
Subject: [PATCH 1/2][NETFILTER] Consolidate nf_sockopt and compat_nf_sockopt
Posted by [Pavel Emelianov](#) on Thu, 01 Nov 2007 15:55:35 GMT
[View Forum Message](#) <> [Reply to Message](#)

Both lookup the nf_sockopt_ops object to call the get/set callbacks from, but they perform it in a completely similar way.

Introduce the helper for finding the ops.

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

```
diff --git a/net/netfilter/nf_sockopt.c b/net/netfilter/nf_sockopt.c
index aa28315..a5e5e30 100644
--- a/net/netfilter/nf_sockopt.c
+++ b/net/netfilter/nf_sockopt.c
@@ -61,48 +61,59 @@ void nf_unregister_sockopt(struct nf_sockopt_ops *reg)
 }
EXPORT_SYMBOL(nf_unregister_sockopt);

-/* Call get/setsockopt() */
-static int nf_sockopt(struct sock *sk, int pf, int val,
-    char __user *opt, int *len, int get)
+static struct nf_sockopt_ops *nf_sockopt_find(struct sock *sk, int pf,
+    int val, int get)
 {
     struct list_head *i;
     struct nf_sockopt_ops *ops;
-    int ret;

     if (sk->sk_net != &init_net)
-        return -ENOPROTOOPT;
+        return ERR_PTR(-ENOPROTOOPT);

     if (mutex_lock_interruptible(&nf_sockopt_mutex) != 0)
-        return -EINTR;
+        return ERR_PTR(-EINTR);

     list_for_each(i, &nf_sockopts) {
         ops = (struct nf_sockopt_ops *)i;
         if (ops->pf == pf) {
             if (!try_module_get(ops->owner))
                 goto out_nosup;
+
+            if (get) {
-                if (val >= ops->get_optmin
-                    && val < ops->get_optmax) {
```

```

- mutex_unlock(&nf_sockopt_mutex);
- ret = ops->get(sk, val, opt, len);
+ if (val >= ops->get_optmin &&
+     val < ops->get_optmax)
+     goto out;
- }
+ } else {
- if (val >= ops->set_optmin
-     && val < ops->set_optmax) {
-     mutex_unlock(&nf_sockopt_mutex);
-     ret = ops->set(sk, val, opt, *len);
+ if (val >= ops->set_optmin &&
+     val < ops->set_optmax)
+     goto out;
- }
+ }
+ module_put(ops->owner);
+ }
+ }
- out_nosup:
+out_nosup:
+ ops = ERR_PTR(-ENOPROTOOPT);
+out:
+     mutex_unlock(&nf_sockopt_mutex);
- return -ENOPROTOOPT;
+ return ops;
+}
+
+/* Call get/setsockopt() */
+static int nf_sockopt(struct sock *sk, int pf, int val,
+    char __user *opt, int *len, int get)
+{
+ struct nf_sockopt_ops *ops;
+ int ret;
+
+ ops = nf_sockopt_find(sk, pf, val, get);
+ if (IS_ERR(ops))
+     return PTR_ERR(ops);
+
+ if (get)
+     ret = ops->get(sk, val, opt, len);
+ else
+     ret = ops->set(sk, val, opt, *len);

- out:
+ module_put(ops->owner);
+ return ret;
+ }

```

```

@@ -124,56 +135,25 @@ EXPORT_SYMBOL(nf_getsockopt);
static int compat_nf_sockopt(struct sock *sk, int pf, int val,
    char __user *opt, int *len, int get)
{
- struct list_head *i;
- struct nf_sockopt_ops *ops;
- int ret;

- if (sk->sk_net != &init_net)
- return -ENOPROTOOPT;
-
- if (mutex_lock_interruptible(&nf_sockopt_mutex) != 0)
- return -EINTR;
-
- list_for_each(i, &nf_sockopts) {
- ops = (struct nf_sockopt_ops *)i;
- if (ops->pf == pf) {
- if (!try_module_get(ops->owner))
- goto out_nosup;
-
- if (get) {
- if (val >= ops->get_optmin
- && val < ops->get_optmax) {
- mutex_unlock(&nf_sockopt_mutex);
- if (ops->compat_get)
- ret = ops->compat_get(sk,
- val, opt, len);
- else
- ret = ops->get(sk,
- val, opt, len);
- goto out;
- }
- } else {
- if (val >= ops->set_optmin
- && val < ops->set_optmax) {
- mutex_unlock(&nf_sockopt_mutex);
- if (ops->compat_set)
- ret = ops->compat_set(sk,
- val, opt, *len);
- else
- ret = ops->set(sk,
- val, opt, *len);
- goto out;
- }
- }
- module_put(ops->owner);
- }
}

```

```

+ ops = nf_sockopt_find(sk, pf, val, get);
+ if (IS_ERR(ops))
+ return PTR_ERR(ops);
+
+ if (get) {
+ if (ops->compat_get)
+ ret = ops->compat_get(sk, val, opt, len);
+ else
+ ret = ops->get(sk, val, ops, len);
+ } else {
+ if (ops->compat_set)
+ ret = ops->compat_set(sk, val, ops, *len);
+ else
+ ret = ops->set(sk, val, ops, *len);
+ }
- out_nosup:
- mutex_unlock(&nf_sockopt_mutex);
- return -ENOPROTOOPT;

- out:
  module_put(ops->owner);
  return ret;
}
--

```

1.5.3.4
