Subject: [PATCH 13/15] Revert "netns: Fix device renaming for sysfs"
Posted by ebiederm on Fri, 04 Jul 2008 01:21:28 GMT
View Forum Message <> Reply to Message

This reverts commit aaf8cdc34ddba08122f02217d9d684e2f9f5d575.

Drivers like the ipw2100 call device_create_group when they
are initialized and device_remove_group when they are shutdown.
Moving them between namespaces deletes their sysfs groups early.

In particular the following call chain results.
netdev_unregister_kobject -> device_del -> kobject_del -> sysfs_remove_dir
With sysfs_remove_dir recursively deleting all of it's subdirectories,
and nothing adding them back.

Ouch!

Therefore we need to call something that ultimate calls sysfs_mv_dir
as that sysfs function can move sysfs directories between namespaces
without deleting their subdirectories or their contents. Allowing
us to avoid placing extra boiler plate into every driver that does
something interesting with sysfs.

Currently the function that provides that capability is device_rename.
That is the code works without nasty side effects as originally written.

So remove the misguided fix for moving devices between namespaces. The
bug in the kobject layer that inspired it has now been recognized and
fixed.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 net/core/dev.c | 4 +---
 net/core/net-sysfs.c | 7 +------
 net/core/net-sysfs.h | 2 +-
 3 files changed, 3 insertions(+), 10 deletions(-)

diff --git a/net/core/dev.c b/net/core/dev.c
index fca23a3..585584d 100644
--- a/net/core/dev.c
+++ b/net/core/dev.c
@@ -3806,7 +3806,6 @@ int register_netdevice(struct net_device *dev)
   }
  }

- netdev_initialize_kobject(dev);
  ret = netdev_register_kobject(dev);
  if (ret)

```
   goto err_uninit;
@@ -4239,8 +4238,7 @@ int dev_change_net_namespace(struct net_device *dev, struct net
*net, const char
  }

  /* Fixup kobjects */
- netdev_unregister_kobject(dev);
- err = netdev_register_kobject(dev);
+ err = device_rename(&dev->dev, dev->name);
  WARN_ON(err);

  /* Add the device back in the hashes */
diff --git a/net/core/net-sysfs.c b/net/core/net-sysfs.c
index 90e2177..4e7b847 100644
--- a/net/core/net-sysfs.c
+++ b/net/core/net-sysfs.c
@@ -449,6 +449,7 @@ int netdev_register_kobject(struct net_device *net)
  struct device *dev = &(net->dev);
  struct attribute_group **groups = net->sysfs_groups;

+ device_initialize(dev);
  dev->class = &net_class;
  dev->platform_data = net;
  dev->groups = groups;
@@ -469,12 +470,6 @@ int netdev_register_kobject(struct net_device *net)
  return device_add(dev);
 }

-void netdev_initialize_kobject(struct net_device *net)
-{
- struct device *device = &(net->dev);
- device_initialize(device);
-}
-
 int netdev_kobject_init(void)
 {
  return class_register(&net_class);
diff --git a/net/core/net-sysfs.h b/net/core/net-sysfs.h
index 14e7524..f5f108d 100644
--- a/net/core/net-sysfs.h
+++ b/net/core/net-sysfs.h
@@ -4,5 +4,5 @@
 int netdev_kobject_init(void);
 int netdev_register_kobject(struct net_device *);
 void netdev_unregister_kobject(struct net_device *);
-void netdev_initialize_kobject(struct net_device *);
+
 #endif
```

--
1.5.3.rc6.17.g1911

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

---

Subject: [PATCH 14/15] netns: Enable tagging for net_class directories in sysfs
Posted by ebiederm on Fri, 04 Jul 2008 01:22:35 GMT
View Forum Message <> Reply to Message

The problem. Network devices show up in sysfs and with the network
namespace active multiple devices with the same name can show up in
the same directory, ouch!

To avoid that problem and allow existing applications in network namespaces
to see the same interface that is currently presented in sysfs, this
patch enables the tagging directory support in sysfs.

