

---

Subject: [PATCH] ipt and ipt\_compat checks unification

Posted by [Mishin Dmitry](#) on Mon, 11 Dec 2006 13:48:02 GMT

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

---

Matches and targets verification is duplicated in normal and compat processing ways. This patch refactors code in order to remove this.

Signed-off-by: Dmitry Mishin <dim@openvz.org>

```
---
ip_tables.c | 176 ++++++-----
1 file changed, 79 insertions(+), 97 deletions(-)
---
```

```
diff --git a/net/ipv4/netfilter/ip_tables.c b/net/ipv4/netfilter/ip_tables.c
```

```
index 0ff2956..83ebbeb 100644
```

```
--- a/net/ipv4/netfilter/ip_tables.c
```

```
+++ b/net/ipv4/netfilter/ip_tables.c
```

```
@@ -484,7 +484,47 @@ cleanup_match(struct ipt_entry_match *m,
 }
```

```
static inline int
```

```
-check_match(struct ipt_entry_match *m,
+check_entry(struct ipt_entry *e, const char *name)
```

```
+{
```

```
+ struct ipt_entry_target *t;
```

```
+
```

```
+ if (!ip_checkentry(&e->ip)) {
```

```
+ duprintf("ip_tables: ip check failed %p %s.\n", e, name);
```

```
+ return -EINVAL;
```

```
+ }
```

```
+
```

```
+ if (e->target_offset + sizeof(struct ipt_entry_target) > e->next_offset)
```

```
+ return -EINVAL;
```

```
+
```

```
+ t = ipt_get_target(e);
```

```
+ if (e->target_offset + t->u.target_size > e->next_offset)
```

```
+ return -EINVAL;
```

```
+
```

```
+ return 0;
```

```
+}
```

```
+
```

```
+static inline int check_match(struct ipt_entry_match *m, const char *name,
```

```
+ const struct ipt_ip *ip, unsigned int hookmask)
```

```
+{
```

```
+ struct ipt_match *match;
```

```
+ int ret;
```

```
+
```

```
+ match = m->u.kernel.match;
```

```
+ ret = xt_check_match(match, AF_INET, m->u.match_size - sizeof(*m),
```

```

+   name, hookmask, ip->proto,
+   ip->invflags & IPT_INV_PROTO);
+ if (!ret && m->u.kernel.match->checkentry
+   && !m->u.kernel.match->checkentry(name, ip, match, m->data,
+   hookmask)) {
+   duprintf("ip_tables: check failed for `%s'.\n",
+   m->u.kernel.match->name);
+   ret = -EINVAL;
+ }
+ return ret;
+}
+
+static inline int
+find_check_match(struct ipt_entry_match *m,
+   const char *name,
+   const struct ipt_ip *ip,
+   unsigned int hookmask,
@@ -497,26 +537,15 @@ check_match(struct ipt_entry_match *m,
+   m->u.user.revision),
+   "ipt_%s", m->u.user.name);
+ if (IS_ERR(match) || !match) {
- duprintf("check_match: `%s' not found\n", m->u.user.name);
+ duprintf("find_check_match: `%s' not found\n", m->u.user.name);
+   return match ? PTR_ERR(match) : -ENOENT;
+ }
+   m->u.kernel.match = match;

- ret = xt_check_match(match, AF_INET, m->u.match_size - sizeof(*m),
-   name, hookmask, ip->proto,
-   ip->invflags & IPT_INV_PROTO);
+ ret = check_match(m, name, ip, hookmask);
+   if (ret)
+     goto err;

- if (m->u.kernel.match->checkentry
-   && !m->u.kernel.match->checkentry(name, ip, match, m->data,
-   hookmask)) {
-   duprintf("ip_tables: check failed for `%s'.\n",
-   m->u.kernel.match->name);
-   ret = -EINVAL;
-   goto err;
- }
-
+ (*i)++;
+   return 0;
err:
@@ -524,10 +553,29 @@ err:
+   return ret;

