Subject: [PATCH 5/7] Consolidate xxx_find() in fragment management
Posted by Pavel Emelianov on Tue, 16 Oct 2007 14:03:37 GMT
View Forum Message <> Reply to Message

Here we need another callback ->match to check whether the
entry found in hash matches the key passed. The key used
is the same as the creation argument for inet_frag_create.

Yet again, this ->match is the same for netfilter and ipv6.
Running a frew steps forward - this callback will later
replace the ->equal one.

Since the inet_frag_find() uses the already consolidated
inet_frag_create() remove the xxx_frag_create from protocol
codes.

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

```
diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
index e33072b..6429926 100644
--- a/include/net/inet_frag.h
+++ b/include/net/inet_frag.h
@@ -45,6 +45,8 @@ struct inet_frags {
  void   (*skb_free)(struct sk_buff *);
  int    (*equal)(struct inet_frag_queue *q1,
        struct inet_frag_queue *q2);
+ int   (*match)(struct inet_frag_queue *q,
+       void *arg);
  void   (*frag_expire)(unsigned long data);
 };

@@ -55,8 +57,8 @@ void inet_frag_kill(struct inet_frag_queue *q, struct inet_frags *f);
 void inet_frag_destroy(struct inet_frag_queue *q,
     struct inet_frags *f, int *work);
 int inet_frag_evictor(struct inet_frags *f);
-struct inet_frag_queue *inet_frag_create(struct inet_frags *f,
-  void *create_arg, unsigned int hash);
+struct inet_frag_queue *inet_frag_find(struct inet_frags *f, void *key,
+  unsigned int hash);

 static inline void inet_frag_put(struct inet_frag_queue *q, struct inet_frags *f)
 {
diff --git a/include/net/ipv6.h b/include/net/ipv6.h
index 9dc99bf..005853a 100644
--- a/include/net/ipv6.h
+++ b/include/net/ipv6.h
```

```
@@ -387,6 +387,7 @@ struct ip6_create_arg {
 };

 void ip6_frag_init(struct inet_frag_queue *q, void *a);
+int ip6_frag_match(struct inet_frag_queue *q, void *a);

 static inline int ipv6_addr_any(const struct in6_addr *a)
 {
diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
index 0124885..08901b4 100644
--- a/net/ipv4/inet_fragment.c
+++ b/net/ipv4/inet_fragment.c
@@ -226,8 +226,8 @@ static struct inet_frag_queue *inet_frag_alloc(struct inet_frags *f, void
*arg)
  return q;
 }

-struct inet_frag_queue *inet_frag_create(struct inet_frags *f, void *arg,
-  unsigned int hash)
+static struct inet_frag_queue *inet_frag_create(struct inet_frags *f,
+  void *arg, unsigned int hash)
 {
  struct inet_frag_queue *q;

@@ -237,4 +237,23 @@ struct inet_frag_queue *inet_frag_create(struct inet_frags *f, void *arg,

  return inet_frag_intern(q, f, hash);
 }
-EXPORT_SYMBOL(inet_frag_create);
+
+struct inet_frag_queue *inet_frag_find(struct inet_frags *f, void *key,
+  unsigned int hash)
+{
+ struct inet_frag_queue *q;
+ struct hlist_node *n;
+
+ read_lock(&f->lock);
+ hlist_for_each_entry(q, n, &f->hash[hash], list) {
+  if (f->match(q, key)) {
+   atomic_inc(&q->refcnt);
+   read_unlock(&f->lock);
+   return q;
+  }
+ }
+ read_unlock(&f->lock);
+
+ return inet_frag_create(f, key, hash);
+}
```

```
+EXPORT_SYMBOL(inet_frag_find);
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index db29c3c..46f8de6 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -142,6 +142,19 @@ static int ip4_frag_equal(struct inet_frag_queue *q1,
    qp1->user == qp2->user);
 }

+static int ip4_frag_match(struct inet_frag_queue *q, void *a)
+{
+ struct ipq *qp;
+ struct ip4_create_arg *arg = a;
+
+ qp = container_of(q, struct ipq, q);
+ return (qp->id == arg->iph->id &&
+   qp->saddr == arg->iph->saddr &&
+   qp->daddr == arg->iph->daddr &&
+   qp->protocol == arg->iph->protocol &&
+   qp->user == arg->user);
+}
+
 /* Memory Tracking Functions. */
 static __inline__ void frag_kfree_skb(struct sk_buff *skb, int *work)
 {
@@ -235,18 +248,20 @@ out:
  ipq_put(qp);
 }

-/* Creation primitives. */
-
-/* Add an entry to the 'ipq' queue for a newly received IP datagram. */
-static struct ipq *ip_frag_create(struct iphdr *iph, u32 user, unsigned int h)
+/* Find the correct entry in the "incomplete datagrams" queue for
+ * this IP datagram, and create new one, if nothing is found.
