Subject: [patch 1/1][NETNS][IPV6] protect addrconf from loopback registration
Posted by Daniel Lezcano on Mon, 12 Nov 2007 15:19:54 GMT
View Forum Message <> Reply to Message

The loopback is now dynamically allocated. The ipv6 code was written
considering the loopback is allocated before the ipv6 protocol
initialization. This is still the case when we don't use multiple
network namespaces.

In the case of the network namespaces, ipv6 notification handler is
already setup and active (done by the initial network namespace),
so when a network namespace is created, a new instance of the
loopback device, via dynamic allocation, will trigger a REGISTER event
to addrconf_notify and this one will try to setup the network device
while the ipv6 protocol is not yet initialized for the network namespace.

Because the ipv6 is relying on the fact that the loopback device will
not trigger REGISTER/UNREGISTER events, I just protect the addrconf_notify
function when the loopback register event is triggered.

In the case of multiple network namespaces, the usual ipv6 protocol
initialization will be done after the loopback initialization with
the subsystem registration mechanism.

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>
---
 net/ipv6/addrconf.c | 9 +++++++--
 1 file changed, 7 insertions(+), 2 deletions(-)

Index: linux-2.6-netns/net/ipv6/addrconf.c
===
--- linux-2.6-netns.orig/net/ipv6/addrconf.c
+++ linux-2.6-netns/net/ipv6/addrconf.c
@@ -2272,7 +2272,8 @@ static int addrconf_notify(struct notifi

  switch(event) {
  case NETDEV_REGISTER:
- if (!idev && dev->mtu >= IPV6_MIN_MTU) {
+ if (!(dev->flags & IFF_LOOPBACK) &&
+   !idev && dev->mtu >= IPV6_MIN_MTU) {
    idev = ipv6_add_dev(dev);
    if (!idev)
     return notifier_from_errno(-ENOMEM);
@@ -2366,11 +2367,15 @@ static int addrconf_notify(struct notifi
   /* MTU falled under IPV6_MIN_MTU. Stop IPv6 on this interface. */

  case NETDEV_DOWN:

```
+  addrconf_ifdown(dev, 0);
+  break;
+
   case NETDEV_UNREGISTER:
    /*
     * Remove all addresses from this interface.
     */
-  addrconf_ifdown(dev, event != NETDEV_DOWN);
+  if (!(dev->flags & IFF_LOOPBACK))
+   addrconf_ifdown(dev, 1);
    break;

   case NETDEV_CHANGENAME:

--
```

Subject: Re: [patch 1/1][NETNS][IPV6] protect addrconf from loopback registration
Posted by den on Mon, 12 Nov 2007 16:05:23 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> The loopback is now dynamically allocated. The ipv6 code was written
> considering the loopback is allocated before the ipv6 protocol
> initialization. This is still the case when we don't use multiple
> network namespaces.
>
> In the case of the network namespaces, ipv6 notification handler is
> already setup and active (done by the initial network namespace),
> so when a network namespace is created, a new instance of the
> loopback device, via dynamic allocation, will trigger a REGISTER event
> to addrconf_notify and this one will try to setup the network device
> while the ipv6 protocol is not yet initialized for the network namespace.
>
> Because the ipv6 is relying on the fact that the loopback device will
> not trigger REGISTER/UNREGISTER events, I just protect the addrconf_notify
> function when the loopback register event is triggered.
>
> In the case of multiple network namespaces, the usual ipv6 protocol
> initialization will be done after the loopback initialization with
> the subsystem registration mechanism.
>
> Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
> Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>

```
> ---
>  net/ipv6/addrconf.c |    9 +++++++--
>  1 file changed, 7 insertions(+), 2 deletions(-)
>
> Index: linux-2.6-netns/net/ipv6/addrconf.c
> ===================================================================
> --- linux-2.6-netns.orig/net/ipv6/addrconf.c
> +++ linux-2.6-netns/net/ipv6/addrconf.c
> @@ -2272,7 +2272,8 @@ static int addrconf_notify(struct notifi
>
>   switch(event) {
>   case NETDEV_REGISTER:
> -  if (!idev && dev->mtu >= IPV6_MIN_MTU) {
> +  if (!(dev->flags & IFF_LOOPBACK) &&
> +      !idev && dev->mtu >= IPV6_MIN_MTU) {
>     idev = ipv6_add_dev(dev);
>     if (!idev)
>      return notifier_from_errno(-ENOMEM);
> @@ -2366,11 +2367,15 @@ static int addrconf_notify(struct notifi
>    /* MTU falled under IPV6_MIN_MTU. Stop IPv6 on this interface. */
>
>   case NETDEV_DOWN:
> +  addrconf_ifdown(dev, 0);
> +  break;
> +
>   case NETDEV_UNREGISTER:
>    /*
>     * Remove all addresses from this interface.
>     */
> -  addrconf_ifdown(dev, event != NETDEV_DOWN);
> +  if (!(dev->flags & IFF_LOOPBACK))
> +   addrconf_ifdown(dev, 1);
>    break;
>
>   case NETDEV_CHANGENAME:
>
```

why should we care on down? we are destroying the device. It should
gone. All references to it should also gone. So, we should perform the
cleaning and remove all IPv6 addresses, so notifier should also work.

