Subject: [PATCH 4/4] Fix the race between sk_filter_(de|at)tach and sk_clone()
Posted by Pavel Emelianov on Wed, 17 Oct 2007 09:53:40 GMT
View Forum Message <> Reply to Message

The proposed fix is to delay the reference counter decrement
until the quiescent state pass. This will give sk_clone() a
chance to get the reference on the cloned filter.

Regular sk_filter_uncharge can happen from the sk_free() only
and there's no need in delaying the put - the socket is dead
anyway and is to be release itself.

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

---

```
diff --git a/include/net/sock.h b/include/net/sock.h
index b9cfe12..43fc3fa 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -905,16 +905,6 @@ static inline int sk_filter(struct sock *sk, struct sk_buff *skb)
 }

 /**
- *  sk_filter_rcu_free: Free a socket filter
- * @rcu: rcu_head that contains the sk_filter to free
- */
-static inline void sk_filter_rcu_free(struct rcu_head *rcu)
-{
- struct sk_filter *fp = container_of(rcu, struct sk_filter, rcu);
- kfree(fp);
-}
-
-/**
  * sk_filter_release: Release a socket filter
  * @sk: socket
  * @fp: filter to remove
@@ -925,7 +915,7 @@ static inline void sk_filter_rcu_free(struct rcu_head *rcu)
 static inline void sk_filter_release(struct sk_filter *fp)
 {
  if (atomic_dec_and_test(&fp->refcnt))
-  call_rcu_bh(&fp->rcu, sk_filter_rcu_free);
+  kfree(fp);
 }

 static inline void sk_filter_uncharge(struct sock *sk, struct sk_filter *fp)
diff --git a/net/core/filter.c b/net/core/filter.c
index 54dddc9..b8bc7d3 100644
```

```
--- a/net/core/filter.c
+++ b/net/core/filter.c
@@ -387,6 +387,25 @@ int sk_chk_filter(struct sock_filter *filter, int flen)
 }

 /**
+ *  sk_filter_rcu_release: Release a socket filter by rcu_head
+ * @rcu: rcu_head that contains the sk_filter to free
+ */
+static void sk_filter_rcu_release(struct rcu_head *rcu)
+{
+ struct sk_filter *fp = container_of(rcu, struct sk_filter, rcu);
+
+ sk_filter_release(fp);
+}
+
+static void sk_filter_delayed_uncharge(struct sock *sk, struct sk_filter *fp)
+{
+ unsigned int size = sk_filter_len(fp);
+
+ atomic_sub(size, &sk->sk_omem_alloc);
+ call_rcu_bh(&fp->rcu, sk_filter_rcu_release);
+}
+
+/**
  * sk_attach_filter - attach a socket filter
  * @fprog: the filter program
  * @sk: the socket to use
@@ -428,7 +447,7 @@ int sk_attach_filter(struct sock_fprog *fprog, struct sock *sk)
  rcu_assign_pointer(sk->sk_filter, fp);
  rcu_read_unlock_bh();

- sk_filter_uncharge(sk, old_fp);
+ sk_filter_delayed_uncharge(sk, old_fp);
  return 0;
 }

@@ -441,7 +460,7 @@ int sk_detach_filter(struct sock *sk)
  filter = rcu_dereference(sk->sk_filter);
  if (filter) {
   rcu_assign_pointer(sk->sk_filter, NULL);
-  sk_filter_uncharge(sk, filter);
+  sk_filter_delayed_uncharge(sk, filter);
   ret = 0;
  }
  rcu_read_unlock_bh();
--
1.5.3.4
```