
Subject: [PATCH 3/3] Fix race in ipv6_flowlabel_opt() when inserting two labels
Posted by [Pavel Emelianov](#) on Thu, 18 Oct 2007 11:59:14 GMT

[View Forum Message](#) <> [Reply to Message](#)

In the IPV6_FL_A_GET case the hash is checked for flowlabels with the given label. If it is not found, the lock, protecting the hash, is dropped to be re-get for writing. After this a newly allocated entry is inserted, but no checks are performed to catch a classical SMP race, when the conflicting label may be inserted on another cpu.

Use the (currently unused) return value from fl_intern() to return the conflicting entry (if found) and re-check, whether we can reuse it (IPV6_FL_F_EXCL) or return -EEXISTS.

Also add the comment, about why not re-lookup the current sock for conflicting flowlabel entry.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
diff --git a/net/ipv6/ip6_flowlabel.c b/net/ipv6/ip6_flowlabel.c
index f40a086..e55ae1a 100644
--- a/net/ipv6/ip6_flowlabel.c
+++ b/net/ipv6/ip6_flowlabel.c
@@ -154,8 +154,10 @@ static void ip6_fl_gc(unsigned long dummy)
     write_unlock(&ip6_fl_lock);
 }

-static int fl_intern(struct ip6_flowlabel *fl, __be32 label)
+static struct ip6_flowlabel *fl_intern(struct ip6_flowlabel *fl, __be32 label)
 {
+ struct ip6_flowlabel *lfl;
+
     fl->label = label & IPV6_FLOWLABEL_MASK;

     write_lock_bh(&ip6_fl_lock);
@@ -163,12 +165,26 @@ static int fl_intern(struct ip6_flowlabel *fl, __be32 label)
     for (;;) {
         fl->label = htonl(net_random()) & IPV6_FLOWLABEL_MASK;
         if (fl->label) {
- struct ip6_flowlabel *lfl;
- lfl = __fl_lookup(fl->label);
- if (lfl == NULL)
- break;
+ struct ip6_flowlabel *lfl;
+ lfl = __fl_lookup(fl->label);
+ if (lfl == NULL)
+ break;
         }
     }
 }
```

```

+ } else {
+ /*
+  * we dropper the ip6_fl_lock, so this entry could reappear
+  * and we need to recheck with it.
+  *
+  * OTOH no need to search the active socket first, like it is
+  * done in ipv6_flowlabel_opt - sock is locked, so new entry
+  * with the same label can only appear on another sock
+  */
+ lfl = __fl_lookup(fl->label);
+ if (lfl != NULL) {
+   atomic_inc(&lfl->users);
+   write_unlock_bh(&ip6_fl_lock);
+   return lfl;
+ }
+ }

fl->lastuse = jiffies;
@@ -176,7 +192,7 @@ static int fl_intern(struct ip6_flowlabel *fl, __be32 label)
fl_ht[FL_HASH(fl->label)] = fl;
atomic_inc(&fl_size);
write_unlock_bh(&ip6_fl_lock);
- return 0;
+ return NULL;
}

@@ -429,7 +445,8 @@ int ipv6_flowlabel_opt(struct sock *sk, char __user *optval, int optlen)
struct in6_flowlabel_req freq;
struct ipv6_fl_socklist *sfl1=NULL;
struct ipv6_fl_socklist *sfl, **sflp;
- struct ip6_flowlabel *fl;
+ struct ip6_flowlabel *fl, *fl1 = NULL;
+

if (optlen < sizeof(freq))
return -EINVAL;
@@ -485,8 +502,6 @@ int ipv6_flowlabel_opt(struct sock *sk, char __user *optval, int optlen)
sfl1 = kmalloc(sizeof(*sfl1), GFP_KERNEL);

if (freq.flr_label) {
- struct ip6_flowlabel *fl1 = NULL;
-
err = -EEXIST;
read_lock_bh(&ip6_sk_fl_lock);
for (sfl = np->ipv6_fl_list; sfl; sfl = sfl->next) {
@@ -505,6 +520,7 @@ int ipv6_flowlabel_opt(struct sock *sk, char __user *optval, int optlen)
if (fl1 == NULL)

```

```

    fl1 = fl_lookup(freq.flr_label);
    if (fl1) {
+recheck:
        err = -EEXIST;
        if (freq.flr_flags & IPV6_FL_F_EXCL)
            goto release;
@@ -543,9 +559,9 @@ release:
        if (sfl1 == NULL || (err = mem_check(sk)) != 0)
            goto done;

- err = fl_intern(fl, freq.flr_label);
- if (err)
-     goto done;
+ fl1 = fl_intern(fl, freq.flr_label);
+ if (fl1 != NULL)
+     goto recheck;

    if (!freq.flr_label) {
        if (copy_to_user(&((struct in6_flowlabel_req __user *) optval)->flr_label,

```