The code relies on the "persistent" loopback and this is a _bad_ thing.
This is longstanding bug in the code, that the dst_entry should have a
valid reference to a device. This is the only purpose for a loopback
persistence. Though, at the namespace death no such entries must be and
this will be checked during unregister process. This patch definitely
breaks this assumption :(

Namespaces are good to catch leakage using standard codepaths, so they
should be preserved as much as possible. So, _all_ normal down code
should be called for a loopback device in other than init_net context.

Regards,
 Den

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

---

Subject: Re: [patch 1/1][NETNS][IPV6] protect addrconf from loopback registration
Posted by Daniel Lezcano on Mon, 12 Nov 2007 16:11:29 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Daniel Lezcano wrote:
>> The loopback is now dynamically allocated. The ipv6 code was written
>> considering the loopback is allocated before the ipv6 protocol
>> initialization. This is still the case when we don't use multiple
>> network namespaces.
>>
>> In the case of the network namespaces, ipv6 notification handler is
>> already setup and active (done by the initial network namespace),
>> so when a network namespace is created, a new instance of the
>> loopback device, via dynamic allocation, will trigger a REGISTER event
>> to addrconf_notify and this one will try to setup the network device
>> while the ipv6 protocol is not yet initialized for the network namespace.
>>
>> Because the ipv6 is relying on the fact that the loopback device will
>> not trigger REGISTER/UNREGISTER events, I just protect the addrconf_notify
>> function when the loopback register event is triggered.
>>
>> In the case of multiple network namespaces, the usual ipv6 protocol
>> initialization will be done after the loopback initialization with
>> the subsystem registration mechanism.
>>
>> Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
>> Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>
>> ---
>> net/ipv6/addrconf.c | 9 +++++++--
>> 1 file changed, 7 insertions(+), 2 deletions(-)
>>
>> Index: linux-2.6-netns/net/ipv6/addrconf.c
>> ===
>> --- linux-2.6-netns.orig/net/ipv6/addrconf.c
>> +++ linux-2.6-netns/net/ipv6/addrconf.c

```
>> @@ -2272,7 +2272,8 @@ static int addrconf_notify(struct notifi
>>
>>   switch(event) {
>>   case NETDEV_REGISTER:
>> -  if (!idev && dev->mtu >= IPV6_MIN_MTU) {
>> +  if (!(dev->flags & IFF_LOOPBACK) &&
>> +      !idev && dev->mtu >= IPV6_MIN_MTU) {
>>     idev = ipv6_add_dev(dev);
>>     if (!idev)
>>      return notifier_from_errno(-ENOMEM);
>> @@ -2366,11 +2367,15 @@ static int addrconf_notify(struct notifi
>>    /* MTU falled under IPV6_MIN_MTU. Stop IPv6 on this interface. */
>>
>>   case NETDEV_DOWN:
>> +  addrconf_ifdown(dev, 0);
>> +  break;
>> +
>>   case NETDEV_UNREGISTER:
>>    /*
>>     * Remove all addresses from this interface.
>>     */
>> -  addrconf_ifdown(dev, event != NETDEV_DOWN);
>> +  if (!(dev->flags & IFF_LOOPBACK))
>> +   addrconf_ifdown(dev, 1);
>>    break;
>>
>>   case NETDEV_CHANGENAME:
>>
>
> why should we care on down? we are destroying the device. It should
> gone. All references to it should also gone. So, we should perform the
> cleaning and remove all IPv6 addresses, so notifier should also work.

We need to take care of netdev down, someone can put the loopback down
if he wants.

> The code relies on the "persistent" loopback and this is a _bad_ thing.
> This is longstanding bug in the code, that the dst_entry should have a
> valid reference to a device. This is the only purpose for a loopback
> persistence. Though, at the namespace death no such entries must be and
> this will be checked during unregister process. This patch definitely
> breaks this assumption :(
>
> Namespaces are good to catch leakage using standard codepaths, so they
> should be preserved as much as possible. So, _all_ normal down code
> should be called for a loopback device in other than init_net context.

I agree with you, this is a bug in ipv6 and the loopback; when playing
```

with ipv6 we found that the loopback is still referenced 9 times when
the system is shutdown.

