Subject: Re: [PATCH] net: Fix race when opening a proc file while a network namespace is exiting.
Posted by paulmck on Wed, 12 Sep 2007 22:46:53 GMT
View Forum Message <> Reply to Message

On Wed, Sep 12, 2007 at 10:24:34AM -0600, Eric W. Biederman wrote:
>
> The problem: proc_net files remember which network namespace the are
> against but do not remember hold a reference count (as that would pin
> the network namespace).   So we currently have a small window where
> the reference count on a network namespace may be incremented when opening
> a /proc file when it has already gone to zero.
>
> To fix this introduce maybe_get_net and get_proc_net.
>
> maybe_get_net increments the network namespace reference count only if it is
> greater then zero, ensuring we don't increment a reference count after it
> has gone to zero.
>
> get_proc_net handles all of the magic to go from a proc inode to the network
> namespace instance and call maybe_get_net on it.
>
> PROC_NET the old accessor is removed so that we don't get confused and use
> the wrong helper function.
>
> Then I fix up the callers to use get_proc_net and handle the case case
> where get_proc_net returns NULL. In that case I return -ENXIO because
> effectively the network namespace has already gone away so the files
> we are trying to access don't exist anymore.

Looks much better!

Acked-by: Paul E. McKenney <paulmck@us.ibm.com>

> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
> ---
> fs/proc/proc_net.c | 6 ++++++
> include/linux/proc_fs.h | 5 +----
> include/net/net_namespace.h | 12 ++++++++++++
> net/core/dev.c | 6 +++++-
> net/core/dev_mcast.c | 6 +++++-
> net/netlink/af_netlink.c | 6 +++++-
> net/wireless/wext.c | 6 +++++-
> 7 files changed, 39 insertions(+), 8 deletions(-)
>
> diff --git a/fs/proc/proc_net.c b/fs/proc/proc_net.c
> index 358930a..85cc8e8 100644
> --- a/fs/proc/proc_net.c

> +++ b/fs/proc/proc_net.c
> @@ -51,6 +51,12 @@ void proc_net_remove(struct net *net, const char *name)
> }
> EXPORT_SYMBOL_GPL(proc_net_remove);
>
> +struct net *get_proc_net(const struct inode *inode)
> +{
> + return maybe_get_net(PDE_NET(PDE(inode)));
> +}
> +EXPORT_SYMBOL_GPL(get_proc_net);
> +
> static struct proc_dir_entry *proc_net_shadow;
>
> static struct dentry *proc_net_shadow_dentry(struct dentry *parent,
> diff --git a/include/linux/proc_fs.h b/include/linux/proc_fs.h
> index 5964670..20741f6 100644
> --- a/include/linux/proc_fs.h
> +++ b/include/linux/proc_fs.h
> @@ -270,10 +270,7 @@ static inline struct net *PDE_NET(struct proc_dir_entry *pde)
>   return pde->parent->data;
> }
>
> -static inline struct net *PROC_NET(const struct inode *inode)
> -{
> - return PDE_NET(PDE(inode));
> -}
> +struct net *get_proc_net(const struct inode *inode);
>
> struct proc_maps_private {
>   struct pid *pid;
> diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
> index fac42db..dda03f3 100644
> --- a/include/net/net_namespace.h
> +++ b/include/net/net_namespace.h
> @@ -39,6 +39,18 @@ static inline struct net *get_net(struct net *net)
>   return net;
> }
>
> +static inline struct net *maybe_get_net(struct net *net)
> +{
> + /* Used when we know struct net exists but we
> + * aren't guaranteed a previous reference count
> + * exists. If the reference count is zero this
> + * function fails and returns NULL.
> + */
> + if (!atomic_inc_not_zero(&net->count))
> + net = NULL;
> + return net;

> +}
> +
> static inline void put_net(struct net *net)
> {
>   if (atomic_dec_and_test(&net->count))
> diff --git a/net/core/dev.c b/net/core/dev.c
> index a22a95d..f119dc0 100644
> --- a/net/core/dev.c
> +++ b/net/core/dev.c
> @@ -2446,7 +2446,11 @@ static int dev_seq_open(struct inode *inode, struct file *file)
>   res = seq_open(file, &dev_seq_ops);
>   if (!res) {
>   seq = file->private_data;
> - seq->private = get_net(PROC_NET(inode));
> + seq->private = get_proc_net(inode);
> + if (!seq->private) {
> +   seq_release(inode, file);
> +   res = -ENXIO;
> + }
>   }
>   return res;
> }
> diff --git a/net/core/dev_mcast.c b/net/core/dev_mcast.c
> index 1c4f619..896b0ca 100644
> --- a/net/core/dev_mcast.c
> +++ b/net/core/dev_mcast.c
> @@ -246,7 +246,11 @@ static int dev_mc_seq_open(struct inode *inode, struct file *file)
>   res = seq_open(file, &dev_mc_seq_ops);
>   if (!res) {
>   seq = file->private_data;
> - seq->private = get_net(PROC_NET(inode));
> + seq->private = get_proc_net(inode);
> + if (!seq->private) {
> +   seq_release(inode, file);
> +   res = -ENXIO;
> + }
>   }
>   return res;
> }
> diff --git a/net/netlink/af_netlink.c b/net/netlink/af_netlink.c
> index 3029f86..dc9f8c2 100644
> --- a/net/netlink/af_netlink.c
> +++ b/net/netlink/af_netlink.c
> @@ -1859,7 +1859,11 @@ static int netlink_seq_open(struct inode *inode, struct file *file)
>
>   seq = file->private_data;
>   seq->private = iter;
> - iter->net = get_net(PROC_NET(inode));

```
> + iter->net = get_proc_net(inode);
> + if (!iter->net) {
> +  seq_release_private(inode, file);
> +  return -ENXIO;
> + }
>   return 0;
>  }
>
> diff --git a/net/wireless/wext.c b/net/wireless/wext.c
> index e8b3409..85e5f9d 100644
> --- a/net/wireless/wext.c
> +++ b/net/wireless/wext.c
> @@ -678,7 +678,11 @@ static int wireless_seq_open(struct inode *inode, struct file *file)
>   res = seq_open(file, &wireless_seq_ops);
>   if (!res) {
>    seq = file->private_data;
> -  seq->private = get_net(PROC_NET(inode));
> +  seq->private = get_proc_net(inode);
> +  if (!seq->private) {
> +   seq_release(inode, file);
> +   res = -ENXIO;
> +  }
>   }
>   return res;
>  }
> --
> 1.5.3.rc6.17.g1911
>
```

_____

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