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__)
```

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