The purpose of this patch is to protect the __actual__ code from the new
loopback behavior. We are looking at a more generic approach with the
namespace for ipv6, as you mentioned, namespaces are good for network
leakage detection as we create several instances of the network stack.

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

---

Subject: Re: [patch 1/1][NETNS][IPV6] protect addrconf from loopback registration
Posted by ebiederm on Mon, 12 Nov 2007 16:40:32 GMT
View Forum Message <> Reply to Message

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

> The loopback is now dynamically allocated. The ipv6 code was written
> considering the loopback is allocated before the ipv6 protocol
> initialization. This is still the case when we don't use multiple
> network namespaces.

You do know that register_netdevice_notifier delivers events
REGISTER and UP events for devices that are already up?

Thinking about it I wonder if unregister_netdevice_notifier should
actually deliver UNREGISTER events. It wouldn't change the ipv6
case as I don't believe you can unregister ipv6.

> In the case of the network namespaces, ipv6 notification handler is
> already setup and active (done by the initial network namespace),
> so when a network namespace is created, a new instance of the
> loopback device, via dynamic allocation, will trigger a REGISTER event
> to addrconf_notify and this one will try to setup the network device
> while the ipv6 protocol is not yet initialized for the network namespace.

Ok. This sounds like a race in ipv6 that should get fixed.

I know last time my patchset covered ipv6 I did send patches for several
reference counting problems. I'm surprised something bad still exists.

Anyway let's not patch around this and fix whatever the real problem.

> Because the ipv6 is relying on the fact that the loopback device will
> not trigger REGISTER/UNREGISTER events, I just protect the addrconf_notify

> function when the loopback register event is triggered.

This can't be the case REGISTER events happen.

> In the case of multiple network namespaces, the usual ipv6 protocol
> initialization will be done after the loopback initialization with
> the subsystem registration mechanism.
>
> Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
> Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>
> ---
> net/ipv6/addrconf.c | 9 +++++++--
> 1 file changed, 7 insertions(+), 2 deletions(-)
>
> Index: linux-2.6-netns/net/ipv6/addrconf.c
> ===
> --- linux-2.6-netns.orig/net/ipv6/addrconf.c
> +++ linux-2.6-netns/net/ipv6/addrconf.c
> @@ -2272,7 +2272,8 @@ static int addrconf_notify(struct notifi
>
>   switch(event) {
>   case NETDEV_REGISTER:
> - if (!idev && dev->mtu >= IPV6_MIN_MTU) {
> + if (!(dev->flags & IFF_LOOPBACK) &&
> +   !idev && dev->mtu >= IPV6_MIN_MTU) {
>   idev = ipv6_add_dev(dev);
>   if (!idev)
>   return notifier_from_errno(-ENOMEM);

This hunk is clearly bogus.

> @@ -2366,11 +2367,15 @@ static int addrconf_notify(struct notifi
>   /* MTU falled under IPV6_MIN_MTU. Stop IPv6 on this
> interface. */
>
>   case NETDEV_DOWN:
> + addrconf_ifdown(dev, 0);
> + break;
> +
>   case NETDEV_UNREGISTER:
>   /*
>   * Remove all addresses from this interface.
>   */
> - addrconf_ifdown(dev, event != NETDEV_DOWN);
> + if (!(dev->flags & IFF_LOOPBACK))
> +   addrconf_ifdown(dev, 1);

I can see how this could be a problem.

> break;
>
> case NETDEV_CHANGENAME:
>
> --

_____

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

---

Subject: Re: [patch 1/1][NETNS][IPV6] protect addrconf from loopback registration
Posted by den on Mon, 12 Nov 2007 16:47:10 GMT
View Forum Message <> Reply to Message

>> why should we care on down? we are destroying the device. It should
>> gone. All references to it should also gone. So, we should perform the
>> cleaning and remove all IPv6 addresses, so notifier should also work.
>
> We need to take care of netdev down, someone can put the loopback down
> if he wants.
>
>> The code relies on the "persistent" loopback and this is a _bad_ thing.
>> This is longstanding bug in the code, that the dst_entry should have a
>> valid reference to a device. This is the only purpose for a loopback
>> persistence. Though, at the namespace death no such entries must be and
>> this will be checked during unregister process. This patch definitely
>> breaks this assumption :(
>>
>> Namespaces are good to catch leakage using standard codepaths, so they
>> should be preserved as much as possible. So, _all_ normal down code
>> should be called for a loopback device in other than init_net context.
>
> I agree with you, this is a bug in ipv6 and the loopback; when playing
> with ipv6 we found that the loopback is still referenced 9 times when
> the system is shutdown.

