Subject: [PATCH] various dst_ifdown routines to catch refcounting bugs
Posted by den on Thu, 27 Sep 2007 13:46:47 GMT
View Forum Message <> Reply to Message

Moving dst entries into init_net.loopback_dev is not a good thing.
This hides obvious and non-obvious ref-counting bugs.

This patch uses net_ns loopback instead of init_net loopback.
This allowes to catch various bugs like recent one in IPv6 DAD handling.

Signed-off-by: Denis V. Lunev <den@openvz.org>

```
--- ./net/core/dst.c.loop 2007-08-26 19:30:38.000000000 +0400
+++ ./net/core/dst.c 2007-08-26 19:30:38.000000000 +0400
@@ -279,11 +279,11 @@ static inline void dst_ifdown(struct dst
  if (!unregister) {
   dst->input = dst->output = dst_discard;
  } else {
-  dst->dev = init_net.loopback_dev;
+  dst->dev = dst->dev->nd_net->loopback_dev;
   dev_hold(dst->dev);
   dev_put(dev);
   if (dst->neighbour && dst->neighbour->dev == dev) {
-   dst->neighbour->dev = init_net.loopback_dev;
+   dst->neighbour->dev = dst->dev;
    dev_put(dev);
    dev_hold(dst->neighbour->dev);
   }
--- ./net/ipv4/route.c.loop 2007-08-26 19:30:38.000000000 +0400
+++ ./net/ipv4/route.c 2007-08-26 19:30:38.000000000 +0400
@@ -1402,8 +1402,9 @@ static void ipv4_dst_ifdown(struct dst_e
 {
  struct rtable *rt = (struct rtable *) dst;
  struct in_device *idev = rt->idev;
- if (dev != init_net.loopback_dev && idev && idev->dev == dev) {
-  struct in_device *loopback_idev = in_dev_get(init_net.loopback_dev);
+ if (dev != dev->nd_net->loopback_dev && idev && idev->dev == dev) {
+  struct in_device *loopback_idev =
+   in_dev_get(dev->nd_net->loopback_dev);
   if (loopback_idev) {
    rt->idev = loopback_idev;
    in_dev_put(idev);
--- ./net/ipv4/xfrm4_policy.c.loop 2007-08-26 19:30:38.000000000 +0400
+++ ./net/ipv4/xfrm4_policy.c 2007-08-26 19:30:38.000000000 +0400
@@ -306,7 +306,8 @@ static void xfrm4_dst_ifdown(struct dst_

  xdst = (struct xfrm_dst *)dst;
  if (xdst->u.rt.idev->dev == dev) {
```

```
-  struct in_device *loopback_idev = in_dev_get(init_net.loopback_dev);
+  struct in_device *loopback_idev =
+   in_dev_get(dev->nd_net->loopback_dev);
   BUG_ON(!loopback_idev);

   do {
--- ./net/ipv6/route.c.loop 2007-08-26 19:30:38.000000000 +0400
+++ ./net/ipv6/route.c 2007-08-26 19:30:38.000000000 +0400
@@ -220,9 +220,12 @@ static void ip6_dst_ifdown(struct dst_en
 {
  struct rt6_info *rt = (struct rt6_info *)dst;
  struct inet6_dev *idev = rt->rt6i_idev;
+ struct net_device *loopback_dev =
+  dev->nd_net->loopback_dev;

- if (dev != init_net.loopback_dev && idev != NULL && idev->dev == dev) {
-  struct inet6_dev *loopback_idev = in6_dev_get(init_net.loopback_dev);
+ if (dev != loopback_dev && idev != NULL && idev->dev == dev) {
+  struct inet6_dev *loopback_idev =
+   in6_dev_get(loopback_dev);
   if (loopback_idev != NULL) {
    rt->rt6i_idev = loopback_idev;
    in6_dev_put(idev);
@@ -1185,12 +1188,12 @@ int ip6_route_add(struct fib6_config *cf
  if ((cfg->fc_flags & RTF_REJECT) ||
      (dev && (dev->flags&IFF_LOOPBACK) && !(addr_type&IPV6_ADDR_LOOPBACK))) {
   /* hold loopback dev/idev if we haven't done so. */
-  if (dev != init_net.loopback_dev) {
+  if (dev != dev->nd_net->loopback_dev) {
    if (dev) {
     dev_put(dev);
     in6_dev_put(idev);
    }
-   dev = init_net.loopback_dev;
+   dev = dev->nd_net->loopback_dev;
    dev_hold(dev);
    idev = in6_dev_get(dev);
    if (!idev) {
@@ -1894,13 +1897,13 @@ struct rt6_info *addrconf_dst_alloc(stru
  if (rt == NULL)
   return ERR_PTR(-ENOMEM);

- dev_hold(init_net.loopback_dev);
+ dev_hold(idev->dev->nd_net->loopback_dev);
  in6_dev_hold(idev);

  rt->u.dst.flags = DST_HOST;
  rt->u.dst.input = ip6_input;
```

```
  rt->u.dst.output = ip6_output;
- rt->rt6i_dev = init_net.loopback_dev;
+ rt->rt6i_dev = idev->dev->nd_net->loopback_dev;
  rt->rt6i_idev = idev;
  rt->u.dst.metrics[RTAX_MTU-1] = ipv6_get_mtu(rt->rt6i_dev);
  rt->u.dst.metrics[RTAX_ADVMSS-1] = ipv6_advmss(dst_mtu(&rt->u.dst));
--- ./net/ipv6/xfrm6_policy.c.loop 2007-08-26 19:30:38.000000000 +0400
+++ ./net/ipv6/xfrm6_policy.c 2007-08-26 19:30:38.000000000 +0400
@@ -375,7 +375,8 @@ static void xfrm6_dst_ifdown(struct dst_

  xdst = (struct xfrm_dst *)dst;
  if (xdst->u.rt6.rt6i_idev->dev == dev) {
-  struct inet6_dev *loopback_idev = in6_dev_get(init_net.loopback_dev);
+  struct inet6_dev *loopback_idev =
+   in6_dev_get(dev->nd_net->loopback_dev);
   BUG_ON(!loopback_idev);

   do {
--- ./net/xfrm/xfrm_policy.c.loop 2007-08-26 19:30:38.000000000 +0400
+++ ./net/xfrm/xfrm_policy.c 2007-08-26 19:30:38.000000000 +0400
@@ -1949,7 +1949,7 @@ static int stale_bundle(struct dst_entry
 void xfrm_dst_ifdown(struct dst_entry *dst, struct net_device *dev)
 {
  while ((dst = dst->child) && dst->xfrm && dst->dev == dev) {
-  dst->dev = init_net.loopback_dev;
+  dst->dev = dev->nd_net->loopback_dev;
   dev_hold(dst->dev);
   dev_put(dev);
  }
```

