(ifr.ifr_name);
> + if (!dev)
> +  goto out;
> +
> + if (colon)
> +  *colon = ':';
> +
> + err = -EADDRNOTAVAIL;
> + in_dev = __in_dev_get_rtnl(dev);
> + if (!in_dev)
> +  goto out;
> +
> + for (ifap = &in_dev->ifa_list; (ifa = *ifap) != NULL;
> +      ifap = &ifa->ifa_next)
> +  if (!strcmp(ifr.ifr_name, ifa->ifa_label) &&
> +      sin->sin_addr.s_addr == ifa->ifa_local)
> +   break;
> + if (!ifa)
> +  goto out;
> +
> + err = -EINVAL;
> + switch(cmd) {
> +
> + case SIOCNETNSPUSHIF:
> +  ifa->ifa_net_ns = net_ns;
> +  break;
> +
> + case SIOCNETNSPULLIF:
> +  ifa->ifa_net_ns = net_ns->parent;
> +  break;
> + default:
> +  goto out;
> + }
> +
> + err = 0;
> +out:
> + rtnl_unlock();
> + return err;
> +}
> +
>  #endif /* CONFIG_NET_NS */
>
```

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

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

Subject: Re: [patch 08/12] net namespace : find namespace by addr
Posted by Herbert Poetzl on Sat, 20 Jan 2007 04:56:13 GMT
View Forum Message <> Reply to Message

On Fri, Jan 19, 2007 at 04:47:22PM +0100, dlezcano@fr.ibm.com wrote:
> From: Daniel Lezcano <dlezcano@fr.ibm.com>
>
> Switch to the the l3 namespace using the destination address.
>
> Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
>
> ---
> include/linux/net_namespace.h | 7 +++++++
> net/core/net_namespace.c | 35 +++++++++++++++++++++++++++++++++++
> net/ipv4/ip_input.c | 16 ++++++++++++++-
> 3 files changed, 57 insertions(+), 1 deletion(-)
>
> Index: 2.6.20-rc4-mm1/net/ipv4/ip_input.c
> ===
> --- 2.6.20-rc4-mm1.orig/net/ipv4/ip_input.c
> +++ 2.6.20-rc4-mm1/net/ipv4/ip_input.c
> @@ -374,6 +374,9 @@
> {
>   struct iphdr *iph;
>   u32 len;
> + int err;
> + struct net_namespace *net_ns = current_net_ns;
> + struct net_namespace *dst_net_ns = NULL;
>
>   /* When the interface is in promisc. mode, drop all the crap
>   * that it receives, do not try to analyse it.
> @@ -393,6 +396,9 @@
>
>   iph = skb->nh.iph;
>
> + dst_net_ns = net_ns_find_from_dest_addr(iph->daddr);
> + if (dst_net_ns && !net_ns_match(net_ns, dst_net_ns))

> + push_net_ns(dst_net_ns);
>   /*
>   * RFC1122: 3.1.2.2 MUST silently discard any IP frame that fails the checksum.
>   *
> @@ -431,10 +437,18 @@
>   /* 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);
>
> + if (dst_net_ns && !net_ns_match(net_ns, dst_net_ns))
> + pop_net_ns(net_ns);
> +
> + return err;
> +
> inhdr_error:
> + if (dst_net_ns && !net_ns_match(net_ns, dst_net_ns))
> + pop_net_ns(net_ns);
> +
>   IP_INC_STATS_BH(IPSTATS_MIB_INHDRERRORS);
> drop:
>   kfree_skb(skb);
> Index: 2.6.20-rc4-mm1/include/linux/net_namespace.h
> ===
> --- 2.6.20-rc4-mm1.orig/include/linux/net_namespace.h
> +++ 2.6.20-rc4-mm1/include/linux/net_namespace.h
> @@ -99,6 +99,8 @@
> 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 */
> @@ -167,6 +169,11 @@
>   return 0;
> }
>
> +static inline struct net_namespace *net_ns_find_from_dest_addr(u32 daddr)
> +{
> + return NULL;
> +}
> +
> #define SELECT_SRC_ADDR inet_select_addr
>

> #endif /* !CONFIG_NET_NS */
> Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
> ===
> --- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
> +++ 2.6.20-rc4-mm1/net/core/net_namespace.c
> @@ -385,4 +385,39 @@
> out:
>   return addr;
> }
> +
> +/*
> + * This function finds the network namespace destination deduced from
> + * the destination address. The network namespace is retrieved from
> + * the ifaddr owned by a network namespace

this basically disallows to 'share' IPs between
namespaces, as it is permitted in Linux-VServer
right now, or am I misinterpreting this?

