## Subject: [PATCH 0/4] Fix race between sk\_filter reassign and sk\_clone() Posted by Pavel Emelianov on Wed, 17 Oct 2007 09:45:54 GMT

View Forum Message <> Reply to Message

The race can result in that some sock will get an sk\_filter pointer set to kfree-d memory. Look

that's it - after a quiescent state pass the new\_sk will have a pointer on kfree-d filter.

The same problem exists for detaching filter (SO\_DETACH\_FILTER).

The proposed fix consists of 3 preparation patches and the fix itself.

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

Subject: [PATCH 1/4] Introduce the sk\_detach\_filter() call Posted by Pavel Emelianov on Wed, 17 Oct 2007 09:47:31 GMT View Forum Message <> Reply to Message

Filter is attached in a separate function, so do the same for filter detaching.

This also removes one variable sock\_setsockopt().

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

---

diff --git a/include/linux/filter.h b/include/linux/filter.h index 91b2e3b..ddfa037 100644

```
--- a/include/linux/filter.h
+++ b/include/linux/filter.h
@@ -146,6 +146,7 @@ struct sock;
extern unsigned int sk_run_filter(struct sk_buff *skb, struct sock_filter *filter, int flen);
extern int sk_attach_filter(struct sock_fprog *fprog, struct sock *sk);
+extern int sk detach filter(struct sock *sk);
extern int sk_chk_filter(struct sock_filter *filter, int flen);
#endif /* KERNEL */
diff --git a/net/core/filter.c b/net/core/filter.c
index bd903aa..fd60758 100644
--- a/net/core/filter.c
+++ b/net/core/filter.c
@@ -433,5 +433,21 @@ int sk_attach_filter(struct sock_fprog *fprog, struct sock *sk)
 return err:
}
+int sk detach filter(struct sock *sk)
+{
+ int ret = -ENOENT;
+ struct sk filter *filter;
+ rcu_read_lock_bh();
+ filter = rcu_dereference(sk->sk_filter);
+ if (filter) {
+ rcu_assign_pointer(sk->sk_filter, NULL);
+ sk_filter_release(sk, filter);
+ ret = 0;
+ }
+ rcu_read_unlock_bh();
+ return ret;
+}
EXPORT_SYMBOL(sk_chk_filter);
EXPORT SYMBOL(sk run filter):
diff --git a/net/core/sock.c b/net/core/sock.c
index d45ecdc..0710138 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@ @ -428,7 +428,6 @ @ int sock setsockopt(struct socket *sock, int level, int optname,
    char __user *optval, int optlen)
 struct sock *sk=sock->sk;
struct sk_filter *filter;
 int val;
 int valbool:
 struct linger ling;
```

```
@@ -652,16 +651,7 @@ set_rcvbuf:
 break:
 case SO_DETACH_FILTER:
- rcu_read_lock_bh();
- filter = rcu_dereference(sk->sk_filter);
- if (filter) {
rcu_assign_pointer(sk->sk_filter, NULL);
- sk filter release(sk, filter);
- rcu read unlock bh();
- break;
- }
- rcu_read_unlock_bh();
ret = -ENONET;
+ ret = sk_detach_filter(sk);
 break:
 case SO PASSSEC:
1.5.3.4
```

Subject: [PATCH 2/4] Move the filter releasing into a separate call Posted by Pavel Emelianov on Wed, 17 Oct 2007 09:49:38 GMT View Forum Message <> Reply to Message

This is done merely as a preparation for the fix.

The sk\_filter\_uncharge() unaccounts the filter memory and calls the sk\_filter\_release(), which in turn decrements the refcount and frees the filter.

The latter function will be required separately.

