Subject: [PATCH 2/5] netns netfilter: per-netns IPv6 FILTER, MANGLE, RAW
Posted by Alexey Dobriyan on Thu, 24 Jan 2008 12:27:48 GMT
View Forum Message <> Reply to Message

Now it's possible to list and manipulate per-netns ip6tables rules.
Filtering decisions are based on init_net's table so far.

P.S.: remove init_net check in inet6_create() to see the effect

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

```
 include/net/netns/ipv6.h            |    6 +++++
 net/ipv6/netfilter/ip6table_filter.c |   40 ++++++++++++++++++++++++++----------
 net/ipv6/netfilter/ip6table_mangle.c |   40 ++++++++++++++++++++++++++----------
 net/ipv6/netfilter/ip6table_raw.c    |   38 ++++++++++++++++++++++++++---------
 4 files changed, 92 insertions(+), 32 deletions(-)

--- a/include/net/netns/ipv6.h
+++ b/include/net/netns/ipv6.h
@@ -31,5 +31,10 @@ struct netns_ipv6 {
  struct ipv6_devconf *devconf_all;
  struct ipv6_devconf *devconf_dflt;
  struct netns_frags frags;
+#ifdef CONFIG_NETFILTER
+ struct xt_table  *ip6table_filter;
+ struct xt_table  *ip6table_mangle;
+ struct xt_table  *ip6table_raw;
+#endif
 };
 #endif
--- a/net/ipv6/netfilter/ip6table_filter.c
+++ b/net/ipv6/netfilter/ip6table_filter.c
@@ -26,7 +26,7 @@ static struct
  struct ip6t_replace repl;
  struct ip6t_standard entries[3];
  struct ip6t_error term;
-} initial_table __initdata = {
+} initial_table __net_initdata = {
  .repl = {
   .name = "filter",
   .valid_hooks = FILTER_VALID_HOOKS,
@@ -51,14 +51,13 @@ static struct
  .term = IP6T_ERROR_INIT,  /* ERROR */
 };

-static struct xt_table __packet_filter = {
+static struct xt_table packet_filter = {
```

```
  .name  = "filter",
  .valid_hooks = FILTER_VALID_HOOKS,
  .lock  = RW_LOCK_UNLOCKED,
  .me  = THIS_MODULE,
  .af  = AF_INET6,
 };
-static struct xt_table *packet_filter;

 /* The work comes in here from netfilter.c. */
 static unsigned int
@@ -68,7 +67,7 @@ ip6t_hook(unsigned int hook,
   const struct net_device *out,
   int (*okfn)(struct sk_buff *))
 {
- return ip6t_do_table(skb, hook, in, out, packet_filter);
+ return ip6t_do_table(skb, hook, in, out, init_net.ipv6.ip6table_filter);
 }

 static unsigned int
@@ -88,7 +87,7 @@ ip6t_local_out_hook(unsigned int hook,
  }
 #endif

- return ip6t_do_table(skb, hook, in, out, packet_filter);
+ return ip6t_do_table(skb, hook, in, out, init_net.ipv6.ip6table_filter);
 }

 static struct nf_hook_ops ip6t_ops[] __read_mostly = {
@@ -119,6 +118,26 @@ static struct nf_hook_ops ip6t_ops[] __read_mostly = {
 static int forward = NF_ACCEPT;
 module_param(forward, bool, 0000);

+static int __net_init ip6table_filter_net_init(struct net *net)
+{
+ /* Register table */
+ net->ipv6.ip6table_filter =
+  ip6t_register_table(net, &packet_filter, &initial_table.repl);
+ if (IS_ERR(net->ipv6.ip6table_filter))
+  return PTR_ERR(net->ipv6.ip6table_filter);
+ return 0;
+}
+
+static void __net_exit ip6table_filter_net_exit(struct net *net)
+{
+ ip6t_unregister_table(net->ipv6.ip6table_filter);
+}
+
+static struct pernet_operations ip6table_filter_net_ops = {
```

```
+ .init = ip6table_filter_net_init,
+ .exit = ip6table_filter_net_exit,
+};
+
 static int __init ip6table_filter_init(void)
 {
  int ret;
@@ -131,10 +150,9 @@ static int __init ip6table_filter_init(void)
  /* Entry 1 is the FORWARD hook */
  initial_table.entries[1].target.verdict = -forward - 1;

- /* Register table */
