
Subject: Re: [NETFILTER] early_drop() improvement (v4)
Posted by [Patrick McHardy](#) on Wed, 27 Jun 2007 13:54:52 GMT
[View Forum Message](#) <> [Reply to Message](#)

Patrick McHardy wrote:

> Patrick McHardy wrote:

>

> Ideally we should do something like this I think (please let it be
> correct :)):

>

> [...]

> So we always walk chains up to the end and NF_CT_EVICTION_RANGE is
> just a minimum. This ensures we will always get the last entry *and*
> we won't scan less entries than currently if someone has a chain
> longer than 8 entries.

>

> What do you think?

I've added this patch now.

[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 makes it walk the hash until a minimum number of 8 entries are checked.

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

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

commit 31889ee2d8f42f84daec97c3b98c47165e358da8

tree 4a1fe840d3056c2e64ab027cc10f92f3843cd710

parent ca0ac66daa3b264702d72282e388f8ba920f9a91

author Patrick McHardy <kaber@trash.net> Wed, 27 Jun 2007 15:54:22 +0200

committer Patrick McHardy <kaber@trash.net> Wed, 27 Jun 2007 15:54:22 +0200

net/netfilter/nf_conntrack_core.c | 24 ++++++-----
1 files changed, 16 insertions(+), 8 deletions(-)

diff --git a/net/netfilter/nf_conntrack_core.c b/net/netfilter/nf_conntrack_core.c
index ed44a09..ef3f747 100644

```

--- a/net/netfilter/nf_conntrack_core.c
+++ b/net/netfilter/nf_conntrack_core.c
@@ -377,21 +377,30 @@ 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 = 0;

    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;
+           cnt++;
+       }
+       if (ct || cnt >= NF_CT_EVICTION_RANGE)
+           break;
+       hash = (hash + 1) % nf_conntrack_htable_size;
    }
    if (ct)
        atomic_inc(&ct->ct_general.use);
@@ -425,8 +434,7 @@ 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