Pff... I can't guess right now where the error can be :(We have
correct behavior of loopback even for IPv6 within OpenVZ. On down the
count is 0 and device is destroyed correctly without these kludges. So,
something is definitely wrong.

> The purpose of this patch is to protect the __actual__ code from the new
> loopback behavior. We are looking at a more generic approach with the
> namespace for ipv6, as you mentioned, namespaces are good for network
> leakage detection as we create several instances of the network stack.

The only kludge required is already in place. addrconf_ifdown has a

protection for init_net.
	if (dev == init_net.loopback_dev && how == 1)
		how = 0;
Other places should be untouched.

Unregister for a loopback in !init_net is a _valid_ operation and should
be clean, i.e. without kludges in the path. This is the only way to
check the ref-counting.

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

---

Subject: Re: [patch 1/1][NETNS][IPV6] protect addrconf from loopback registration
Posted by ebiederm on Mon, 12 Nov 2007 16:51:58 GMT
View Forum Message <> Reply to Message

"Denis V. Lunev" <den@sw.ru> writes:

>> Index: linux-2.6-netns/net/ipv6/addrconf.c
>> ===
>> --- linux-2.6-netns.orig/net/ipv6/addrconf.c
>> +++ linux-2.6-netns/net/ipv6/addrconf.c
>> @@ -2272,7 +2272,8 @@ static int addrconf_notify(struct notifi
>>
>>   switch(event) {
>>   case NETDEV_REGISTER:
>> - if (!idev && dev->mtu >= IPV6_MIN_MTU) {
>> + if (!(dev->flags & IFF_LOOPBACK) &&
>> +   !idev && dev->mtu >= IPV6_MIN_MTU) {

It is idev being true here for the loopback device that would
prevent things not missing the REGISTER event.

Hmm. But we do call ipv6_add_dev on loopback and now the loopback
device is practically guaranteed to be the first device so we can
probably just remove the special case in addrconf_init.

Anyway Daniels patch makes increasingly less sense the more I look
at it.

> Namespaces are good to catch leakage using standard codepaths, so they
> should be preserved as much as possible. So, _all_ normal down code
> should be called for a loopback device in other than init_net context.

In any context. After the code path is aware of multiple network namespaces

init_net should not be special in any way.

I completely agree about the ability to catch weird leakage scenarios.

Eric

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

---

Subject: Re: [patch 1/1][NETNS][IPV6] protect addrconf from loopback registration
Posted by ebiederm on Mon, 12 Nov 2007 16:59:52 GMT
View Forum Message <> Reply to Message

"Denis V. Lunev" <den@sw.ru> writes:

>>> why should we care on down? we are destroying the device. It should
>>> gone. All references to it should also gone. So, we should perform the
>>> cleaning and remove all IPv6 addresses, so notifier should also work.
>>
>> We need to take care of netdev down, someone can put the loopback down
>> if he wants.
>>
>>> The code relies on the "persistent" loopback and this is a _bad_ thing.
>>> This is longstanding bug in the code, that the dst_entry should have a
>>> valid reference to a device. This is the only purpose for a loopback
>>> persistence. Though, at the namespace death no such entries must be and
>>> this will be checked during unregister process. This patch definitely
>>> breaks this assumption :(
>>>
>>> Namespaces are good to catch leakage using standard codepaths, so they
>>> should be preserved as much as possible. So, _all_ normal down code
>>> should be called for a loopback device in other than init_net context.
>>
>> I agree with you, this is a bug in ipv6 and the loopback; when playing
>> with ipv6 we found that the loopback is still referenced 9 times when
>> the system is shutdown.
>
> Pff... I can't guess right now where the error can be :(We have
> correct behavior of loopback even for IPv6 within OpenVZ. On down the
> count is 0 and device is destroyed correctly without these kludges. So,
> something is definitely wrong.
>
>> The purpose of this patch is to protect the __actual__ code from the new
>> loopback behavior. We are looking at a more generic approach with the
>> namespace for ipv6, as you mentioned, namespaces are good for network
>> leakage detection as we create several instances of the network stack.

>
> The only kludge required is already in place. addrconf_ifdown has a
> protection for init_net.
>   if (dev == init_net.loopback_dev && how == 1)
>   how = 0;
> Other places should be untouched.
>
> Unregister for a loopback in !init_net is a _valid_ operation and should
> be clean, i.e. without kludges in the path. This is the only way to
> check the ref-counting.