+ */
+static inline struct ipq *ip_find(struct iphdr *iph, u32 user)
 {
  struct inet_frag_queue *q;
  struct ip4_create_arg arg;
+ unsigned int hash;

  arg.iph = iph;
  arg.user = user;
+ hash = ipqhashfn(iph->id, iph->saddr, iph->daddr, iph->protocol);

- q = inet_frag_create(&ip4_frags, &arg, h);
+ q = inet_frag_find(&ip4_frags, &arg, hash);
```

```
  if (q == NULL)
   goto out_nomem;

@@ -257,37 +272,6 @@ out_nomem:
  return NULL;
 }

-/* Find the correct entry in the "incomplete datagrams" queue for
- * this IP datagram, and create new one, if nothing is found.
- */
-static inline struct ipq *ip_find(struct iphdr *iph, u32 user)
-{
- __be16 id = iph->id;
- __be32 saddr = iph->saddr;
- __be32 daddr = iph->daddr;
- __u8 protocol = iph->protocol;
- unsigned int hash;
- struct ipq *qp;
- struct hlist_node *n;
-
- read_lock(&ip4_frags.lock);
- hash = ipqhashfn(id, saddr, daddr, protocol);
- hlist_for_each_entry(qp, n, &ip4_frags.hash[hash], q.list) {
-  if (qp->id == id  &&
-      qp->saddr == saddr &&
-      qp->daddr == daddr &&
-      qp->protocol == protocol &&
-      qp->user == user) {
-   atomic_inc(&qp->q.refcnt);
-   read_unlock(&ip4_frags.lock);
-   return qp;
-  }
- }
- read_unlock(&ip4_frags.lock);
-
- return ip_frag_create(iph, user, hash);
-}
-
 /* Is the fragment too far ahead to be part of ipq? */
 static inline int ip_frag_too_far(struct ipq *qp)
 {
@@ -648,6 +632,7 @@ void __init ipfrag_init(void)
  ip4_frags.skb_free = NULL;
  ip4_frags.qsize = sizeof(struct ipq);
  ip4_frags.equal = ip4_frag_equal;
+ ip4_frags.match = ip4_frag_match;
  ip4_frags.frag_expire = ip_expire;
  inet_frags_init(&ip4_frags);
```

```
 }
diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index 72451e2..1ab52ef 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -176,18 +176,19 @@ out:

 /* Creation primitives. */

-static struct nf_ct_frag6_queue *
-nf_ct_frag6_create(unsigned int hash, __be32 id, struct in6_addr *src,
-  struct in6_addr *dst)
+static __inline__ struct nf_ct_frag6_queue *
+fq_find(__be32 id, struct in6_addr *src, struct in6_addr *dst)
 {
  struct inet_frag_queue *q;
  struct ip6_create_arg arg;
+ unsigned int hash;

  arg.id = id;
  arg.src = src;
  arg.dst = dst;
+ hash = ip6qhashfn(id, src, dst);

- q = inet_frag_create(&nf_frags, &arg, hash);
+ q = inet_frag_find(&nf_frags, &arg, hash);
  if (q == NULL)
   goto oom;

@@ -198,28 +199,6 @@ oom:
  return NULL;
 }

-static __inline__ struct nf_ct_frag6_queue *
-fq_find(__be32 id, struct in6_addr *src, struct in6_addr *dst)
-{
- struct nf_ct_frag6_queue *fq;
- struct hlist_node *n;
- unsigned int hash = ip6qhashfn(id, src, dst);
-
- read_lock(&nf_frags.lock);
- hlist_for_each_entry(fq, n, &nf_frags.hash[hash], q.list) {
-  if (fq->id == id &&
-      ipv6_addr_equal(src, &fq->saddr) &&
-      ipv6_addr_equal(dst, &fq->daddr)) {
-   atomic_inc(&fq->q.refcnt);
-   read_unlock(&nf_frags.lock);
-   return fq;
```

```
-  }
- }
- read_unlock(&nf_frags.lock);
