Subject: [PATCH 2/5] netns netfilter: per-netns xt_tables
Posted by Alexey Dobriyan on Mon, 21 Jan 2008 14:52:52 GMT
View Forum Message <> Reply to Message

In fact all we want is per-netns set of rules, however doing that will
unnecessary complicate routines such as ipt_hook()/ipt_do_table, so
make full xt_table array per-netns.

Every user stubbed with init_net for a while.

Signed-off-by: Alexey Dobriyan <adobriyan@sw.ru>
---

 include/linux/netfilter/x_tables.h | 6 ++++--
 include/net/net_namespace.h | 4 ++++
 include/net/netns/x_tables.h | 10 ++++++++++
 net/ipv4/netfilter/arp_tables.c | 12 ++++++++------
 net/ipv4/netfilter/ip_tables.c | 12 ++++++++------
 net/ipv6/netfilter/ip6_tables.c | 12 ++++++++------
 net/netfilter/x_tables.c | 35 +++++++++++++++++++++++++----------
 7 files changed, 60 insertions(+), 31 deletions(-)

--- a/include/linux/netfilter/x_tables.h
+++ b/include/linux/netfilter/x_tables.h
@@ -335,7 +335,8 @@ extern int xt_check_target(const struct xt_target *target, unsigned short
family
       unsigned int size, const char *table, unsigned int hook,
       unsigned short proto, int inv_proto);

-extern struct xt_table *xt_register_table(struct xt_table *table,
+extern struct xt_table *xt_register_table(struct net *net,
+   struct xt_table *table,
        struct xt_table_info *bootstrap,
        struct xt_table_info *newinfo);
 extern void *xt_unregister_table(struct xt_table *table);
@@ -352,7 +353,8 @@ extern struct xt_target *xt_request_find_target(int af, const char *name,
 extern int xt_find_revision(int af, const char *name, u8 revision, int target,
        int *err);

-extern struct xt_table *xt_find_table_lock(int af, const char *name);
+extern struct xt_table *xt_find_table_lock(struct net *net, int af,
+   const char *name);
 extern void xt_table_unlock(struct xt_table *t);