Oh. Speaking of. One way to catch this kind of thing during debugging
is to instrument dev_put and dev_hold to add a print statement. Something
like:

```
/**
 * dev_put - release reference to device
 * @dev: network device
 *
 * Release reference to device to allow it to be freed.
 */
static inline void __dev_put(struct net_device *dev, char *file, char *func, int line)
{
        if (dev->flags & IFF_LOOPBACK)
         printk("%s: %s.%s.%d\n", __func__, file, file, line);
 atomic_dec(&dev->refcnt);
}

#define dev_put(DEV) __dev_put(DEV, __FILE__, __func__, __LINE__)

/**
 * dev_hold - get reference to device
 * @dev: network device
 *
 * Hold reference to device to keep it from being freed.
 */
static inline void __dev_hold(struct net_device *dev, char *file, char *func, int line)
{
        if (dev->flags & IFF_LOOPBACK)
         printk("%s: %s.%s.%d\n", __func__, file, file, line);
 atomic_inc(&dev->refcnt);
}

#define dev_hold(DEV) __dev_hold(DEV, __FILE__, __func__, __LINE__)
```

Subject: Re: [patch 1/1][NETNS][IPV6] protect addrconf from loopback registration
Posted by Daniel Lezcano on Mon, 12 Nov 2007 17:01:22 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> "Denis V. Lunev" <den@sw.ru> writes:
>
>>> Index: linux-2.6-netns/net/ipv6/addrconf.c
>>> ===
>>> --- linux-2.6-netns.orig/net/ipv6/addrconf.c
>>> +++ linux-2.6-netns/net/ipv6/addrconf.c
>>> @@ -2272,7 +2272,8 @@ static int addrconf_notify(struct notifi
>>>
>>> switch(event) {
>>> case NETDEV_REGISTER:
>>> - if (!idev && dev->mtu >= IPV6_MIN_MTU) {
>>> + if (!(dev->flags & IFF_LOOPBACK) &&
>>> +   !idev && dev->mtu >= IPV6_MIN_MTU) {
>
> It is idev being true here for the loopback device that would
> prevent things not missing the REGISTER event.
>
> Hmm. But we do call ipv6_add_dev on loopback and now the loopback
> device is practically guaranteed to be the first device so we can
> probably just remove the special case in addrconf_init.
>
> Anyway Daniels patch makes increasingly less sense the more I look
> at it.

Let me try to clarify:

  * when the init network namespace is created, the loopback is created
first, before ipv6, and the notifier call chain for ipv6 is not setup,
so the protocol does not receive the REGISTER event

  * when the init network namespace is destroyed during shutdown, the
loopback is not unregistered, so there is no UNREGISTER event

  * when we create a new network namespace, a new instance of the
loopback is created and a NETDEV_REGISTER is sent to ipv6 because the
notifier call chain has been setup by the init netns (while ipv6
protocol is not yet configured for the namespace which is being created)

  * when the network namespace exits, the loopback is unregistered after
the ipv6 protocol but the NETDEV_UNREGISTER is sent to addrconf_notify

while the ipv6 protocol has been destroyed.

The objective of the patch is to discard these events because they were never taken into account and they are not expected to be receive by ipv6 protocol.

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

---

Subject: Re: [patch 1/1][NETNS][IPV6] protect addrconf from loopback registration
Posted by ebiederm on Mon, 12 Nov 2007 19:50:53 GMT
View Forum Message <> Reply to Message

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

> Eric W. Biederman wrote:
>> "Denis V. Lunev" <den@sw.ru> writes:
>>
>>>> Index: linux-2.6-netns/net/ipv6/addrconf.c
>>>> ===
>>>> --- linux-2.6-netns.orig/net/ipv6/addrconf.c
>>>> +++ linux-2.6-netns/net/ipv6/addrconf.c
>>>> @@ -2272,7 +2272,8 @@ static int addrconf_notify(struct notifi
>>>>   switch(event) {
>>>>   case NETDEV_REGISTER:
>>>> - if (!idev && dev->mtu >= IPV6_MIN_MTU) {
>>>> + if (!(dev->flags & IFF_LOOPBACK) &&
>>>> +   !idev && dev->mtu >= IPV6_MIN_MTU) {
>>
>> It is idev being true here for the loopback device that would
>> prevent things not missing the REGISTER event.
>>
>> Hmm. But we do call ipv6_add_dev on loopback and now the loopback
>> device is practically guaranteed to be the first device so we can
>> probably just remove the special case in addrconf_init.
>>
>> Anyway Daniels patch makes increasingly less sense the more I look
>> at it.
>
> Let me try to clarify:
>
> * when the init network namespace is created, the loopback is created first,
> before ipv6, and the notifier call chain for ipv6 is not setup, so the protocol
> does not receive the REGISTER event
>

> * when the init network namespace is destroyed during shutdown, the loopback is
> not unregistered, so there is no UNREGISTER event

* When addrconf_init calls register_netdevice_notifier we receive
  NETDEV_REGISTER and NETDEV_UP for all network devices that are in
  the system including the loopback device.

