Subject: Re: [NETFILTER] early_drop() imrovement (v4)
Posted by Patrick McHardy on Wed, 27 Jun 2007 13:18:07 GMT
View Forum Message <> Reply to Message

Vasily Averin wrote:
> Patrick McHardy wrote:
>
> it is incorrect again: when cnt=0 you should break both cycles.

Indeed, thanks. Fixed now. Also changed it to leave the loop
if we found an entry within a chain (we want the last one of
the chain, so we still walk it entirely) and replaced

hash = (hash + 1) % nf_conntrack_htable_size

by

hash = (hash + 1) & (nf_conntrack_htable_size - 1)

since one of my queued patches makes sure that its always
a power of two.

[NETFILTER]: nf_conntrack: early_drop improvement

When the maximum number of conntrack entries is reached and a new
one needs to be allocated, conntrack tries to drop an unassured
connection from the same hash bucket the new conntrack would hash
to. Since with a properly sized hash the average number of entries
per bucket is 1, the chances of actually finding one are not very
good. This patch increases those chances by walking over the hash
until 8 entries are checked.

Based on patch by Vasily Averin <vvs@sw.ru>.

Signed-off-by: Patrick McHardy <kaber@trash.net>

---
commit 047d8f088a71f30d3042cc7615a7a25aa60a668b
tree c83c4a356caf608fe21a646c24678bef74023270
parent 665d98d03473cab252830129f414e1b38fb2b038
author Patrick McHardy <kaber@trash.net> Wed, 27 Jun 2007 15:17:17 +0200
committer Patrick McHardy <kaber@trash.net> Wed, 27 Jun 2007 15:17:17 +0200

 net/netfilter/nf_conntrack_core.c | 26 ++++++++++++++++++++--------
 1 files changed, 18 insertions(+), 8 deletions(-)

diff --git a/net/netfilter/nf_conntrack_core.c b/net/netfilter/nf_conntrack_core.c
index d7e62ad..2d5c10f 100644

```
--- a/net/netfilter/nf_conntrack_core.c
+++ b/net/netfilter/nf_conntrack_core.c
@@ -377,22 +377,33 @@ nf_conntrack_tuple_taken(const struct nf_conntrack_tuple *tuple,
 }
 EXPORT_SYMBOL_GPL(nf_conntrack_tuple_taken);

+#define NF_CT_EVICTION_RANGE 8
+
 /* 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 hlist_head *chain)
+static int early_drop(unsigned int hash)
 {
  /* Use oldest entry, which is roughly LRU */
  struct nf_conntrack_tuple_hash *h;
  struct nf_conn *ct = NULL, *tmp;
  struct hlist_node *n;
- int dropped = 0;
+ unsigned int i;
+ int dropped = 0, cnt = NF_CT_EVICTION_RANGE;

  read_lock_bh(&nf_conntrack_lock);
- hlist_for_each_entry(h, n, chain, hnode) {
-  tmp = nf_ct_tuplehash_to_ctrack(h);
-  if (!test_bit(IPS_ASSURED_BIT, &tmp->status))
-   ct = tmp;
+ for (i = 0; i < nf_conntrack_htable_size; i++) {
+  hlist_for_each_entry(h, n, &nf_conntrack_hash[hash], hnode) {
+   tmp = nf_ct_tuplehash_to_ctrack(h);
+   if (!test_bit(IPS_ASSURED_BIT, &tmp->status))
+    ct = tmp;
+   if (--cnt <= 0)
+    goto stop;
+  }
+  if (ct)
+   break;
+  hash = (hash + 1) & (nf_conntrack_htable_size - 1);
  }
+stop:
  if (ct)
   atomic_inc(&ct->ct_general.use);
  read_unlock_bh(&nf_conntrack_lock);
@@ -425,8 +436,7 @@ struct nf_conn *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(hash)) {
    atomic_dec(&nf_conntrack_count);
    if (net_ratelimit())
     printk(KERN_WARNING
```