TIA,
Herbert

> + * @daddr : destination
> + * Returns : the network namespace destination or NULL if not found
> + */
> +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 */
>
> --
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

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

Subject: Re: [patch 12/12] net namespace : Add broadcasting
Posted by Herbert Poetzl on Sat, 20 Jan 2007 04:58:37 GMT
View Forum Message <> Reply to Message

On Fri, Jan 19, 2007 at 04:47:26PM +0100, dlezcano@fr.ibm.com wrote:
> From: Daniel Lezcano <dlezcano@fr.ibm.com>
>
> Broadcast packets should be delivered to l2 and all l3 childs

hmm, really? shouldn't it only reach those which
actually have related addresses assigned?

best,
Herbert

> Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
>
> ---
> include/linux/net_namespace.h | 11 ++++++++++
> net/core/net_namespace.c | 27 ++++++++++++++++++++++++++++
> net/ipv4/udp.c | 3 ++-
> 3 files changed, 40 insertions(+), 1 deletion(-)
>
> Index: 2.6.20-rc4-mm1/include/linux/net_namespace.h
> ===
> --- 2.6.20-rc4-mm1.orig/include/linux/net_namespace.h
> +++ 2.6.20-rc4-mm1/include/linux/net_namespace.h
> @@ -9,6 +9,7 @@
>
> struct in_ifaddr;
> struct sk_buff;

> +struct sock;
>
> struct net_namespace {
>   struct kref kref;
> @@ -109,6 +110,9 @@
>
> extern void net_ns_tag_sk_buff(struct sk_buff *skb);
>
> +extern int net_ns_sock_is_visible(const struct sock *sk,
> +   const struct net_namespace *net_ns);
> +
> #define SELECT_SRC_ADDR net_ns_select_source_address
>
> #else /* CONFIG_NET_NS */
> @@ -192,6 +196,13 @@
> {
>   ;
> }
> +
> +static inline int net_ns_sock_is_visible(const struct sock *sk,
> +   const struct net_namespace *net_ns)
> +{
> + return 1;
> +}
> +
> #define SELECT_SRC_ADDR inet_select_addr
>
> #endif /* !CONFIG_NET_NS */
> Index: 2.6.20-rc4-mm1/net/core/net_namespace.c
> ===
> --- 2.6.20-rc4-mm1.orig/net/core/net_namespace.c
> +++ 2.6.20-rc4-mm1/net/core/net_namespace.c
> @@ -17,6 +17,7 @@
> #include <linux/ip.h>
>
> #include <net/ip_fib.h>
> +#include <net/sock.h>
>
> struct net_namespace init_net_ns = {
>   .kref = {
> @@ -464,4 +465,30 @@
>   struct net_namespace *net_ns = current_net_ns;
>   skb->net_ns = net_ns;
> }
> +
> +/*
> + * This function checks if the socket is visible from the specified
> + * namespace. This is needed to ensure the broadcast and the multicast

> + * for multiple network namespace l2 and l3 to have the packets to be
> + * delivered. If we have a l3 namespace and its parent (l2 namespace)
> + * listening on a broadcast address, we should deliver the packet to
> + * both. That is done by the udp_v4_mcast_next function. But we should
> + * find a common point between sockets which are relatives to a
> + * namespace. The common point is they have the same parent in case
> + * of l3 network namespace.
> + * @sk : the socket to be checked
> + * @net_ns : the receiving network namespace
> + * Returns: 1 if the socket is visible by the namespace, 0 otherwise.