```

```
}
```

```
-static struct ipt_target ipt_standard_target;  
+static inline int check_target(struct ipt_entry *e, const char *name)  
+{  
+ struct ipt_entry_target *t;  
+ struct ipt_target *target;  
+ int ret;  
+  
+ t = ipt_get_target(e);  
+ target = t->u.kernel.target;  
+ ret = xt_check_target(target, AF_INET, t->u.target_size - sizeof(*t),  
+ name, e->comefrom, e->ip.proto,  
+ e->ip.invflags & IPT_INV_PROTO);  
+ if (!ret && t->u.kernel.target->checkentry  
+ && !t->u.kernel.target->checkentry(name, e, target,  
+ t->data, e->comefrom)) {  
+ duprintf("ip_tables: check failed for '%s'.\n",  
+ t->u.kernel.target->name);  
+ ret = -EINVAL;  
+ }  
+ return ret;  
+}
```

```
static inline int
```

```
-check_entry(struct ipt_entry *e, const char *name, unsigned int size,  
+find_check_entry(struct ipt_entry *e, const char *name, unsigned int size,  
unsigned int *i)
```

```
{
```

```
struct ipt_entry_target *t;
```

```
@@ -535,49 +583,32 @@ check_entry(struct ipt_entry *e, const c  
int ret;  
unsigned int j;
```

```
- if (!ip_checkentry(&e->ip)) {  
- duprintf("ip_tables: ip check failed %p %s.\n", e, name);  
- return -EINVAL;  
- }  
-  
- if (e->target_offset + sizeof(struct ipt_entry_target) > e->next_offset)  
- return -EINVAL;  
+ ret = check_entry(e, name);  
+ if (ret)  
+ return ret;
```

```
j = 0;
```

```
- ret = IPT_MATCH_ITERATE(e, check_match, name, &e->ip, e->comefrom, &j);  
+ ret = IPT_MATCH_ITERATE(e, find_check_match, name, &e->ip,
```

```

+   e->comefrom, &j);
  if (ret != 0)
    goto cleanup_matches;

  t = ipt_get_target(e);
- ret = -EINVAL;
- if (e->target_offset + t->u.target_size > e->next_offset)
-   goto cleanup_matches;
  target = try_then_request_module(xt_find_target(AF_INET,
    t->u.user.name,
    t->u.user.revision),
    "ipt_%s", t->u.user.name);
  if (IS_ERR(target) || !target) {
-   duprintf("check_entry: `%s' not found\n", t->u.user.name);
+   duprintf("find_check_entry: `%s' not found\n", t->u.user.name);
    ret = target ? PTR_ERR(target) : -ENOENT;
    goto cleanup_matches;
  }
  t->u.kernel.target = target;

- ret = xt_check_target(target, AF_INET, t->u.target_size - sizeof(*t),
-   name, e->comefrom, e->ip.proto,
-   e->ip.invflags & IPT_INV_PROTO);
+ ret = check_target(e, name);
  if (ret)
    goto err;

- if (t->u.kernel.target->checkentry
-   && !t->u.kernel.target->checkentry(name, e, target, t->data,
-   e->comefrom)) {
-   duprintf("ip_tables: check failed for `%s'.\n",
-   t->u.kernel.target->name);
-   ret = -EINVAL;
-   goto err;
- }
-
  (*i)++;
  return 0;
err:
@@ -712,7 +743,7 @@ translate_table(const char *name,
/* Finally, each sanity check must pass */
i = 0;
ret = IPT_ENTRY_ITERATE(entry0, newinfo->size,
-   check_entry, name, size, &i);
+   find_check_entry, name, size, &i);

  if (ret != 0) {
    IPT_ENTRY_ITERATE(entry0, newinfo->size,

```

```

@@ -1452,14 +1483,9 @@ check_compat_entry_size_and_hooks(struct
    return -EINVAL;
}

```

```

- if (!ip_checkentry(&e->ip)) {
- duprintf("ip_tables: ip check failed %p %s.\n", e, name);
- return -EINVAL;
- }
-
- if (e->target_offset + sizeof(struct compat_xt_entry_target) >
-     e->next_offset)
- return -EINVAL;
+ ret = check_entry(e, name);
+ if (ret)
+ return ret;

```

```

    off = 0;
    entry_offset = (void *)e - (void *)base;
@@ -1470,15 +1496,13 @@ check_compat_entry_size_and_hooks(struct
    goto cleanup_matches;

```

```

    t = ipt_get_target(e);
- ret = -EINVAL;
- if (e->target_offset + t->u.target_size > e->next_offset)
- goto cleanup_matches;
    target = try_then_request_module(xt_find_target(AF_INET,
        t->u.user.name,
        t->u.user.revision),
        "ipt_%s", t->u.user.name);
    if (IS_ERR(target) || !target) {
- duprintf("check_entry: `%s' not found\n", t->u.user.name);
+ duprintf("check_compat_entry_size_and_hooks: `%s' not found\n",
+     t->u.user.name);
        ret = target ? PTR_ERR(target) : -ENOENT;
        goto cleanup_matches;
    }

```

```

@@ -1555,57 +1579,15 @@ static int compat_copy_entry_from_user(s
    return ret;
}

```

```

-static inline int compat_check_match(struct ipt_entry_match *m, const char *name,
-    const struct ipt_ip *ip, unsigned int hookmask)
- {
- struct ipt_match *match;
- int ret;
-
- match = m->u.kernel.match;
- ret = xt_check_match(match, AF_INET, m->u.match_size - sizeof(*m),

```

```

-     name, hookmask, ip->proto,
-     ip->invflags & IPT_INV_PROTO);
- if (!ret && m->u.kernel.match->checkentry
-     && !m->u.kernel.match->checkentry(name, ip, match, m->data,
-     hookmask)) {
-     duprintf("ip_tables: compat: check failed for `%s'.\n",
-     m->u.kernel.match->name);
-     ret = -EINVAL;
- }
- return ret;
-}
-
-static inline int compat_check_target(struct ipt_entry *e, const char *name)
-{
- struct ipt_entry_target *t;
- struct ipt_target *target;
- int ret;
-
- t = ipt_get_target(e);
- target = t->u.kernel.target;
- ret = xt_check_target(target, AF_INET, t->u.target_size - sizeof(*t),
-     name, e->comefrom, e->ip.proto,
-     e->ip.invflags & IPT_INV_PROTO);
- if (!ret && t->u.kernel.target->checkentry
-     && !t->u.kernel.target->checkentry(name, e, target,
-     t->data, e->comefrom)) {
-     duprintf("ip_tables: compat: check failed for `%s'.\n",
-     t->u.kernel.target->name);
-     ret = -EINVAL;
- }
- return ret;
-}
-
static inline int compat_check_entry(struct ipt_entry *e, const char *name)
{
    int ret;

- ret = IPT_MATCH_ITERATE(e, compat_check_match, name, &e->ip,
-     e->comefrom);
+ ret = IPT_MATCH_ITERATE(e, check_match, name, &e->ip, e->comefrom);
    if (ret)
        return ret;

- return compat_check_target(e, name);
+ return check_target(e, name);
}

static int

```

---