> * when we create a new network namespace, a new instance of the loopback is
> created and a NETDEV_REGISTER is sent to ipv6 because the notifier call chain
> has been setup by the init netns (while ipv6 protocol is not yet configured for
> the namespace which is being created)

Possibly there may be some ordering issues here.

> * when the network namespace exits, the loopback is unregistered after the ipv6
> protocol but the NETDEV_UNREGISTER is sent to addrconf_notify while the ipv6
> protocol has been destroyed.
>
>
> The objective of the patch is to discard these events because they were never
> taken into account and they are not expected to be receive by ipv6 protocol.

My opinion is that both your analysis is slightly off (as to the cause
of your problems) and that your approach to fix your problem is wrong
because you don't untangle the knot you keep it.

...
I have register_pernet_subsys and register_per_net_device to ensure
that when we create a new network namespace all of the subsystems are
initialized before the network devices are initialize. So ipv6 should
be ready before we initialize the new loopback device comes into
existence.

The preservation of the order of the network namespace callbacks
ensures that the loopback device will be the first network device
registered, and if it helps we can take advantage of that in reference
to the weirdness from the comment below.

```
 /* The addrconf netdev notifier requires that loopback_dev
  * has it's ipv6 private information allocated and setup
  * before it can bring up and give link-local addresses
  * to other devices which are up.
  *
  * Unfortunately, loopback_dev is not necessarily the first
  * entry in the global dev_base list of net devices.  In fact,
  * it is likely to be the very last entry on that list.
```

```
  * So this causes the notifier registry below to try and
  * give link-local addresses to all devices besides loopback_dev
  * first, then loopback_dev, which cases all the non-loopback_dev
  * devices to fail to get a link-local address.
  *
  * So, as a temporary fix, allocate the ipv6 structure for
  * loopback_dev first by hand.
  * Longer term, all of the dependencies ipv6 has upon the loopback
  * device and it being up should be removed.
  */
```

We can just special case registration of the loopback device to
do:

```
 ip6_null_entry.u.dst.dev = init_net.loopback_dev;
 ip6_null_entry.rt6i_idev = in6_dev_get(init_net.loopback_dev);
#ifdef CONFIG_IPV6_MULTIPLE_TABLES
 ip6_prohibit_entry.u.dst.dev = init_net.loopback_dev;
 ip6_prohibit_entry.rt6i_idev = in6_dev_get(init_net.loopback_dev);
 ip6_blk_hole_entry.u.dst.dev = init_net.loopback_dev;
 ip6_blk_hole_entry.rt6i_idev = in6_dev_get(init_net.loopback_dev);
#endif
```

Which would remove the special case from addrconf_init.

Eric

Subject: Re: [patch 1/1][NETNS][IPV6] protect addrconf from loopback registration
Posted by dlunev on Mon, 12 Nov 2007 20:59:21 GMT
View Forum Message <> Reply to Message

> Let me try to clarify:
>
> * when the init network namespace is created, the loopback is created
> first, before ipv6, and the notifier call chain for ipv6 is not setup,
> so the protocol does not receive the REGISTER event
>
> * when the init network namespace is destroyed during shutdown, the
> loopback is not unregistered, so there is no UNREGISTER event
>
> * when we create a new network namespace, a new instance of the
> loopback is created and a NETDEV_REGISTER is sent to ipv6 because the
> notifier call chain has been setup by the init netns (while ipv6
> protocol is not yet configured for the namespace which is being created)

>
> * when the network namespace exits, the loopback is unregistered after
> the ipv6 protocol but the NETDEV_UNREGISTER is sent to addrconf_notify
> while the ipv6 protocol has been destroyed.

this should not be a problem :). IPv6 exiting code could remove in_dev6
at the protocol layer. In this case the notifier will be noop. This
approach is completely equivalent to the unloading of IPv6 module with
persistent loopback.

The registration is still the problem. May be we need to separate
registration from initialization and perform it after protocol layer for
all loopback devices? I'd like this right now, but this is tight change...

>
>
> The objective of the patch is to discard these events because they were
> never taken into account and they are not expected to be receive by ipv6
> protocol.
>

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

---

Subject: Re: [patch 1/1][NETNS][IPV6] protect addrconf from loopback registration
Posted by davem on Mon, 12 Nov 2007 22:24:55 GMT
View Forum Message <> Reply to Message

From: "Denis V. Lunev" <den@sw.ru>
Date: Mon, 12 Nov 2007 19:49:03 +0300

> Unregister for a loopback in !init_net is a _valid_ operation and should
> be clean, i.e. without kludges in the path. This is the only way to
> check the ref-counting.