 extern int xt_proto_init(int af);
--- a/include/net/net_namespace.h
+++ b/include/net/net_namespace.h
@@ -12,6 +12,7 @@

```
 #include <net/netns/packet.h>
 #include <net/netns/ipv4.h>
 #include <net/netns/ipv6.h>
+#include <net/netns/x_tables.h>

 struct proc_dir_entry;
 struct net_device;
@@ -56,6 +57,9 @@ struct net {
 #if defined(CONFIG_IPV6) || defined(CONFIG_IPV6_MODULE)
  struct netns_ipv6 ipv6;
 #endif
+#ifdef CONFIG_NETFILTER
+ struct netns_xt  xt;
+#endif
 };

 #ifdef CONFIG_NET
--- /dev/null
+++ b/include/net/netns/x_tables.h
@@ -0,0 +1,10 @@
+#ifndef __NETNS_X_TABLES_H
+#define __NETNS_X_TABLES_H
+
+#include <linux/list.h>
+#include <linux/net.h>
+
+struct netns_xt {
+ struct list_head tables[NPROTO];
+};
+#endif
--- a/net/ipv4/netfilter/arp_tables.c
+++ b/net/ipv4/netfilter/arp_tables.c
@@ -870,7 +870,7 @@ static int get_info(void __user *user, int *len, int compat)
  if (compat)
   xt_compat_lock(NF_ARP);
 #endif
- t = try_then_request_module(xt_find_table_lock(NF_ARP, name),
+ t = try_then_request_module(xt_find_table_lock(&init_net, NF_ARP, name),
         "arptable_%s", name);
  if (t && !IS_ERR(t)) {
   struct arpt_getinfo info;
@@ -926,7 +926,7 @@ static int get_entries(struct arpt_get_entries __user *uptr, int *len)
   return -EINVAL;
  }

- t = xt_find_table_lock(NF_ARP, get.name);
+ t = xt_find_table_lock(&init_net, NF_ARP, get.name);
  if (t && !IS_ERR(t)) {
```

```
   struct xt_table_info *private = t->private;
   duprintf("t->private->number = %u\n",
@@ -966,7 +966,7 @@ static int __do_replace(const char *name, unsigned int valid_hooks,
   goto out;
  }

- t = try_then_request_module(xt_find_table_lock(NF_ARP, name),
+ t = try_then_request_module(xt_find_table_lock(&init_net, NF_ARP, name),
         "arptable_%s", name);
  if (!t || IS_ERR(t)) {
   ret = t ? PTR_ERR(t) : -ENOENT;
@@ -1132,7 +1132,7 @@ static int do_add_counters(void __user *user, unsigned int len, int
compat)
   goto free;
  }

- t = xt_find_table_lock(NF_ARP, name);
+ t = xt_find_table_lock(&init_net, NF_ARP, name);
  if (!t || IS_ERR(t)) {
   ret = t ? PTR_ERR(t) : -ENOENT;
   goto free;
@@ -1604,7 +1604,7 @@ static int compat_get_entries(struct compat_arpt_get_entries __user
*uptr,
  }

  xt_compat_lock(NF_ARP);
- t = xt_find_table_lock(NF_ARP, get.name);
+ t = xt_find_table_lock(&init_net, NF_ARP, get.name);
  if (t && !IS_ERR(t)) {
   struct xt_table_info *private = t->private;
   struct xt_table_info info;
@@ -1751,7 +1751,7 @@ int arpt_register_table(struct arpt_table *table,
   return ret;
  }

- new_table = xt_register_table(table, &bootstrap, newinfo);
+ new_table = xt_register_table(&init_net, table, &bootstrap, newinfo);
  if (IS_ERR(new_table)) {
   xt_free_table_info(newinfo);
   return PTR_ERR(new_table);
--- a/net/ipv4/netfilter/ip_tables.c
+++ b/net/ipv4/netfilter/ip_tables.c
@@ -1112,7 +1112,7 @@ static int get_info(void __user *user, int *len, int compat)
  if (compat)
   xt_compat_lock(AF_INET);
 #endif
- t = try_then_request_module(xt_find_table_lock(AF_INET, name),
+ t = try_then_request_module(xt_find_table_lock(&init_net, AF_INET, name),
```

```
        "iptable_%s", name);
  if (t && !IS_ERR(t)) {
   struct ipt_getinfo info;
@@ -1170,7 +1170,7 @@ get_entries(struct ipt_get_entries __user *uptr, int *len)
   return -EINVAL;
  }

- t = xt_find_table_lock(AF_INET, get.name);
+ t = xt_find_table_lock(&init_net, AF_INET, get.name);
  if (t && !IS_ERR(t)) {