By using the network namespace pointers as tags to separate out the
the sysfs directory entries we ensure that we don't have conflicts
in the directories and applications only see a limited set of
the network devices.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 include/linux/sysfs.h | 1 +
 net/Kconfig | 2 +-
 net/core/net-sysfs.c | 33 +++++++++++++++++++++++++++++++++
 3 files changed, 35 insertions(+), 1 deletions(-)

diff --git a/include/linux/sysfs.h b/include/linux/sysfs.h
index c3a30ce..1ed31bb 100644
--- a/include/linux/sysfs.h
+++ b/include/linux/sysfs.h
@@ -80,6 +80,7 @@ struct sysfs_ops {

 enum sysfs_tag_type {
  SYSFS_TAG_TYPE_NONE = 0,
+ SYSFS_TAG_TYPE_NETNS,
  SYSFS_TAG_TYPES
 };

diff --git a/net/Kconfig b/net/Kconfig
index acbf7c6..9aad03b 100644
--- a/net/Kconfig

```
+++ b/net/Kconfig
@@ -30,7 +30,7 @@ menu "Networking options"
 config NET_NS
  bool "Network namespace support"
  default n
- depends on EXPERIMENTAL && !SYSFS && NAMESPACES
+ depends on EXPERIMENTAL && NAMESPACES
  help
    Allow user space to create what appear to be multiple instances
    of the network stack.
diff --git a/net/core/net-sysfs.c b/net/core/net-sysfs.c
index 4e7b847..6227a28 100644
--- a/net/core/net-sysfs.c
+++ b/net/core/net-sysfs.c
@@ -13,7 +13,9 @@
 #include <linux/kernel.h>
 #include <linux/netdevice.h>
 #include <linux/if_arp.h>
+#include <linux/nsproxy.h>
 #include <net/sock.h>
+#include <net/net_namespace.h>
 #include <linux/rtnetlink.h>
 #include <linux/wireless.h>
 #include <net/iw_handler.h>
@@ -385,6 +387,24 @@ static struct attribute_group wireless_group = {
 };
 #endif

+static const void *net_sysfs_mount_tag(void)
+{
+ return current->nsproxy->net_ns;
+}
+
+static struct sysfs_tag_type_operations net_tag_type_operations = {
+ .mount_tag = net_sysfs_mount_tag,
+};
+
+static void net_sysfs_net_exit(struct net *net)
+{
+ sysfs_exit_tag(SYSFS_TAG_TYPE_NETNS, net);
+}
+
+static struct pernet_operations sysfs_net_ops = {
+ .exit = net_sysfs_net_exit,
+};
+
 #endif /* CONFIG_SYSFS */
```

```
 #ifdef CONFIG_HOTPLUG
@@ -421,6 +441,13 @@ static void netdev_release(struct device *d)
  kfree((char *)dev - dev->padded);
 }

+static const void *net_sysfs_tag(struct device *d)
+{
+ struct net_device *dev;
+ dev = container_of(d, struct net_device, dev);
+ return dev_net(dev);
+}
+
 static struct class net_class = {
  .name = "net",
  .dev_release = netdev_release,
@@ -430,6 +457,8 @@ static struct class net_class = {
 #ifdef CONFIG_HOTPLUG
  .dev_uevent = netdev_uevent,
 #endif
+ .tag_type = SYSFS_TAG_TYPE_NETNS,
+ .sysfs_tag = net_sysfs_tag,
 };

 /* Delete sysfs entries but hold kobject reference until after all
@@ -472,5 +501,9 @@ int netdev_register_kobject(struct net_device *net)

 int netdev_kobject_init(void)
 {
+#ifdef CONFIG_SYSFS
+ sysfs_register_tag_type(SYSFS_TAG_TYPE_NETNS, &net_tag_type_operations);
+ register_pernet_subsys(&sysfs_net_ops);
+#endif
  return class_register(&net_class);
 }
--
1.5.3.rc6.17.g1911
```

Subject: [PATCH 15/15] sysfs: user namespaces: fix bug with clone(CLONE_NEWUSER) with fairsched
Posted by ebiederm on Fri, 04 Jul 2008 01:23:35 GMT
View Forum Message <> Reply to Message

Mark the /sys/kernel/uids directory to be tagged so that processes in
different user namespaces can remount /sys and see their own uid
listings.

Without this patch, having CONFIG_FAIR_SCHED=y makes user namespaces
unusable, because when you
  clone(CLONE_NEWUSER)
it will auto-create the root userid and try to create
/sys/kernel/uids/0. Since that already exists from the parent user
namespace, the create fails, and the clone misleadingly ends up
returning -ENOMEM.

