Subject: [NETFILTER] early_drop() imrovement (v4)
Posted by vaverin on Wed, 27 Jun 2007 08:46:08 GMT
View Forum Message <> Reply to Message

When the number of conntracks is reached nf_conntrack_max limit, early_drop()
tries to free one of already used conntracks. If it does not find any conntracks
that may be freed, it leads to transmission errors.
In current implementation the conntracks are searched in one hash bucket only.
It have some drawbacks: if used hash bucket is empty we have not any chances to
find something. On the other hand the hash bucket can contain a huge number of
conntracks and its check can last a long time.
The proposed patch limits the number of checked conntracks and allows to search
conntracks in other hash buckets. As result in any case the search will have the
same chances to free one of the conntracks and the check will not lead to long
delays.

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

```
diff --git a/net/netfilter/nf_conntrack_core.c b/net/netfilter/nf_conntrack_core.c
index 7a15e30..0540a88 100644
--- a/net/netfilter/nf_conntrack_core.c
+++ b/net/netfilter/nf_conntrack_core.c
@@ -526,7 +526,7 @@ EXPORT_SYMBOL_GPL(nf_conntrack_tuple_taken);

 /* 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, unsigned int *cnt)
 {
  /* Traverse backwards: gives us oldest, which is roughly LRU */
  struct nf_conntrack_tuple_hash *h;
@@ -541,6 +541,8 @@ static int early_drop(struct list_head *chain)
    atomic_inc(&ct->ct_general.use);
    break;
   }
+  if (!--(*cnt))
+   break;
  }
  read_unlock_bh(&nf_conntrack_lock);

@@ -556,6 +558,25 @@ static int early_drop(struct list_head *chain)
  return dropped;
 }

+#define NF_CT_EVICTION_RANGE 8U
+
+static int early_drop(const struct nf_conntrack_tuple *orig)
+{
```

```
+ unsigned int i, hash, cnt;
+ int ret = 0;
+
+ hash = hash_conntrack(orig);
+ cnt = NF_CT_EVICTION_RANGE;
+
+ for (i = 0; i < nf_conntrack_htable_size; i++) {
+  ret = __early_drop(&nf_conntrack_hash[hash], &cnt);
+  if (ret || !cnt)
+   break;
+  hash++; hash %= nf_conntrack_htable_size;
+ }
+ return ret;
+}
+
 static struct nf_conn *
 __nf_conntrack_alloc(const struct nf_conntrack_tuple *orig,
        const struct nf_conntrack_tuple *repl,
@@ -575,9 +596,7 @@ __nf_conntrack_alloc(const struct nf_conntrack_tuple *orig,

  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
```