-
- return nf_ct_frag6_create(hash, id, src, dst);
-}
-

 static int nf_ct_frag6_queue(struct nf_ct_frag6_queue *fq, struct sk_buff *skb,
         struct frag_hdr *fhdr, int nhoff)
@@ -706,6 +685,7 @@ int nf_ct_frag6_init(void)
  nf_frags.destructor = nf_frag_free;
  nf_frags.skb_free = nf_skb_free;
  nf_frags.qsize = sizeof(struct nf_ct_frag6_queue);
+ nf_frags.match = ip6_frag_match;
  nf_frags.equal = ip6_frag_equal;
  nf_frags.frag_expire = nf_ct_frag6_expire;
  inet_frags_init(&nf_frags);
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index ce87340..11fffe7 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -155,6 +155,18 @@ int ip6_frag_equal(struct inet_frag_queue *q1, struct inet_frag_queue
*q2)
 }
 EXPORT_SYMBOL(ip6_frag_equal);

+int ip6_frag_match(struct inet_frag_queue *q, void *a)
+{
+ struct frag_queue *fq;
+ struct ip6_create_arg *arg = a;
+
+ fq = container_of(q, struct frag_queue, q);
+ return (fq->id == arg->id &&
+   ipv6_addr_equal(&fq->saddr, arg->src) &&
+   ipv6_addr_equal(&fq->daddr, arg->dst));
+}
+EXPORT_SYMBOL(ip6_frag_match);
+
 /* Memory Tracking Functions. */
 static inline void frag_kfree_skb(struct sk_buff *skb, int *work)
 {
@@ -245,20 +257,20 @@ out:
  fq_put(fq);
 }

-/* Creation primitives. */
-
```

```
-static struct frag_queue *
-ip6_frag_create(__be32 id, struct in6_addr *src, struct in6_addr *dst,
-  struct inet6_dev *idev, unsigned int hash)
+static __inline__ struct frag_queue *
+fq_find(__be32 id, struct in6_addr *src, struct in6_addr *dst,
+ struct inet6_dev *idev)
 {
  struct inet_frag_queue *q;
  struct ip6_create_arg arg;
+ unsigned int hash;

  arg.id = id;
  arg.src = src;
  arg.dst = dst;
+ hash = ip6qhashfn(id, src, dst);

- q = inet_frag_create(&ip6_frags, &arg, hash);
+ q = inet_frag_find(&ip6_frags, &arg, hash);
  if (q == NULL)
   goto oom;

@@ -269,31 +281,6 @@ oom:
  return NULL;
 }

-static __inline__ struct frag_queue *
-fq_find(__be32 id, struct in6_addr *src, struct in6_addr *dst,
- struct inet6_dev *idev)
-{
- struct frag_queue *fq;
- struct hlist_node *n;
- unsigned int hash;
-
- read_lock(&ip6_frags.lock);
- hash = ip6qhashfn(id, src, dst);
- hlist_for_each_entry(fq, n, &ip6_frags.hash[hash], q.list) {
-  if (fq->id == id &&
-      ipv6_addr_equal(src, &fq->saddr) &&
-      ipv6_addr_equal(dst, &fq->daddr)) {
-   atomic_inc(&fq->q.refcnt);
-   read_unlock(&ip6_frags.lock);
-   return fq;
-  }
- }
- read_unlock(&ip6_frags.lock);
-
- return ip6_frag_create(id, src, dst, idev, hash);
-}
```

```
-
-
 static int ip6_frag_queue(struct frag_queue *fq, struct sk_buff *skb,
       struct frag_hdr *fhdr, int nhoff)
 {
@@ -673,6 +660,7 @@ void __init ipv6_frag_init(void)
  ip6_frags.destructor = ip6_frag_free;
  ip6_frags.skb_free = NULL;
  ip6_frags.qsize = sizeof(struct frag_queue);
+ ip6_frags.match = ip6_frag_match;
  ip6_frags.equal = ip6_frag_equal;
  ip6_frags.frag_expire = ip6_frag_expire;
  inet_frags_init(&ip6_frags);
--
1.5.3.4
```