This patch fixes the issue by allowing each user namespace to remount
/sys, and having /sys filter the /sys/kernel/uid/ entries by user
namespace.

Changelong:
v2 - Reworked for the updated sysfs api

Signed-off-by: Serge Hallyn <serue@us.ibm.com>
Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 include/linux/sched.h | 1 +
 include/linux/sysfs.h | 1 +
 kernel/user.c | 22 ++++++++++++++++++++++
 kernel/user_namespace.c | 1 +
 4 files changed, 25 insertions(+), 0 deletions(-)

diff --git a/include/linux/sched.h b/include/linux/sched.h
index c5d3f84..d2be6a5 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -598,6 +598,7 @@ struct user_struct {
  /* Hash table maintenance information */
  struct hlist_node uidhash_node;
  uid_t uid;
+ struct user_namespace *user_ns;

 #ifdef CONFIG_USER_SCHED
  struct task_group *tg;
diff --git a/include/linux/sysfs.h b/include/linux/sysfs.h
index 1ed31bb..ecb942c 100644
--- a/include/linux/sysfs.h
+++ b/include/linux/sysfs.h
@@ -81,6 +81,7 @@ struct sysfs_ops {
 enum sysfs_tag_type {

```
  SYSFS_TAG_TYPE_NONE = 0,
  SYSFS_TAG_TYPE_NETNS,
+ SYSFS_TAG_TYPE_USERNS,
  SYSFS_TAG_TYPES
 };

diff --git a/kernel/user.c b/kernel/user.c
index 865ecf5..ca29fbc 100644
--- a/kernel/user.c
+++ b/kernel/user.c
@@ -53,6 +53,7 @@ struct user_struct root_user = {
  .files  = ATOMIC_INIT(0),
  .sigpending = ATOMIC_INIT(0),
  .locked_shm     = 0,
+ .user_ns = &init_user_ns,
 #ifdef CONFIG_USER_SCHED
  .tg  = &init_task_group,
 #endif
@@ -230,16 +231,33 @@ static struct attribute *uids_attributes[] = {
  NULL
 };

+static const void *uids_mount_tag(void)
+{
+ return current->nsproxy->user_ns;
+}
+
+static struct sysfs_tag_type_operations uids_tag_type_operations = {
+ .mount_tag = uids_mount_tag,
+};
+
 /* the lifetime of user_struct is not managed by the core (now) */
 static void uids_release(struct kobject *kobj)
 {
  return;
 }

+static const void *uids_sysfs_tag(struct kobject *kobj)
+{
+ struct user_struct *up;
+ up = container_of(kobj, struct user_struct, kobj);
+ return up->user_ns;
+}
+
 static struct kobj_type uids_ktype = {
  .sysfs_ops = &kobj_sysfs_ops,
  .default_attrs = uids_attributes,
  .release = uids_release,
```

```
+ .sysfs_tag = uids_sysfs_tag,
 };

 /* create /sys/kernel/uids/<uid>/cpu_share file for this user */
@@ -272,6 +290,9 @@ int __init uids_sysfs_init(void)
  if (!uids_kset)
   return -ENOMEM;

+ sysfs_register_tag_type(SYSFS_TAG_TYPE_USERNS, &uids_tag_type_operations);
+ sysfs_make_tagged_dir(&uids_kset->kobj, SYSFS_TAG_TYPE_USERNS);
+
  return uids_user_create(&root_user);
 }

@@ -405,6 +426,7 @@ struct user_struct *alloc_uid(struct user_namespace *ns, uid_t uid)

   new->uid = uid;
   atomic_set(&new->__count, 1);
+  new->user_ns = ns;

   if (sched_create_user(new) < 0)
    goto out_free_user;
diff --git a/kernel/user_namespace.c b/kernel/user_namespace.c
index a9ab059..f67bbe0 100644
--- a/kernel/user_namespace.c
+++ b/kernel/user_namespace.c
@@ -71,6 +71,7 @@ void free_user_ns(struct kref *kref)
  struct user_namespace *ns;

  ns = container_of(kref, struct user_namespace, kref);
+ sysfs_exit_tag(SYSFS_TAG_TYPE_USERNS, ns);
  release_uids(ns);
  kfree(ns);
 }
--
1.5.3.rc6.17.g1911
```