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

```
diff --git a/include/net/sock.h b/include/net/sock.h index 453c79d..b9cfe12 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@ @ -922,14 +922,18 @ @ static inline void sk_filter_rcu_free(struct rcu_head *rcu)
  * Remove a filter from a socket and release its resources.
  */
-static inline void sk_filter_release(struct sock *sk, struct sk_filter *fp)
+static inline void sk_filter_release(struct sk_filter *fp)
```

```
+{
+ if (atomic dec and test(&fp->refcnt))
+ call_rcu_bh(&fp->rcu, sk_filter_rcu_free);
+}
+static inline void sk_filter_uncharge(struct sock *sk, struct sk_filter *fp)
 unsigned int size = sk_filter_len(fp);
 atomic sub(size, &sk->sk omem alloc);
if (atomic dec and test(&fp->refcnt))
call_rcu_bh(&fp->rcu, sk_filter_rcu_free);
+ sk_filter_release(fp);
static inline void sk filter charge(struct sock *sk, struct sk filter *fp)
diff --git a/net/core/filter.c b/net/core/filter.c
index fd60758..2be1830 100644
--- a/net/core/filter.c
+++ b/net/core/filter.c
@@ -429,7 +429,7 @@ int sk attach filter(struct sock fprog *fprog, struct sock *sk)
 }
 if (fp)
- sk_filter_release(sk, fp);
+ sk_filter_uncharge(sk, fp);
 return err;
}
@ @ -442,7 +442,7 @ @ int sk detach filter(struct sock *sk)
 filter = rcu_dereference(sk->sk_filter);
 if (filter) {
 rcu_assign_pointer(sk->sk_filter, NULL);
sk_filter_release(sk, filter);
+ sk_filter_uncharge(sk, filter);
 ret = 0:
 }
 rcu_read_unlock_bh();
diff --git a/net/core/sock.c b/net/core/sock.c
index 0710138..d292b41 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@ @ -915,7 +915,7 @ @ void sk_free(struct sock *sk)
 filter = rcu_dereference(sk->sk_filter);
 if (filter) {
- sk filter release(sk, filter);
```

```
+ sk_filter_uncharge(sk, filter);rcu_assign_pointer(sk->sk_filter, NULL);}--1.5.3.4
```

Subject: [PATCH 3/4] Cleanup the error path in sk\_attach\_filter Posted by Pavel Emelianov on Wed, 17 Oct 2007 09:51:11 GMT View Forum Message <> Reply to Message

The sk\_filter\_uncharge is called for error handling and for releasing the former filter, but this will have to be done in a bit different manner, so cleanup the error path a bit.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org> diff --git a/net/core/filter.c b/net/core/filter.c index 2be1830..54dddc9 100644 --- a/net/core/filter.c +++ b/net/core/filter.c @ @ -398,7 +398,7 @ @ int sk\_chk\_filter(struct sock\_filter \*filter, int flen) \*/ int sk attach filter(struct sock fprog \*fprog, struct sock \*sk) - struct sk filter \*fp; + struct sk filter \*fp, \*old fp; unsigned int fsize = sizeof(struct sock filter) \* fprog->len; int err; @@-418,19 +418,18 @@ int sk attach filter(struct sock fprog \*fprog, struct sock \*sk) fp->len = fprog->len; err = sk\_chk\_filter(fp->insns, fp->len); - if (!err) { struct sk\_filter \*old\_fp; rcu read lock bh(); old fp = rcu dereference(sk->sk filter); - rcu assign pointer(sk->sk filter, fp); rcu\_read\_unlock\_bh(); - fp = old\_fp; + if (err) { + sk\_filter\_uncharge(sk, fp);

```
+ return err;
}

- if (fp)
- sk_filter_uncharge(sk, fp);
- return err;
+ rcu_read_lock_bh();
+ old_fp = rcu_dereference(sk->sk_filter);
+ rcu_assign_pointer(sk->sk_filter, fp);
+ rcu_read_unlock_bh();
+
+ sk_filter_uncharge(sk, old_fp);
+ return 0;
}

int sk_detach_filter(struct sock *sk)
--
1.5.3.4
```

Subject: [PATCH 4/4] Fix the race between sk\_filter\_(de|at)tach and sk\_clone() Posted by Pavel Emelianov on Wed, 17 Oct 2007 09:53:40 GMT

View Forum Message <> Reply to Message

The proposed fix is to delay the reference counter decrement until the quiescent state pass. This will give sk\_clone() a chance to get the reference on the cloned filter.