   struct xt_table_info *private = t->private;
   duprintf("t->private->number = %u\n", private->number);
@@ -1208,7 +1208,7 @@ __do_replace(const char *name, unsigned int valid_hooks,
   goto out;
  }

- t = try_then_request_module(xt_find_table_lock(AF_INET, name),
+ t = try_then_request_module(xt_find_table_lock(&init_net, AF_INET, name),
         "iptable_%s", name);
  if (!t || IS_ERR(t)) {
   ret = t ? PTR_ERR(t) : -ENOENT;
@@ -1383,7 +1383,7 @@ do_add_counters(void __user *user, unsigned int len, int compat)
   goto free;
  }

- t = xt_find_table_lock(AF_INET, name);
+ t = xt_find_table_lock(&init_net, AF_INET, name);
  if (!t || IS_ERR(t)) {
   ret = t ? PTR_ERR(t) : -ENOENT;
   goto free;
@@ -1924,7 +1924,7 @@ compat_get_entries(struct compat_ipt_get_entries __user *uptr, int
*len)
  }

  xt_compat_lock(AF_INET);
- t = xt_find_table_lock(AF_INET, get.name);
+ t = xt_find_table_lock(&init_net, AF_INET, get.name);
  if (t && !IS_ERR(t)) {
   struct xt_table_info *private = t->private;
   struct xt_table_info info;
@@ -2075,7 +2075,7 @@ int ipt_register_table(struct xt_table *table, const struct ipt_replace
*repl)
   return ret;
  }

- new_table = xt_register_table(table, &bootstrap, newinfo);
+ new_table = xt_register_table(&init_net, table, &bootstrap, newinfo);
  if (IS_ERR(new_table)) {
```

```
   xt_free_table_info(newinfo);
   return PTR_ERR(new_table);
--- a/net/ipv6/netfilter/ip6_tables.c
+++ b/net/ipv6/netfilter/ip6_tables.c
@@ -1138,7 +1138,7 @@ static int get_info(void __user *user, int *len, int compat)
  if (compat)
   xt_compat_lock(AF_INET6);
 #endif
- t = try_then_request_module(xt_find_table_lock(AF_INET6, name),
+ t = try_then_request_module(xt_find_table_lock(&init_net, AF_INET6, name),
         "ip6table_%s", name);
  if (t && !IS_ERR(t)) {
   struct ip6t_getinfo info;
@@ -1196,7 +1196,7 @@ get_entries(struct ip6t_get_entries __user *uptr, int *len)
   return -EINVAL;
  }

- t = xt_find_table_lock(AF_INET6, get.name);
+ t = xt_find_table_lock(&init_net, AF_INET6, get.name);
  if (t && !IS_ERR(t)) {
   struct xt_table_info *private = t->private;
   duprintf("t->private->number = %u\n", private->number);
@@ -1235,7 +1235,7 @@ __do_replace(const char *name, unsigned int valid_hooks,
   goto out;
  }

- t = try_then_request_module(xt_find_table_lock(AF_INET6, name),
+ t = try_then_request_module(xt_find_table_lock(&init_net, AF_INET6, name),
         "ip6table_%s", name);
  if (!t || IS_ERR(t)) {
   ret = t ? PTR_ERR(t) : -ENOENT;
@@ -1410,7 +1410,7 @@ do_add_counters(void __user *user, unsigned int len, int compat)
   goto free;
  }

- t = xt_find_table_lock(AF_INET6, name);
+ t = xt_find_table_lock(&init_net, AF_INET6, name);
  if (!t || IS_ERR(t)) {
   ret = t ? PTR_ERR(t) : -ENOENT;
   goto free;
@@ -1950,7 +1950,7 @@ compat_get_entries(struct compat_ip6t_get_entries __user *uptr, int
*len)
  }

  xt_compat_lock(AF_INET6);
- t = xt_find_table_lock(AF_INET6, get.name);
+ t = xt_find_table_lock(&init_net, AF_INET6, get.name);
  if (t && !IS_ERR(t)) {
```

```
   struct xt_table_info *private = t->private;
   struct xt_table_info info;
@@ -2101,7 +2101,7 @@ int ip6t_register_table(struct xt_table *table, const struct ip6t_replace
*repl)
   return ret;
  }

- new_table = xt_register_table(table, &bootstrap, newinfo);
+ new_table = xt_register_table(&init_net, table, &bootstrap, newinfo);
  if (IS_ERR(new_table)) {
   xt_free_table_info(newinfo);
