
Subject: [PATCH net-2.6.25 7/10][NETNS][FRAGS]: Make thresholds work in namespaces.

Posted by [Pavel Emelianov](#) on Tue, 22 Jan 2008 14:05:47 GMT

[View Forum Message](#) <> [Reply to Message](#)

This is the same as with the timeout variable.

Currently, after exceeding the high threshold `_all_` the fragments are evicted, but it will be fixed in later patch.

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

```
---
include/net/inet_frag.h          | 4 +++
net/ipv4/inet_fragment.c        | 2 +-
net/ipv4/ip_fragment.c          | 26 ++++++++-----
net/ipv6/netfilter/nf_conntrack_reasm.c | 12 +++++-----
net/ipv6/reassembly.c           | 15 +++++-----
5 files changed, 29 insertions(+), 30 deletions(-)
```

```
diff --git a/include/net/inet_frag.h b/include/net/inet_frag.h
```

```
index f56e296..de41359 100644
```

```
--- a/include/net/inet_frag.h
```

```
+++ b/include/net/inet_frag.h
```

```
@@ -7,6 +7,8 @@ struct netns_frags {
```

```
    /* sysctls */
    int timeout;
+ int high_thresh;
+ int low_thresh;
};
```

```
struct inet_frag_queue {
@@ -30,8 +32,6 @@ struct inet_frag_queue {
#define INETFRAGS_HASHSZ 64
```

```
struct inet_frags_ctl {
- int high_thresh;
- int low_thresh;
    int secret_interval;
};
```

```
diff --git a/net/ipv4/inet_fragment.c b/net/ipv4/inet_fragment.c
```

```
index 9da9679..5ab399c 100644
```

```
--- a/net/ipv4/inet_fragment.c
```

```
+++ b/net/ipv4/inet_fragment.c
```

```
@@ -153,7 +153,7 @@ int inet_frag_evictor(struct netns_frags *nf, struct inet_frags *f)
```

```

struct inet_frag_queue *q;
int work, evicted = 0;

- work = atomic_read(&nf->mem) - f->ctl->low_thresh;
+ work = atomic_read(&nf->mem) - nf->low_thresh;
  while (work > 0) {
    read_lock(&f->lock);
    if (list_empty(&f->lru_list)) {
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 70d241c..80c2c19 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -75,14 +75,6 @@ struct ipq {
};

static struct inet_frags_ctl ip4_frags_ctl __read_mostly = {
- /*
- * Fragment cache limits. We will commit 256K at one time. Should we
- * cross that limit we will prune down to 192K. This should cope with
- * even the most extreme cases without allowing an attacker to
- * measurably harm machine performance.
- */
- .high_thresh = 256 * 1024,
- .low_thresh = 192 * 1024,
- .secret_interval = 10 * 60 * HZ,
};

@@ -582,7 +574,7 @@ int ip_defrag(struct sk_buff *skb, u32 user)

net = skb->dev->nd_net;
/* Start by cleaning up the memory. */
- if (atomic_read(&net->ipv4.frags.mem) > ip4_frags_ctl.high_thresh)
+ if (atomic_read(&net->ipv4.frags.mem) > net->ipv4.frags.high_thresh)
  ip_evictor(net);

/* Lookup (or create) queue header */
@@ -610,7 +602,7 @@ static struct ctl_table ip4_frags_ctl_table[] = {
{
- .ctl_name = NET_IPV4_IPFRAG_HIGH_THRESH,
- .procname = "ipfrag_high_thresh",
- .data = &ip4_frags_ctl.high_thresh,
+ .data = &init_net.ipv4.frags.high_thresh,
- .maxlen = sizeof(int),
- .mode = 0644,
- .proc_handler = &proc_dointvec
@@ -618,7 +610,7 @@ static struct ctl_table ip4_frags_ctl_table[] = {
{
- .ctl_name = NET_IPV4_IPFRAG_LOW_THRESH,

```

```

.procname = "ipfrag_low_thresh",
- .data = &ip4_frags_ctl.low_thresh,
+ .data = &init_net.ipv4.frags.low_thresh,
.maxlen = sizeof(int),
.mode = 0644,
.proc_handler = &proc_dointvec
@@ -663,8 +655,8 @@ static int ip4_frags_ctl_register(struct net *net)
if (table == NULL)
goto err_alloc;

- table[0].mode &= ~0222;
- table[1].mode &= ~0222;
+ table[0].data = &net->ipv4.frags.high_thresh;
+ table[1].data = &net->ipv4.frags.low_thresh;
table[2].data = &net->ipv4.frags.timeout;
table[3].mode &= ~0222;
table[4].mode &= ~0222;
@@ -706,6 +698,14 @@ static inline void ip4_frags_ctl_unregister(struct net *net)
static int ipv4_frags_init_net(struct net *net)
{
/*
+ * Fragment cache limits. We will commit 256K at one time. Should we
+ * cross that limit we will prune down to 192K. This should cope with
+ * even the most extreme cases without allowing an attacker to
+ * measurably harm machine performance.
+ */
+ net->ipv4.frags.high_thresh = 256 * 1024;
+ net->ipv4.frags.low_thresh = 192 * 1024;
+ /*
* Important NOTE! Fragment queue must be destroyed before MSL expires.
* RFC791 is wrong proposing to prolongate timer each fragment arrival
* by TTL.
diff --git a/net/ipv6/netfilter/nf_conntrack_reasm.c b/net/ipv6/netfilter/nf_conntrack_reasm.c
index 92a311f..c75ac17 100644
--- a/net/ipv6/netfilter/nf_conntrack_reasm.c
+++ b/net/ipv6/netfilter/nf_conntrack_reasm.c
@@ -71,8 +71,6 @@ struct nf_ct_frag6_queue
};

static struct inet_frags_ctl nf_frags_ctl __read_mostly = {
- .high_thresh = 256 * 1024,
- .low_thresh = 192 * 1024,
.secret_interval = 10 * 60 * HZ,
};

@@ -91,7 +89,7 @@ struct ctl_table nf_ct_ipv6_sysctl_table[] = {
{
.ctl_name = NET_NF_CONNTRACK_FRAG6_LOW_THRESH,

