
Subject: [PATCH 3/3][UNIX] The unix_nr_socks limit can be exceeded

Posted by [Pavel Emelianov](#) on Wed, 07 Nov 2007 14:01:17 GMT

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

The unix_nr_socks value is limited with the 2 * get_max_files() value, as seen from the unix_create1(). However, the check and the actual increment are separated with the GFP_KERNEL allocation, so this limit can be exceeded under a memory pressure - task may go to sleep freeing the pages and some other task will be allowed to allocate a new sock and so on and so forth.

So make the increment before the check (similar thing is done in the sock_kmalloc) and go to kmalloc after this.

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

diff --git a/net/unix/af_unix.c b/net/unix/af_unix.c

index ab9048a..e835da8 100644

--- a/net/unix/af_unix.c

+++ b/net/unix/af_unix.c

@@ -599,15 +599,14 @@ static struct sock * unix_create1(struct net *net, struct socket *sock)

struct sock *sk = NULL;

struct unix_sock *u;

- if (atomic_read(&unix_nr_socks) >= 2*get_max_files())

+ atomic_inc(&unix_nr_socks);

+ if (atomic_read(&unix_nr_socks) > 2 * get_max_files())

goto out;

sk = sk_alloc(net, PF_UNIX, GFP_KERNEL, &unix_proto);

if (!sk)

goto out;

- atomic_inc(&unix_nr_socks);

-

sock_init_data(sock, sk);

lockdep_set_class(&sk->sk_receive_queue.lock,

&af_unix_sk_receive_queue_lock_key);

@@ -625,6 +624,8 @@ static struct sock * unix_create1(struct net *net, struct socket *sock)

init_waitqueue_head(&u->peer_wait);

unix_insert_socket(unix_sockets_unbound, sk);

out:

+ if (sk == NULL)

+ atomic_dec(&unix_nr_socks);

return sk;

}

--

1.5.3.4

Subject: Re: [PATCH 3/3][UNIX] The unix_nr_socks limit can be exceeded
Posted by [davem](#) on Sun, 11 Nov 2007 06:08:43 GMT

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

From: Pavel Emelyanov <xemul@openvz.org>

Date: Wed, 07 Nov 2007 17:01:17 +0300

> The unix_nr_socks value is limited with the 2 * get_max_files() value,
> as seen from the unix_create1(). However, the check and the actual
> increment are separated with the GFP_KERNEL allocation, so this limit
> can be exceeded under a memory pressure - task may go to sleep freeing
> the pages and some other task will be allowed to allocate a new sock
> and so on and so forth.
>
> So make the increment before the check (similar thing is done in the
> sock_kmalloc) and go to kmalloc after this.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied, good catch Pavel.
