Subject: [PATCH 5/9] Consolidate xxx_the secret_rebuild
Posted by Pavel Emelianov on Fri, 12 Oct 2007 13:16:25 GMT
View Forum Message <> Reply to Message

This code works with the generic data types as well, so
move this into inet_fragment.c

This move makes it possible to hide the secret_timer
management and the secret_rebuild routine completely in
the inet_fragment.c

Introduce the ->hashfn() callback in inet_frags() to get
the hashfun for a given inet_frag_queue() object.

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

```
diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index 9902363..e374412 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -36,6 +36,8 @@ struct inet_frags {
  atomic_t  mem;
  struct timer_list secret_timer;
  struct inet_frags_ctl *ctl;
+
+ unsigned int  (*hashfn)(struct inet_frag_queue *);
 };

 void inet_frags_init(struct inet_frags *);
diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index 534eaa8..ec10e05 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -16,9 +16,38 @@
 #include <linux/module.h>
 #include <linux/timer.h>
 #include <linux/mm.h>
+#include <linux/random.h>

 #include <net/inet_frag.h>

+static void inet_frag_secret_rebuild(unsigned long dummy)
+{
+ struct inet_frags *f = (struct inet_frags *)dummy;
+ unsigned long now = jiffies;
+ int i;
```

```
+
+ write_lock(&f->lock);
+ get_random_bytes(&f->rnd, sizeof(u32));
+ for (i = 0; i < INETFRAGS_HASHSZ; i++) {
+  struct inet_frag_queue *q;
+  struct hlist_node *p, *n;
+
+  hlist_for_each_entry_safe(q, p, n, &f->hash[i], list) {
+   unsigned int hval = f->hashfn(q);
+
+   if (hval != i) {
+    hlist_del(&q->list);
+
+    /* Relink to new hash chain. */
+    hlist_add_head(&q->list, &f->hash[hval]);
+   }
+  }
+ }
+ write_unlock(&f->lock);
+
+ mod_timer(&f->secret_timer, now + f->ctl->secret_interval);
+}
+
 void inet_frags_init(struct inet_frags *f)
 {
  int i;
@@ -35,11 +64,17 @@ void inet_frags_init(struct inet_frags *f)
  f->nqueues = 0;
  atomic_set(&f->mem, 0);

+ init_timer(&f->secret_timer);
+ f->secret_timer.function = inet_frag_secret_rebuild;
+ f->secret_timer.data = (unsigned long)f;
+ f->secret_timer.expires = jiffies + f->ctl->secret_interval;
+ add_timer(&f->secret_timer);
 }
 EXPORT_SYMBOL(inet_frags_init);

 void inet_frags_fini(struct inet_frags *f)
 {
+ del_timer(&f->secret_timer);
 }
 EXPORT_SYMBOL(inet_frags_fini);

diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 5b376c4..7aee137 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
```

```
@@ -112,32 +112,12 @@ static unsigned int ipqhashfn(__be16 id, __be32 saddr, __be32 daddr,
u8 prot)
        ip4_frags.rnd) & (INETFRAGS_HASHSZ - 1);
 }

-static void ipfrag_secret_rebuild(unsigned long dummy)
+static unsigned int ip4_hashfn(struct inet_frag_queue *q)
 {
- unsigned long now = jiffies;
- int i;
+ struct ipq *ipq;

- write_lock(&ip4_frags.lock);
- get_random_bytes(&ip4_frags.rnd, sizeof(u32));
- for (i = 0; i < INETFRAGS_HASHSZ; i++) {
-  struct ipq *q;
-  struct hlist_node *p, *n;
-
-  hlist_for_each_entry_safe(q, p, n, &ip4_frags.hash[i], q.list) {
-   unsigned int hval = ipqhashfn(q->id, q->saddr,
-           q->daddr, q->protocol);
-
-   if (hval != i) {
-    hlist_del(&q->q.list);
-
-    /* Relink to new hash chain. */
-    hlist_add_head(&q->q.list, &ip4_frags.hash[hval]);
-   }
-  }
- }
- write_unlock(&ip4_frags.lock);
-
- mod_timer(&ip4_frags.secret_timer, now + ip4_frags_ctl.secret_interval);
+ ipq = container_of(q, struct ipq, q);
+ return ipqhashfn(ipq->id, ipq->saddr, ipq->daddr, ipq->protocol);
 }

 /* Memory Tracking Functions. */
@@ -705,12 +685,8 @@ struct sk_buff *ip_defrag(struct sk_buff *skb, u32 user)

 void __init ipfrag_init(void)
 {
- init_timer(&ip4_frags.secret_timer);