For ipv6 the stack really wants to pin down the loopback
device because we need a valid inet6_dev object to reference
at all times in order to simplify the per-device SNMP
statistic bumping.

When a non-loopback device goes down, we point any existing
references to that device's idev to the loopback one instead.

I really consider taking down the loopback device to be
an invalid operation at least how things are implemented
currently.

There was a suggestion to have a "dummy" device that takes the
place of "point dangling idev refs to loopback's one". But
some people get upset when statistical events get lost, and
rightly so. Such a dummy device would either need to be
invisible to the user, or show up and be utterly confusing.

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

---

Subject: Re: [patch 1/1][NETNS][IPV6] protect addrconf from loopback registration
Posted by yoshfuji on Tue, 13 Nov 2007 01:52:29 GMT
View Forum Message <> Reply to Message

In article <m16407450i.fsf@ebiederm.dsl.xmission.com> (at Mon, 12 Nov 2007 12:50:53 -0700),
ebiederm@xmission.com (Eric W. Biederman) says:

> My opinion is that both your analysis is slightly off (as to the cause
> of your problems) and that your approach to fix your problem is wrong
> because you don't untangle the knot you keep it.
:
> I have register_pernet_subsys and register_per_net_device to ensure
> that when we create a new network namespace all of the subsystems are
> initialized before the network devices are initialize. So ipv6 should
> be ready before we initialize the new loopback device comes into
> existence.

User may not load ipv6.ko at boot, and then do "modprobe ipv6".
Do you take this into account?

--yoshfuji

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

---

Subject: Re: [patch 1/1][NETNS][IPV6] protect addrconf from loopback registration
Posted by Daniel Lezcano on Tue, 13 Nov 2007 10:55:32 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>

>> Eric W. Biederman wrote:
>>> "Denis V. Lunev" <den@sw.ru> writes:
>>>
>>>>> Index: linux-2.6-netns/net/ipv6/addrconf.c
>>>>> ===
>>>>> --- linux-2.6-netns.orig/net/ipv6/addrconf.c
>>>>> +++ linux-2.6-netns/net/ipv6/addrconf.c
>>>>> @@ -2272,7 +2272,8 @@ static int addrconf_notify(struct notifi
>>>>>   switch(event) {
>>>>>   case NETDEV_REGISTER:
>>>>> - if (!idev && dev->mtu >= IPV6_MIN_MTU) {
>>>>> + if (!(dev->flags & IFF_LOOPBACK) &&
>>>>> +   !idev && dev->mtu >= IPV6_MIN_MTU) {
>>> It is idev being true here for the loopback device that would
>>> prevent things not missing the REGISTER event.
>>>
>>> Hmm. But we do call ipv6_add_dev on loopback and now the loopback
>>> device is practically guaranteed to be the first device so we can
>>> probably just remove the special case in addrconf_init.
>>>
>>> Anyway Daniels patch makes increasingly less sense the more I look
>>> at it.
>> Let me try to clarify:
>>
>> * when the init network namespace is created, the loopback is created first,
>> before ipv6, and the notifier call chain for ipv6 is not setup, so the protocol
>> does not receive the REGISTER event
>>
>> * when the init network namespace is destroyed during shutdown, the loopback is
>> not unregistered, so there is no UNREGISTER event
>
> * When addrconf_init calls register_netdevice_notifier we receive
>   NETDEV_REGISTER and NETDEV_UP for all network devices that are in
>   the system including the loopback device.

Thanks for the information. Effectively, I missed this :|

>> * when we create a new network namespace, a new instance of the loopback is
>> created and a NETDEV_REGISTER is sent to ipv6 because the notifier call chain
>> has been setup by the init netns (while ipv6 protocol is not yet configured for
>> the namespace which is being created)
>
> Possibly there may be some ordering issues here.
>
>> * when the network namespace exits, the loopback is unregistered after the ipv6
>> protocol but the NETDEV_UNREGISTER is sent to addrconf_notify while the ipv6
>> protocol has been destroyed.
>>

>>
>> The objective of the patch is to discard these events because they were never
>> taken into account and they are not expected to be receive by ipv6 protocol.
>
> My opinion is that both your analysis is slightly off (as to the cause
> of your problems) and that your approach to fix your problem is wrong
> because you don't untangle the knot you keep it.

Yes, I will look at how to fix that properly.

