Subject: [PATCH 2.6.21-rc6] [netfilter] early_drop imrovement
Posted by vaverin on Fri, 06 Apr 2007 08:00:29 GMT
View Forum Message <> Reply to Message

When the number of conntracks is reached ip_conntrack_max limit, early_drop() is called and tries to free one of already used conntracks in one of the hash buckets. If it does not find any conntracks that may be freed, it leads to transmission errors.
However it is not fair because of current hash bucket may be empty but the neighbour ones can have the number of conntracks that can be freed. With the following patch early_drop() will search conntracks in all hash buckets.

Signed-off-by: Vasily Averin <vvs@sw.ru>

```
--- 2.6.21-rc6/net/ipv4/netfilter/ip_conntrack_core.c.erdrp
+++ 2.6.21-rc6/net/ipv4/netfilter/ip_conntrack_core.c
@@ -517,7 +517,7 @@ ip_conntrack_tuple_taken(const struct ip

 /* There's a small race here where we may free a just-assured
    connection.  Too bad: we're in trouble anyway. */
-static int early_drop(struct list_head *chain)
+static int __early_drop(struct list_head *chain)
 {
  /* Traverse backwards: gives us oldest, which is roughly LRU */
  struct ip_conntrack_tuple_hash *h;
@@ -547,6 +547,20 @@ static int early_drop(struct list_head *
  return dropped;
 }

+static int early_drop(const struct ip_conntrack_tuple *orig)
+{
+ unsigned int i, hash;
+ int ret = 0;
+
+ hash = hash_conntrack(orig);
+
+ for (i = 0;
+    !ret && i < ip_conntrack_htable_size;
+    ++i, hash = ++hash % ip_conntrack_htable_size)
+  ret = __early_drop(&ip_conntrack_hash[hash]);
+ return ret;
+}
+
 static struct ip_conntrack_helper *
 __ip_conntrack_helper_find( const struct ip_conntrack_tuple *tuple)
 {
@@ -631,9 +645,7 @@ struct ip_conntrack *ip_conntrack_alloc(
```

```
  if (ip_conntrack_max
      && atomic_read(&ip_conntrack_count) > ip_conntrack_max) {
-  unsigned int hash = hash_conntrack(orig);
-  /* Try dropping from this hash chain. */
-  if (!early_drop(&ip_conntrack_hash[hash])) {
+  if (!early_drop(orig)) {
     atomic_dec(&ip_conntrack_count);
     if (net_ratelimit())
      printk(KERN_WARNING
--- 2.6.21-rc6/net/netfilter/nf_conntrack_core.c.erdrp
+++ 2.6.21-rc6/net/netfilter/nf_conntrack_core.c
@@ -542,7 +542,7 @@ EXPORT_SYMBOL_GPL(nf_conntrack_tuple_tak

 /* There's a small race here where we may free a just-assured
    connection.  Too bad: we're in trouble anyway. */
-static int early_drop(struct list_head *chain)
+static int __early_drop(struct list_head *chain)
 {
  /* Traverse backwards: gives us oldest, which is roughly LRU */
  struct nf_conntrack_tuple_hash *h;
@@ -572,6 +572,20 @@ static int early_drop(struct list_head *
  return dropped;
 }

+static int early_drop(const struct nf_conntrack_tuple *orig)
+{
+ unsigned int i, hash;
+ int ret = 0;
+
+ hash = hash_conntrack(orig);
+
+ for (i = 0;
+    !ret && i < nf_conntrack_htable_size;
+    ++i, hash = ++hash % nf_conntrack_htable_size)
+  ret = __early_drop(&nf_conntrack_hash[hash]);
+ return ret;
+}
+
 static struct nf_conn *
 __nf_conntrack_alloc(const struct nf_conntrack_tuple *orig,
        const struct nf_conntrack_tuple *repl,
@@ -591,9 +605,7 @@ __nf_conntrack_alloc(const struct nf_con

  if (nf_conntrack_max
      && atomic_read(&nf_conntrack_count) > nf_conntrack_max) {
-  unsigned int hash = hash_conntrack(orig);
-  /* Try dropping from this hash chain. */
-  if (!early_drop(&nf_conntrack_hash[hash])) {
```

```
+  if (!early_drop(orig)) {
    atomic_dec(&nf_conntrack_count);
    if (net_ratelimit())
     printk(KERN_WARNING
```