   return PTR_ERR(new_table);
--- a/net/netfilter/x_tables.c
+++ b/net/netfilter/x_tables.c
@@ -44,7 +44,6 @@ struct xt_af {
  struct mutex mutex;
  struct list_head match;
  struct list_head target;
- struct list_head tables;
 #ifdef CONFIG_COMPAT
  struct mutex compat_mutex;
  struct compat_delta *compat_offsets;
@@ -597,14 +596,14 @@ void xt_free_table_info(struct xt_table_info *info)
 EXPORT_SYMBOL(xt_free_table_info);

 /* Find table by name, grabs mutex & ref.  Returns ERR_PTR() on error. */
-struct xt_table *xt_find_table_lock(int af, const char *name)
+struct xt_table *xt_find_table_lock(struct net *net, int af, const char *name)
 {
  struct xt_table *t;

  if (mutex_lock_interruptible(&xt[af].mutex) != 0)
   return ERR_PTR(-EINTR);

- list_for_each_entry(t, &xt[af].tables, list)
+ list_for_each_entry(t, &net->xt.tables[af], list)
   if (strcmp(t->name, name) == 0 && try_module_get(t->me))
    return t;
  mutex_unlock(&xt[af].mutex);
@@ -660,7 +659,7 @@ xt_replace_table(struct xt_table *table,
 }
 EXPORT_SYMBOL_GPL(xt_replace_table);

-struct xt_table *xt_register_table(struct xt_table *table,
+struct xt_table *xt_register_table(struct net *net, struct xt_table *table,
        struct xt_table_info *bootstrap,
        struct xt_table_info *newinfo)
 {
```

```
@@ -673,7 +672,7 @@ struct xt_table *xt_register_table(struct xt_table *table,
   goto out;

  /* Don't autoload: we'd eat our tail... */
- list_for_each_entry(t, &xt[table->af].tables, list) {
+ list_for_each_entry(t, &net->xt.tables[table->af], list) {
   if (strcmp(t->name, table->name) == 0) {
    ret = -EEXIST;
    goto unlock;
@@ -692,7 +691,7 @@ struct xt_table *xt_register_table(struct xt_table *table,
  /* save number of initial entries */
  private->initial_entries = private->number;

- list_add(&table->list, &xt[table->af].tables);
+ list_add(&table->list, &net->xt.tables[table->af]);
  mutex_unlock(&xt[table->af].mutex);
  return table;

@@ -744,7 +743,7 @@ static struct list_head *type2list(u_int16_t af, u_int16_t type)
   list = &xt[af].match;
   break;
  case TABLE:
-  list = &xt[af].tables;
+  list = &init_net.xt.tables[af];
   break;
  default:
   list = NULL;
@@ -919,10 +918,22 @@ void xt_proto_fini(int af)
 }
 EXPORT_SYMBOL_GPL(xt_proto_fini);

+static int __net_init xt_net_init(struct net *net)
+{
+ int i;
+
+ for (i = 0; i < NPROTO; i++)
+  INIT_LIST_HEAD(&net->xt.tables[i]);
+ return 0;
+}
+
+static struct pernet_operations xt_net_ops = {
+ .init = xt_net_init,
+};

 static int __init xt_init(void)
 {
- int i;
+ int i, rv;
```

```
  xt = kmalloc(sizeof(struct xt_af) * NPROTO, GFP_KERNEL);
  if (!xt)
@@ -936,13 +947,16 @@ static int __init xt_init(void)
 #endif
   INIT_LIST_HEAD(&xt[i].target);
   INIT_LIST_HEAD(&xt[i].match);
-  INIT_LIST_HEAD(&xt[i].tables);
  }
- return 0;
+ rv = register_pernet_subsys(&xt_net_ops);
+ if (rv < 0)
+  kfree(xt);
+ return rv;
 }

 static void __exit xt_fini(void)
 {
+ unregister_pernet_subsys(&xt_net_ops);
  kfree(xt);
 }
```

Subject: Re: [PATCH 2/5] netns netfilter: per-netns xt_tables
Posted by Patrick McHardy on Tue, 22 Jan 2008 16:51:45 GMT
View Forum Message <> Reply to Message

Alexey Dobriyan wrote:
> In fact all we want is per-netns set of rules, however doing that will
> unnecessary complicate routines such as ipt_hook()/ipt_do_table, so
> make full xt_table array per-netns.
>
> Every user stubbed with init_net for a while.

Applied.