- ip4_frags.secret_timer.function = ipfrag_secret_rebuild;
- ip4_frags.secret_timer.expires = jiffies + ip4_frags_ctl.secret_interval;
- add_timer(&ip4_frags.secret_timer);
-
  ip4_frags.ctl = &ip4_frags_ctl;
```

```
+ ip4_frags.hashfn = ip4_hashfn;
  inet_frags_init(&ip4_frags);
 }

diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index 2ebe515..a3aef38 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -106,32 +106,12 @@ static unsigned int ip6qhashfn(__be32 id, struct in6_addr *saddr,
  return c & (INETFRAGS_HASHSZ - 1);
 }

-static void nf_ct_frag6_secret_rebuild(unsigned long dummy)
+static unsigned int nf_hashfn(struct inet_frag_queue *q)
 {
- unsigned long now = jiffies;
- int i;
+ struct nf_ct_frag6_queue *nq;

- write_lock(&nf_frags.lock);
- get_random_bytes(&nf_frags.rnd, sizeof(u32));
- for (i = 0; i < INETFRAGS_HASHSZ; i++) {
-  struct nf_ct_frag6_queue *q;
-  struct hlist_node *p, *n;
-
-  hlist_for_each_entry_safe(q, p, n, &nf_frags.hash[i], q.list) {
-   unsigned int hval = ip6qhashfn(q->id,
-             &q->saddr,
-             &q->daddr);
-   if (hval != i) {
-    hlist_del(&q->q.list);
-    /* Relink to new hash chain. */
-    hlist_add_head(&q->q.list,
-            &nf_frags.hash[hval]);
-   }
-  }
- }
- write_unlock(&nf_frags.lock);
-
- mod_timer(&nf_frags.secret_timer, now + nf_frags_ctl.secret_interval);
+ nq = container_of(q, struct nf_ct_frag6_queue, q);
+ return ip6qhashfn(nq->id, &nq->saddr, &nq->daddr);
 }

 /* Memory Tracking Functions. */
@@ -817,11 +797,8 @@ int nf_ct_frag6_kfree_frags(struct sk_buff *skb)

 int nf_ct_frag6_init(void)
```

```
 {
- setup_timer(&nf_frags.secret_timer, nf_ct_frag6_secret_rebuild, 0);
- nf_frags.secret_timer.expires = jiffies + nf_frags_ctl.secret_interval;
- add_timer(&nf_frags.secret_timer);
-
  nf_frags.ctl = &nf_frags_ctl;
+ nf_frags.hashfn = nf_hashfn;
  inet_frags_init(&nf_frags);

  return 0;
@@ -831,7 +808,6 @@ void nf_ct_frag6_cleanup(void)
 {
  inet_frags_fini(&nf_frags);

- del_timer(&nf_frags.secret_timer);
  nf_frags_ctl.low_thresh = 0;
  nf_ct_frag6_evictor();
 }
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 57e32f4..7a357fd 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -131,35 +131,12 @@ static unsigned int ip6qhashfn(__be32 id, struct in6_addr *saddr,
  return c & (INETFRAGS_HASHSZ - 1);
 }

-static void ip6_frag_secret_rebuild(unsigned long dummy)
+static unsigned int ip6_hashfn(struct inet_frag_queue *q)
 {
- unsigned long now = jiffies;
- int i;
-
- write_lock(&ip6_frags.lock);
- get_random_bytes(&ip6_frags.rnd, sizeof(u32));
- for (i = 0; i < INETFRAGS_HASHSZ; i++) {
-  struct frag_queue *q;
-  struct hlist_node *p, *n;
-
-  hlist_for_each_entry_safe(q, p, n, &ip6_frags.hash[i], q.list) {
-   unsigned int hval = ip6qhashfn(q->id,
-             &q->saddr,
-             &q->daddr);
-
-   if (hval != i) {
-    hlist_del(&q->q.list);
-
-    /* Relink to new hash chain. */
-    hlist_add_head(&q->q.list,
```

```
-           &ip6_frags.hash[hval]);
-
-   }
-  }
- }
- write_unlock(&ip6_frags.lock);
+ struct frag_queue *fq;

- mod_timer(&ip6_frags.secret_timer, now + ip6_frags_ctl.secret_interval);
+ fq = container_of(q, struct frag_queue, q);
+ return ip6qhashfn(fq->id, &fq->saddr, &fq->daddr);
 }

 /* Memory Tracking Functions. */
@@ -742,11 +719,7 @@ void __init ipv6_frag_init(void)
  if (inet6_add_protocol(&frag_protocol, IPPROTO_FRAGMENT) < 0)
   printk(KERN_ERR "ipv6_frag_init: Could not register protocol\n");

- init_timer(&ip6_frags.secret_timer);
- ip6_frags.secret_timer.function = ip6_frag_secret_rebuild;
- ip6_frags.secret_timer.expires = jiffies + ip6_frags_ctl.secret_interval;
- add_timer(&ip6_frags.secret_timer);
-
  ip6_frags.ctl = &ip6_frags_ctl;
+ ip6_frags.hashfn = ip6_hashfn;
  inet_frags_init(&ip6_frags);
 }
--
1.5.3.4
```