- packet_filter = ip6t_register_table(&init_net, &__packet_filter, &initial_table.repl);
- if (IS_ERR(packet_filter))
-  return PTR_ERR(packet_filter);
+ ret = register_pernet_subsys(&ip6table_filter_net_ops);
+ if (ret < 0)
+  return ret;

  /* Register hooks */
  ret = nf_register_hooks(ip6t_ops, ARRAY_SIZE(ip6t_ops));
@@ -144,14 +162,14 @@ static int __init ip6table_filter_init(void)
  return ret;

  cleanup_table:
- ip6t_unregister_table(packet_filter);
+ unregister_pernet_subsys(&ip6table_filter_net_ops);
  return ret;
 }

 static void __exit ip6table_filter_fini(void)
 {
  nf_unregister_hooks(ip6t_ops, ARRAY_SIZE(ip6t_ops));
- ip6t_unregister_table(packet_filter);
+ unregister_pernet_subsys(&ip6table_filter_net_ops);
 }

 module_init(ip6table_filter_init);
--- a/net/ipv6/netfilter/ip6table_mangle.c
+++ b/net/ipv6/netfilter/ip6table_mangle.c
@@ -26,7 +26,7 @@ static struct
  struct ip6t_replace repl;
  struct ip6t_standard entries[5];
  struct ip6t_error term;
-} initial_table __initdata = {
+} initial_table __net_initdata = {
  .repl = {
   .name = "mangle",
```

```
   .valid_hooks = MANGLE_VALID_HOOKS,
@@ -57,14 +57,13 @@ static struct
  .term = IP6T_ERROR_INIT,  /* ERROR */
 };

-static struct xt_table __packet_mangler = {
+static struct xt_table packet_mangler = {
  .name  = "mangle",
  .valid_hooks = MANGLE_VALID_HOOKS,
  .lock  = RW_LOCK_UNLOCKED,
  .me  = THIS_MODULE,
  .af  = AF_INET6,
 };
-static struct xt_table *packet_mangler;

 /* The work comes in here from netfilter.c. */
 static unsigned int
@@ -74,7 +73,7 @@ ip6t_route_hook(unsigned int hook,
   const struct net_device *out,
   int (*okfn)(struct sk_buff *))
 {
- return ip6t_do_table(skb, hook, in, out, packet_mangler);
+ return ip6t_do_table(skb, hook, in, out, init_net.ipv6.ip6table_mangle);
 }

 static unsigned int
@@ -109,7 +108,7 @@ ip6t_local_hook(unsigned int hook,
  /* flowlabel and prio (includes version, which shouldn't change either */
  flowlabel = *((u_int32_t *)ipv6_hdr(skb));

- ret = ip6t_do_table(skb, hook, in, out, packet_mangler);
+ ret = ip6t_do_table(skb, hook, in, out, init_net.ipv6.ip6table_mangle);

  if (ret != NF_DROP && ret != NF_STOLEN
   && (memcmp(&ipv6_hdr(skb)->saddr, &saddr, sizeof(saddr))
@@ -159,14 +158,33 @@ static struct nf_hook_ops ip6t_ops[] __read_mostly = {
  },
 };

+static int __net_init ip6table_mangle_net_init(struct net *net)
+{
+ /* Register table */
+ net->ipv6.ip6table_mangle =
+  ip6t_register_table(net, &packet_mangler, &initial_table.repl);
+ if (IS_ERR(net->ipv6.ip6table_mangle))
+  return PTR_ERR(net->ipv6.ip6table_mangle);
+ return 0;
+}
```

```
+
+static void __net_exit ip6table_mangle_net_exit(struct net *net)
+{
+ ip6t_unregister_table(net->ipv6.ip6table_mangle);
+}
+
+static struct pernet_operations ip6table_mangle_net_ops = {
+ .init = ip6table_mangle_net_init,
+ .exit = ip6table_mangle_net_exit,
+};
+
 static int __init ip6table_mangle_init(void)
 {
  int ret;

- /* Register table */
- packet_mangler = ip6t_register_table(&init_net, &__packet_mangler, &initial_table.repl);
- if (IS_ERR(packet_mangler))
-  return PTR_ERR(packet_mangler);
+ ret = register_pernet_subsys(&ip6table_mangle_net_ops);
+ if (ret < 0)
+  return ret;

  /* Register hooks */
  ret = nf_register_hooks(ip6t_ops, ARRAY_SIZE(ip6t_ops));
@@ -176,14 +194,14 @@ static int __init ip6table_mangle_init(void)
  return ret;

  cleanup_table:
- ip6t_unregister_table(packet_mangler);
+ unregister_pernet_subsys(&ip6table_mangle_net_ops);
  return ret;
 }

 static void __exit ip6table_mangle_fini(void)
 {
  nf_unregister_hooks(ip6t_ops, ARRAY_SIZE(ip6t_ops));
- ip6t_unregister_table(packet_mangler);
+ unregister_pernet_subsys(&ip6table_mangle_net_ops);
 }

 module_init(ip6table_mangle_init);
--- a/net/ipv6/netfilter/ip6table_raw.c
+++ b/net/ipv6/netfilter/ip6table_raw.c
@@ -13,7 +13,7 @@ static struct
  struct ip6t_replace repl;
  struct ip6t_standard entries[2];
  struct ip6t_error term;
```

```
-} initial_table __initdata = {
+} initial_table __net_initdata = {
  .repl = {
   .name = "raw",
   .valid_hooks = RAW_VALID_HOOKS,
@@ -35,14 +35,13 @@ static struct
  .term = IP6T_ERROR_INIT,  /* ERROR */
 };

-static struct xt_table __packet_raw = {
+static struct xt_table packet_raw = {
  .name = "raw",
  .valid_hooks = RAW_VALID_HOOKS,
  .lock = RW_LOCK_UNLOCKED,
  .me = THIS_MODULE,
  .af = AF_INET6,
 };
-static struct xt_table *packet_raw;

 /* The work comes in here from netfilter.c. */
 static unsigned int
@@ -52,7 +51,7 @@ ip6t_hook(unsigned int hook,
   const struct net_device *out,
   int (*okfn)(struct sk_buff *))
 {
- return ip6t_do_table(skb, hook, in, out, packet_raw);
+ return ip6t_do_table(skb, hook, in, out, init_net.ipv6.ip6table_raw);
 }

 static struct nf_hook_ops ip6t_ops[] __read_mostly = {
@@ -72,14 +71,33 @@ static struct nf_hook_ops ip6t_ops[] __read_mostly = {
  },
 };

+static int __net_init ip6table_raw_net_init(struct net *net)
+{
+ /* Register table */
+ net->ipv6.ip6table_raw =
+  ip6t_register_table(net, &packet_raw, &initial_table.repl);
+ if (IS_ERR(net->ipv6.ip6table_raw))
+  return PTR_ERR(net->ipv6.ip6table_raw);
+ return 0;
+}
+
+static void __net_exit ip6table_raw_net_exit(struct net *net)
+{
+ ip6t_unregister_table(net->ipv6.ip6table_raw);
+}
```

```
+
+static struct pernet_operations ip6table_raw_net_ops = {
+ .init = ip6table_raw_net_init,
+ .exit = ip6table_raw_net_exit,
+};
+
 static int __init ip6table_raw_init(void)
 {
  int ret;

- /* Register table */
- packet_raw = ip6t_register_table(&init_net, &__packet_raw, &initial_table.repl);
- if (IS_ERR(packet_raw))
-  return PTR_ERR(packet_raw);
+ ret = register_pernet_subsys(&ip6table_raw_net_ops);
+ if (ret < 0)
+  return ret;

  /* Register hooks */
  ret = nf_register_hooks(ip6t_ops, ARRAY_SIZE(ip6t_ops));
@@ -89,14 +107,14 @@ static int __init ip6table_raw_init(void)
  return ret;

  cleanup_table:
- ip6t_unregister_table(packet_raw);
+ unregister_pernet_subsys(&ip6table_raw_net_ops);
  return ret;
 }

 static void __exit ip6table_raw_fini(void)
 {
  nf_unregister_hooks(ip6t_ops, ARRAY_SIZE(ip6t_ops));
- ip6t_unregister_table(packet_raw);
+ unregister_pernet_subsys(&ip6table_raw_net_ops);
 }

 module_init(ip6table_raw_init);
```

Subject: Re: [PATCH 2/5] netns netfilter: per-netns IPv6 FILTER, MANGLE, RAW
Posted by Patrick McHardy on Thu, 24 Jan 2008 17:40:41 GMT

View Forum Message <> Reply to Message

Alexey Dobriyan wrote:
> Now it's possible to list and manipulate per-netns ip6tables rules.
> Filtering decisions are based on init_net's table so far.
>
> P.S.: remove init_net check in inet6_create() to see the effect

OK, this patch fixes all but one checkpatch warning again.
Please try to make each patch checkpatch-clean next time
since that causes unnecessary work. Thanks (and applied).