
Subject: [patch 3/3][netns] remove timewait sockets at cleanup
Posted by [Daniel Lezcano](#) on Mon, 24 Sep 2007 13:29:38 GMT
[View Forum Message](#) <> [Reply to Message](#)

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

Denis Lunev spotted that if we take a reference to the network namespace with the timewait sockets, we will need to wait for their expiration to have the network namespace freed. This is a waste of time, the timewait sockets are for avoiding to receive a duplicate packet from the network, if the network namespace is freed, the network stack is removed, so no chance to receive any packets from the outside world.

This patchset remove/destroy the timewait sockets when the network namespace is freed.

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>

net/core/net_namespace.c | 55 +++++
1 file changed, 55 insertions(+)

Index: linux-2.6-netns/net/core/net_namespace.c

=====

--- linux-2.6-netns.orig/net/core/net_namespace.c

+++ linux-2.6-netns/net/core/net_namespace.c

@@ -7,6 +7,7 @@

#include <linux/sched.h>

#include <linux/kallsyms.h>

#include <net/net_namespace.h>

+#include <net/tcp.h>

/*

* Our network namespace constructor/destructor lists

@@ -59,6 +60,57 @@ static int pernet_debug_max=-1;

module_param(pernet_debug, int, S_IRUSR|S_IWUSR);

module_param(pernet_debug_max, int, S_IRUSR|S_IWUSR);

*/

+ * This function is called when the network namespace is freed.

+ * It allows to wipe out all timewait sockets. They are useless

+ * because the network namespace will be destroyed and the network

+ * stack with it, so no risks to have a duplicate packet coming

+ * from the outside world.

+ */

+static void clean_up_timewait(struct net *net)

+

+ struct inet_timewait_sock *tw;

+ struct sock *sk;

```

+ struct hlist_node *node, *tmp;
+ int h;
+
+ /* Browse the the established hash table */
+ for (h = 0; h < (tcp_hashinfo.ehash_size); h++) {
+     struct inet_ehash_bucket *head =
+         inet_ehash_bucket(&tcp_hashinfo, h);
+
+     /* Take the lock and disable bh */
+     write_lock_bh(&head->lock);
+
+     sk_for_each_safe(sk, node, tmp, &head->twchain) {
+
+         tw = inet_tws(sk);
+         if (tw->tw_net != net)
+             continue;
+
+         /* deschedule the timewait socket */
+         spin_lock(&tcp_death_row.death_lock);
+         if (inet_tws_del_dead_node(tw)) {
+             inet_tws_put(tw);
+             if (--tcp_death_row.tw_count == 0)
+                 del_timer(&tcp_death_row.tw_timer);
+         }
+         spin_unlock(&tcp_death_row.death_lock);
+
+         /* remove it from the established hash table */
+         __inet_tws_unhash(tw);
+
+         /* remove it from the bind hash table */
+         inet_tws_unbhash(tw, tcp_death_row.hashinfo);
+
+         /* last put */
+         inet_tws_put(tw);
+     }
+
+     write_unlock_bh(&head->lock);
+ }
+
+ static void cleanup_net(struct work_struct *work)
+ {
+     struct pernet_operations *ops;
@@ -96,6 +148,9 @@ static void cleanup_net(struct work_struct *work)
+
+     mutex_unlock(&net_mutex);
+
+     /* The timewait sockets are pointless */

```

```
+ clean_up_timewait(net);  
+  
/* Ensure there are no outstanding rcu callbacks using this  
 * network namespace.  
 */  
  
--
```

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