Regular sk\_filter\_uncharge can happen from the sk\_free() only and there's no need in delaying the put - the socket is dead anyway and is to be release itself.

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

```
diff --git a/include/net/sock.h b/include/net/sock.h index b9cfe12..43fc3fa 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@ @ -905,16 +905,6 @ @ static inline int sk_filter(struct sock *sk, struct sk_buff *skb)
}

/**
- * sk_filter_rcu_free: Free a socket filter
- * @ rcu: rcu_head that contains the sk_filter to free
- */
-static inline void sk_filter_rcu_free(struct rcu_head *rcu)
```

```
-{
- struct sk filter *fp = container of(rcu, struct sk filter, rcu);
- kfree(fp);
-}
-/**
 * sk filter release: Release a socket filter
 * @sk: socket
 * @fp: filter to remove
@ @ -925,7 +915,7 @ @ static inline void sk filter rcu free(struct rcu head *rcu)
static inline void sk_filter_release(struct sk_filter *fp)
 if (atomic_dec_and_test(&fp->refcnt))
call_rcu_bh(&fp->rcu, sk_filter_rcu_free);
+ kfree(fp);
}
static inline void sk_filter_uncharge(struct sock *sk, struct sk_filter *fp)
diff --git a/net/core/filter.c b/net/core/filter.c
index 54dddc9..b8bc7d3 100644
--- a/net/core/filter.c
+++ b/net/core/filter.c
@@ -387,6 +387,25 @@ int sk_chk_filter(struct sock_filter *filter, int flen)
}
+ * sk_filter_rcu_release: Release a socket filter by rcu_head
+ * @rcu: rcu head that contains the sk filter to free
+ */
+static void sk_filter_rcu_release(struct rcu_head *rcu)
+ struct sk_filter *fp = container_of(rcu, struct sk_filter, rcu);
+ sk_filter_release(fp);
+}
+static void sk_filter_delayed_uncharge(struct sock *sk, struct sk_filter *fp)
+{
+ unsigned int size = sk_filter_len(fp);
+ atomic sub(size, &sk->sk omem alloc);
+ call_rcu_bh(&fp->rcu, sk_filter_rcu_release);
+}
+
 * sk_attach_filter - attach a socket filter
 * @fprog: the filter program
 * @sk: the socket to use
```

```
@ @ -428,7 +447,7 @ @ int sk_attach_filter(struct sock_fprog *fprog, struct sock *sk)
 rcu assign pointer(sk->sk filter, fp);
 rcu_read_unlock_bh();
sk_filter_uncharge(sk, old_fp);
+ sk_filter_delayed_uncharge(sk, old_fp);
 return 0:
}
@ @ -441,7 +460,7 @ @ int sk detach filter(struct sock *sk)
 filter = rcu dereference(sk->sk filter);
 if (filter) {
 rcu_assign_pointer(sk->sk_filter, NULL);
sk_filter_uncharge(sk, filter);
+ sk_filter_delayed_uncharge(sk, filter);
 ret = 0:
 rcu_read_unlock_bh();
1.5.3.4
```

Subject: Re: [PATCH 0/4] Fix race between sk\_filter reassign and sk\_clone() Posted by davem on Thu, 18 Oct 2007 04:23:02 GMT View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>

Date: Wed, 17 Oct 2007 13:45:54 +0400

```
> The race can result in that some sock will get an sk filter
> pointer set to kfree-d memory. Look
> CPU1:
                           CPU2:
> sk clone():
                            sk attach filter():
> new sk = sk alloc(...);
> sock_copy(new_sk, sk);
> /* copies the filter ptr */
>
  filter = new_sk->sk_filter;
   if (filter)
>
                          old fp = sk->sk filter;
>
>
                          sk filter release(old fp);
>
                           if (atomic dec and test(&old fp->refcnt))
>
     atomic_inc(&filter->refcnt);
>
                             /* true */
>
                             call_rcu(&fp->rcu, kfree);
>
```

```
> that's it - after a quiescent state pass the new_sk will have
> a pointer on kfree-d filter.
> The same problem exists for detaching filter (SO_DETACH_FILTER).
> The proposed fix consists of 3 preparation patches and the fix itself.
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
Looks good, applied.
Thanks for fixing this bug Pavel!
```