Subject: Re: [PATCH] various dst_ifdown routines to catch refcounting bugs
Posted by ebiederm on Thu, 27 Sep 2007 16:27:43 GMT
View Forum Message <> Reply to Message

"Denis V. Lunev" <den@openvz.org> writes:

> Moving dst entries into init_net.loopback_dev is not a good thing.
> This hides obvious and non-obvious ref-counting bugs.

Acked-by: "Eric W. Biederman" <ebiederm@xmission.com>

To be clear using init_net.loopback is currently safe because we don't
have any destination cache entries for anything except the initial
network namespace.

I have not yet made this change simply because I haven't gotten around
to this part in my patches.

I do have a question I would like to bring up, because I like avoiding
explicit references to loopback_dev when I can.

/* Dirty hack. We did it in 2.2 (in __dst_free),
 * we have _very_ good reasons not to repeat
 * this mistake in 2.3, but we have no choice
 * now. _It_ _is_ _explicit_ _deliberate_
 * _race_ _condition_.
 *
 * Commented and originally written by Alexey.
 */

What is the race that is talked about in that comment. Can we just
assign NULL instead of the loopback device when we bring a route down.
My gut feeling is that something like:
 dst->input = dst->output = dst_discard;
may be enough. But I don't know where the deliberate race is.

I haven't traced this all of the way through but from the obvious
parts I just get this nagging feeling that something isn't quite
right.

Eric

Subject: Re: [PATCH] various dst_ifdown routines to catch refcounting bugs
Posted by davem on Thu, 27 Sep 2007 19:44:38 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Thu, 27 Sep 2007 10:27:43 -0600

> "Denis V. Lunev" <den@openvz.org> writes:
>
> > Moving dst entries into init_net.loopback_dev is not a good thing.
> > This hides obvious and non-obvious ref-counting bugs.
>
> Acked-by: "Eric W. Biederman" <ebiederm@xmission.com>

Patch applied.

> I do have a question I would like to bring up, because I like avoiding
> explicit references to loopback_dev when I can.
>
> /* Dirty hack. We did it in 2.2 (in __dst_free),
> * we have _very_ good reasons not to repeat
> * this mistake in 2.3, but we have no choice

> * now. _It_ _is_ _explicit_ _deliberate_
> * _race_ _condition_.
> *
> * Commented and originally written by Alexey.
> */
>
> What is the race that is talked about in that comment. Can we just
> assign NULL instead of the loopback device when we bring a route down.
> My gut feeling is that something like:
> dst->input = dst->output = dst_discard;
> may be enough.   But I don't know where the deliberate race is.

The packet output path accesses the cached route device
asynchronously, and we are resetting the device to be loopback without
any synchronization whatsoever. None is in fact possible, and we
don't want to add it because that would be way too expensive.

So another thread on the system can either see the original device or
the loopback one.

It all works out because as the device goes down we'll purge any
packets queued into the transmit queue and packet scheduler for that
device.