> ...
> I have register_pernet_subsys and register_per_net_device to ensure
> that when we create a new network namespace all of the subsystems are
> initialized before the network devices are initialize. So ipv6 should
> be ready before we initialize the new loopback device comes into
> existence.
>
> The preservation of the order of the network namespace callbacks
> ensures that the loopback device will be the first network device
> registered, and if it helps we can take advantage of that in reference
> to the weirdness from the comment below.
>
> /* The addrconf netdev notifier requires that loopback_dev
>   * has it's ipv6 private information allocated and setup
>   * before it can bring up and give link-local addresses
>   * to other devices which are up.
>   *
>   * Unfortunately, loopback_dev is not necessarily the first
>   * entry in the global dev_base list of net devices. In fact,
>   * it is likely to be the very last entry on that list.
>   * So this causes the notifier registry below to try and
>   * give link-local addresses to all devices besides loopback_dev
>   * first, then loopback_dev, which cases all the non-loopback_dev
>   * devices to fail to get a link-local address.
>   *
>   * So, as a temporary fix, allocate the ipv6 structure for
>   * loopback_dev first by hand.
>   * Longer term, all of the dependencies ipv6 has upon the loopback
>   * device and it being up should be removed.
>   */
>
> We can just special case registration of the loopback device to
> do:
> ip6_null_entry.u.dst.dev = init_net.loopback_dev;
> ip6_null_entry.rt6i_idev = in6_dev_get(init_net.loopback_dev);
> #ifdef CONFIG_IPV6_MULTIPLE_TABLES
> ip6_prohibit_entry.u.dst.dev = init_net.loopback_dev;
> ip6_prohibit_entry.rt6i_idev = in6_dev_get(init_net.loopback_dev);

> ip6_blk_hole_entry.u.dst.dev = init_net.loopback_dev;
> ip6_blk_hole_entry.rt6i_idev = in6_dev_get(init_net.loopback_dev);
> #endif
>
> Which would remove the special case from addrconf_init.

Subject: Re: [patch 1/1][NETNS][IPV6] protect addrconf from loopback registration
Posted by ebiederm on Tue, 13 Nov 2007 12:59:38 GMT
View Forum Message <> Reply to Message

David Miller <davem@davemloft.net> writes:

Well this is a weird way to get to this part of the conversation.

> From: "Denis V. Lunev" <den@sw.ru>
> Date: Mon, 12 Nov 2007 19:49:03 +0300
>
>> Unregister for a loopback in !init_net is a _valid_ operation and should
>> be clean, i.e. without kludges in the path. This is the only way to
>> check the ref-counting.
>
> For ipv6 the stack really wants to pin down the loopback
> device because we need a valid inet6_dev object to reference
> at all times in order to simplify the per-device SNMP
> statistic bumping.
>
> When a non-loopback device goes down, we point any existing
> references to that device's idev to the loopback one instead.
>
> I really consider taking down the loopback device to be
> an invalid operation at least how things are implemented
> currently.

In a secondary network namespace the current implementation
registers the loopback device before all other network devices
in a network namespace and it unregisters the loopback device
after all other network devices.

So we don't endanger the current scheme of pointing any existing
reference to the loopback device. In fact the current ipv6 addrconf_cleanup
largely does the same thing.

Unregistering the loopback device is definitely a case where we need
to tread very carefully.

Bug we absolutely need to do all of our cleanup for a network
namespace went it goes away and that includes removing the per network
namespace copy of the loopback device.

Eric

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

---

Subject: Re: [patch 1/1][NETNS][IPV6] protect addrconf from loopback registration
Posted by ebiederm on Tue, 13 Nov 2007 13:11:37 GMT
View Forum Message <> Reply to Message

> In article <m16407450i.fsf@ebiederm.dsl.xmission.com> (at Mon, 12 Nov 2007
> 12:50:53 -0700), ebiederm@xmission.com (Eric W. Biederman) says:
>
>> My opinion is that both your analysis is slightly off (as to the cause
>> of your problems) and that your approach to fix your problem is wrong
>> because you don't untangle the knot you keep it.
> :
>> I have register_pernet_subsys and register_per_net_device to ensure
>> that when we create a new network namespace all of the subsystems are
>> initialized before the network devices are initialize. So ipv6 should
>> be ready before we initialize the new loopback device comes into
>> existence.
>
> User may not load ipv6.ko at boot, and then do "modprobe ipv6".
> Do you take this into account?

Absolutely.

In the general case the infrastructure has to work for netfilter, ipv6,
and other parts of the networking stack that can be made modular.

The only limitation is that if you update struct net to add a new field
to help a modular ipv6 the core kernel needs to be recompiled.

When you load ipv6.ko late in the game first we call the init methods
which will eventually register the per network namespace registration
methods. Then register_netdevice_notifier is called. At which point

ipv6 is ready for the registration method.

For additional network namespace (which is the case that was claimed
was in trouble) the pernet_susbsys logic initializes all of the
subsystems before it initializes any of the network devices.
Effectively persevering the initialization order that exists today
with just the init methods and register_netdevice_notifier.

Eric

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