Subject: [PATCH] [RESEND] small possible memory leak in FIB rules
Posted by den on Wed, 07 Nov 2007 13:33:26 GMT
View Forum Message <> Reply to Message

This patch fixes a small memory leak. Default fib rules can be deleted by
the user if the rule does not carry FIB_RULE_PERMANENT flag, f.e. by
 ip rule flush

Such a rule will not be freed as the ref-counter has 2 on start and becomes
clearly unreachable after removal.

Signed-off-by: Denis V. Lunev <den@openvz.org>
Acked-by: Alexey Kuznetsov <kuznet@ms2.inr.ac.ru>

---

 include/net/fib_rules.h | 3 ++
 net/core/fib_rules.c | 22 ++++++++++++++++++++
 net/decnet/dn_rules.c | 13 +-----------
 net/ipv4/fib_rules.c | 51 ++
 net/ipv6/fib6_rules.c | 37 ++++++++++++++++++++++++++++++++++
 5 files changed, 62 insertions(+), 64 deletions(-)

```
diff --git a/include/net/fib_rules.h b/include/net/fib_rules.h
--- a/include/net/fib_rules.h
+++ b/include/net/fib_rules.h
@@ -107,4 +107,7 @@ extern int   fib_rules_unregister(struct fib_rules_ops *);
 extern int   fib_rules_lookup(struct fib_rules_ops *,
        struct flowi *, int flags,
        struct fib_lookup_arg *);
+extern int   fib_default_rule_add(struct fib_rules_ops *,
+             u32 pref, u32 table,
+          u32 flags);
 #endif
diff --git a/net/core/fib_rules.c b/net/core/fib_rules.c
index 13de6f5..848132b 100644
--- a/net/core/fib_rules.c
+++ b/net/core/fib_rules.c
@@ -18,6 +18,28 @@
 static LIST_HEAD(rules_ops);
 static DEFINE_SPINLOCK(rules_mod_lock);

+int fib_default_rule_add(struct fib_rules_ops *ops,
+    u32 pref, u32 table, u32 flags)
+{
+ struct fib_rule *r;
```

```
+
+ r = kzalloc(ops->rule_size, GFP_KERNEL);
+ if (r == NULL)
+  return -ENOMEM;
+
+ atomic_set(&r->refcnt, 1);
+ r->action = FR_ACT_TO_TBL;
+ r->pref = pref;
+ r->table = table;
+ r->flags = flags;
+
+ /* The lock is not required here, the list in unreacheable
+  * at the moment this function is called */
+ list_add_tail(&r->list, &ops->rules_list);
+ return 0;
+}
+EXPORT_SYMBOL(fib_default_rule_add);
+
 static void notify_rule_change(int event, struct fib_rule *rule,
           struct fib_rules_ops *ops, struct nlmsghdr *nlh,
           u32 pid);
diff --git a/net/decnet/dn_rules.c b/net/decnet/dn_rules.c
index ddd3f04..ffebea0 100644
--- a/net/decnet/dn_rules.c
+++ b/net/decnet/dn_rules.c
@@ -48,15 +48,6 @@ struct dn_fib_rule
  u8   flags;
 };

-static struct dn_fib_rule default_rule = {
- .common = {
-  .refcnt =  ATOMIC_INIT(2),
-  .pref =   0x7fff,
-  .table =  RT_TABLE_MAIN,
-  .action =  FR_ACT_TO_TBL,
- },
-};
-

 int dn_fib_lookup(struct flowi *flp, struct dn_fib_res *res)
 {
@@ -262,8 +253,8 @@ static struct fib_rules_ops dn_fib_rules_ops = {

 void __init dn_fib_rules_init(void)
 {
- list_add_tail(&default_rule.common.list,
-   &dn_fib_rules_ops.rules_list);
+ BUG_ON(fib_default_rule_add(&dn_fib_rules_ops, 0x7fff,
```

```
+               RT_TABLE_MAIN, 0));
  fib_rules_register(&dn_fib_rules_ops);
 }

diff --git a/net/ipv4/fib_rules.c b/net/ipv4/fib_rules.c
index f16839c..a0ada3a 100644