```

```

    .procname = "nf_contrack_frag6_low_thresh",
- .data = &nf_frags_ctl.low_thresh,
+ .data = &nf_init_frags.low_thresh,
    .maxlen = sizeof(unsigned int),
    .mode = 0644,
    .proc_handler = &proc_dointvec,
@@ -99,7 +97,7 @@ struct ctl_table nf_ct_ipv6_sysctl_table[] = {
{
    .ctl_name = NET_NF_CONTRACK_FRAG6_HIGH_THRESH,
    .procname = "nf_contrack_frag6_high_thresh",
- .data = &nf_frags_ctl.high_thresh,
+ .data = &nf_init_frags.high_thresh,
    .maxlen = sizeof(unsigned int),
    .mode = 0644,
    .proc_handler = &proc_dointvec,
@@ -632,7 +630,7 @@ struct sk_buff *nf_ct_frag6_gather(struct sk_buff *skb)
    goto ret_orig;
}

- if (atomic_read(&nf_init_frags.mem) > nf_frags_ctl.high_thresh)
+ if (atomic_read(&nf_init_frags.mem) > nf_init_frags.high_thresh)
    nf_ct_frag6_evictor();

    fq = fq_find(fhdr->identification, &hdr->saddr, &hdr->daddr);
@@ -712,6 +710,8 @@ int nf_ct_frag6_init(void)
    nf_frags.match = ip6_frag_match;
    nf_frags.frag_expire = nf_ct_frag6_expire;
    nf_init_frags.timeout = IPV6_FRAG_TIMEOUT;
+ nf_init_frags.high_thresh = 256 * 1024;
+ nf_init_frags.low_thresh = 192 * 1024;
    inet_frags_init_net(&nf_init_frags);
    inet_frags_init(&nf_frags);

@@ -722,6 +722,6 @@ void nf_ct_frag6_cleanup(void)
{
    inet_frags_fini(&nf_frags);

- nf_frags_ctl.low_thresh = 0;
+ nf_init_frags.low_thresh = 0;
    nf_ct_frag6_evictor();
}
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index 9176136..85f3fa3 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -601,8 +601,7 @@ static int ipv6_frag_rcv(struct sk_buff *skb)
}

```

```

net = skb->dev->nd_net;
- if (atomic_read(&net->ipv6.frags.mem) >
-   init_net.ipv6.sysctl.frags.high_thresh)
+ if (atomic_read(&net->ipv6.frags.mem) > net->ipv6.frags.high_thresh)
  ip6_evictor(net, ip6_dst_idev(skb->dst));

if ((fq = fq_find(net, fhdr->identification, &hdr->saddr, &hdr->daddr,
@@ -634,7 +633,7 @@ static struct ctl_table ip6_frags_ctl_table[] = {
{
  .ctl_name = NET_IPV6_IP6FRAG_HIGH_THRESH,
  .procname = "ip6frag_high_thresh",
- .data = &init_net.ipv6.sysctl.frags.high_thresh,
+ .data = &init_net.ipv6.frags.high_thresh,
  .maxlen = sizeof(int),
  .mode = 0644,
  .proc_handler = &proc_dointvec
@@ -642,7 +641,7 @@ static struct ctl_table ip6_frags_ctl_table[] = {
{
  .ctl_name = NET_IPV6_IP6FRAG_LOW_THRESH,
  .procname = "ip6frag_low_thresh",
- .data = &init_net.ipv6.sysctl.frags.low_thresh,
+ .data = &init_net.ipv6.frags.low_thresh,
  .maxlen = sizeof(int),
  .mode = 0644,
  .proc_handler = &proc_dointvec
@@ -679,8 +678,8 @@ static int ip6_frags_sysctl_register(struct net *net)
if (table == NULL)
  goto err_alloc;

- table[0].mode &= ~0222;
- table[1].mode &= ~0222;
+ table[0].data = &net->ipv6.frags.high_thresh;
+ table[1].data = &net->ipv6.frags.low_thresh;
  table[2].data = &net->ipv6.frags.timeout;
  table[3].mode &= ~0222;
}
@@ -722,8 +721,8 @@ static int ipv6_frags_init_net(struct net *net)
{
  ip6_frags.ctl = &net->ipv6.sysctl.frags;

- net->ipv6.sysctl.frags.high_thresh = 256 * 1024;
- net->ipv6.sysctl.frags.low_thresh = 192 * 1024;
+ net->ipv6.frags.high_thresh = 256 * 1024;
+ net->ipv6.frags.low_thresh = 192 * 1024;
  net->ipv6.frags.timeout = IPV6_FRAG_TIMEOUT;
  net->ipv6.sysctl.frags.secret_interval = 10 * 60 * HZ;

--

```

1.5.3.4