> + */
> +int net_ns_sock_is_visible(const struct sock *sk,
> +   const struct net_namespace *net_ns)
> +{
> + if (net_ns->level == NET_NS_LEVEL3)
> + net_ns = net_ns->parent;
> +
> + if (sk->sk_net_ns->level == NET_NS_LEVEL3)
> + return sk->sk_net_ns->parent == net_ns;
> + else
> + return sk->sk_net_ns == net_ns;
> +}
> #endif /* CONFIG_NET_NS */
> Index: 2.6.20-rc4-mm1/net/ipv4/udp.c
> ===
> --- 2.6.20-rc4-mm1.orig/net/ipv4/udp.c
> +++ 2.6.20-rc4-mm1/net/ipv4/udp.c
> @@ -309,9 +309,10 @@
>   (inet->dport != rmt_port && inet->dport) ||
>   (inet->rcv_saddr && inet->rcv_saddr != loc_addr) ||
>   ipv6_only_sock(s) ||
> -   !net_ns_match(sk->sk_net_ns, ns) ||
>   (s->sk_bound_dev_if && s->sk_bound_dev_if != dif))
>   continue;
> + if (!net_ns_sock_is_visible(sk, ns))
> +   continue;
>   if (!ip_mc_sf_allow(s, loc_addr, rmt_addr, dif))
>   continue;
>   goto found;
>
> --
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.osdl.org

Subject: Re: [patch 00/12] net namespace : L3 namespace - introduction
Posted by Daniel Lezcano on Sat, 20 Jan 2007 11:42:45 GMT
View Forum Message <> Reply to Message

Herbert Poetzl wrote:
> On Fri, Jan 19, 2007 at 04:47:14PM +0100, dlezcano@fr.ibm.com wrote:
>> This patchset provide a network isolation similar at what
>> Linux-Vserver provides. It is based on the L2 namespaces and relies on
>> the mechanisms provided by the namespace. This L3 namespaces does not
>> aim to bring full virtualization for the network, it provides an IP
>> isolation which can be reused for Linux-Vserver, jailed application or
>> application containers.
>>
>> A L3 namespace are always L2 s' childs and they can not create more
>> network namespaces, furthermore, they lose their NET_ADMIN
>> capability. They share their parent's network ressources. From the
>> parent namespace, IP addresses are created and assigned to the
>> different L3 childs. From this point, L3 namespaces can use their
>> assigned IP address and all computed broadcast addresses.
> ~~~~~~~~~~~~~~~~~~~
>
> okay, I conclude that this only handles a single address
> for now. what are your plans to handle entire sets?
>
You can assign more than one IP address to a L3 network namespace.

Subject: Re: [patch 05/12] net namespace : ioctl to push ifa to net namespace l3
Posted by Daniel Lezcano on Sat, 20 Jan 2007 11:48:27 GMT
View Forum Message <> Reply to Message

Herbert Poetzl wrote:
> On Fri, Jan 19, 2007 at 04:47:19PM +0100, dlezcano@fr.ibm.com wrote:
>> From: Daniel Lezcano <dlezcano@fr.ibm.com>
>>
>> New ioctl to "push" ifaddr to a container. Actually, the push is done
>> from the current namespace, so the right word is "pull". That will be
>> changed to move ifaddr from l2 network namespace to l3.
>>

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

[...]

>> + * namespace to his l2 network namespace parent.
>
> hmm, sounds like the address is moved between the
> namespaces? does that mean that the 'parent' will
> not see the 'isolated' ip anymore?

Yes, exact. If the IP address is visible into several namespaces (L2
parent and L3 child), nothing ensures the two will not use the IP address.

Subject: Re: [patch 12/12] net namespace : Add broadcasting
Posted by Daniel Lezcano on Sat, 20 Jan 2007 11:54:33 GMT
View Forum Message <> Reply to Message

Herbert Poetzl wrote:
> On Fri, Jan 19, 2007 at 04:47:26PM +0100, dlezcano@fr.ibm.com wrote:
>> From: Daniel Lezcano <dlezcano@fr.ibm.com>
>>
>> Broadcast packets should be delivered to l2 and all l3 childs
>
> hmm, really? shouldn't it only reach those which
> actually have related addresses assigned?

Yes right but this is what happens, My sentence is incomplete.

"broadcast packets should be delivered to l2 and all l3 childs having
sockets listening for this broadcast address"

Thanks.