--- a/net/ipv4/fib_rules.c
+++ b/net/ipv4/fib_rules.c
@@ -49,33 +49,6 @@ struct fib4_rule
 #endif
 };

-static struct fib4_rule default_rule = {
- .common = {
-  .refcnt = ATOMIC_INIT(2),
-  .pref =  0x7FFF,
-  .table = RT_TABLE_DEFAULT,
-  .action = FR_ACT_TO_TBL,
- },
-};
-
-static struct fib4_rule main_rule = {
- .common = {
-  .refcnt = ATOMIC_INIT(2),
-  .pref =  0x7FFE,
-  .table = RT_TABLE_MAIN,
-  .action = FR_ACT_TO_TBL,
- },
-};
-
-static struct fib4_rule local_rule = {
- .common = {
-  .refcnt = ATOMIC_INIT(2),
-  .table = RT_TABLE_LOCAL,
-  .action = FR_ACT_TO_TBL,
-  .flags = FIB_RULE_PERMANENT,
- },
-};
-
 #ifdef CONFIG_NET_CLS_ROUTE
 u32 fib_rules_tclass(struct fib_result *res)
 {
@@ -319,11 +292,27 @@ static struct fib_rules_ops fib4_rules_ops = {
  .owner  = THIS_MODULE,
 };

-void __init fib4_rules_init(void)
+static int __init fib_default_rules_init(void)
```

```
 {
- list_add_tail(&local_rule.common.list, &fib4_rules_ops.rules_list);
- list_add_tail(&main_rule.common.list, &fib4_rules_ops.rules_list);
- list_add_tail(&default_rule.common.list, &fib4_rules_ops.rules_list);
+ int err;
+
+ err = fib_default_rule_add(&fib4_rules_ops, 0,
+       RT_TABLE_LOCAL, FIB_RULE_PERMANENT);
+ if (err < 0)
+  return err;
+ err = fib_default_rule_add(&fib4_rules_ops, 0x7FFE,
+       RT_TABLE_MAIN, 0);
+ if (err < 0)
+  return err;
+ err = fib_default_rule_add(&fib4_rules_ops, 0x7FFF,
+       RT_TABLE_DEFAULT, 0);
+ if (err < 0)
+  return err;
+ return 0;
+}

+void __init fib4_rules_init(void)
+{
+ BUG_ON(fib_default_rules_init());
  fib_rules_register(&fib4_rules_ops);
 }
diff --git a/net/ipv6/fib6_rules.c b/net/ipv6/fib6_rules.c
index 706622a..428c6b0 100644
--- a/net/ipv6/fib6_rules.c
+++ b/net/ipv6/fib6_rules.c
@@ -31,25 +31,6 @@ struct fib6_rule

 static struct fib_rules_ops fib6_rules_ops;

-static struct fib6_rule main_rule = {
- .common = {
-  .refcnt = ATOMIC_INIT(2),
-  .pref =  0x7FFE,
-  .action = FR_ACT_TO_TBL,
-  .table = RT6_TABLE_MAIN,
- },
-};
-
-static struct fib6_rule local_rule = {
- .common = {
-  .refcnt = ATOMIC_INIT(2),
-  .pref =  0,
-  .action = FR_ACT_TO_TBL,
```

```
-  .table = RT6_TABLE_LOCAL,
-  .flags = FIB_RULE_PERMANENT,
- },
-};
-
 struct dst_entry *fib6_rule_lookup(struct flowi *fl, int flags,
        pol_lookup_t lookup)
 {
@@ -270,11 +251,23 @@ static struct fib_rules_ops fib6_rules_ops = {
  .owner   = THIS_MODULE,
 };

-void __init fib6_rules_init(void)
+static int __init fib6_default_rules_init(void)
 {
- list_add_tail(&local_rule.common.list, &fib6_rules_ops.rules_list);
- list_add_tail(&main_rule.common.list, &fib6_rules_ops.rules_list);
+ int err;
+
+ err = fib_default_rule_add(&fib6_rules_ops, 0,
+       RT6_TABLE_LOCAL, FIB_RULE_PERMANENT);
+ if (err < 0)
+  return err;
+ err = fib_default_rule_add(&fib6_rules_ops, 0x7FFE, RT6_TABLE_MAIN, 0);
+ if (err < 0)
+  return err;
+ return 0;
+}

+void __init fib6_rules_init(void)
+{
+ BUG_ON(fib6_default_rules_init());
  fib_rules_register(&fib6_rules_ops);
 }
```