Subject: Re: [PATCH 0/4] Fix race between sk\_filter reassign and sk\_clone() Posted by Olof Johansson on Fri, 19 Oct 2007 02:29:47 GMT View Forum Message <> Reply to Message

```
On Wed, Oct 17, 2007 at 09:23:02PM -0700, David Miller wrote:
[...]
>> The same problem exists for detaching filter (SO_DETACH_FILTER).
>> The proposed fix consists of 3 preparation patches and the fix itself.
> >
> > Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
> Looks good, applied.
> Thanks for fixing this bug Pavel!
Looks like this might be causing problems, at least for me on ppc. This
happened during a normal boot, right around first interface config/dhcp
run..
cpu 0x0: Vector: 300 (Data Access) at [c00000000147b820]
  pc: c00000000435e5c: .sk_filter_delayed_uncharge+0x1c/0x60
  lr: c0000000004360d0: .sk attach filter+0x170/0x180
  sp: c0000000147baa0
 msr: 9000000000009032
 dar: 4
dsisr: 40000000
 current = 0xc000000004780fa0
 paca = 0xc000000000650480
  pid = 1295, comm = dhclient3
0:mon> t
[c0000000147bb20] c000000004360d0 .sk_attach_filter+0x170/0x180
[c0000000147bbd0] c00000000418988 .sock_setsockopt+0x788/0x7f0
[c0000000147bcb0] c00000000438a74 .compat_sys_setsockopt+0x4e4/0x5a0
```

```
[c0000000147bd90] c0000000043955c .compat sys socketcall+0x25c/0x2b0
[c0000000147be30] c00000000007508 syscall exit+0x0/0x40
--- Exception: c01 (System Call) at 000000000ff618d8
SP (fffdf040) is in userspace
0:mon>
I.e. null pointer deref at sk_filter_delayed_uncharge+0x1c:
0:mon> di $.sk filter delayed uncharge
c000000000435e40 7c0802a6
                                 mflr
                                       r0
c000000000435e44 fbc1fff0
                               std
                                     r30,-16(r1)
c000000000435e48 7c8b2378
                                       r11.r4
                                 mr
c000000000435e4c ebc2cdd0
                                 ld
                                      r30,-12848(r2)
c000000000435e50 f8010010
                                 std
                                       r0,16(r1)
c000000000435e54 f821ff81
                               stdu r1,-128(r1)
                                 addi r0,r3,164
c000000000435e58 380300a4
c000000000435e5c 81240004
                                       r9,4(r4)
                                 lwz
That's the deref of fp:
static void sk filter delayed uncharge(struct sock *sk, struct sk filter *fp)
{
    unsigned int size = sk_filter_len(fp);
That is called from sk_attach_filter():
    rcu read lock bh();
    old_fp = rcu_dereference(sk->sk_filter);
    rcu assign pointer(sk->sk filter, fp);
    rcu_read_unlock_bh();
    sk_filter_delayed_uncharge(sk, old_fp);
    return 0:
So, looks like rcu dereference() returned NULL. I don't know the
filter code at all, but it seems like it might be a valid case?
sk detach filter() seems to handle a NULL sk filter, at least.
```

Signed-off-by: Olof Johansson <olof@lixom.net>

problem for me:

So, this needs review by someone who knows the filter, but it fixes the

```
diff --git a/net/core/filter.c b/net/core/filter.c index 1f0068e..e0a0694 100644
--- a/net/core/filter.c
+++ b/net/core/filter.c
@ @ -447,7 +447,8 @ @ int sk_attach_filter(struct sock_fprog *fprog, struct sock *sk) rcu_assign_pointer(sk->sk_filter, fp); rcu_read_unlock_bh();
- sk_filter_delayed_uncharge(sk, old_fp);
+ if (old_fp)
+ sk_filter_delayed_uncharge(sk, old_fp); return 0;
}
```

Subject: Re: [PATCH 0/4] Fix race between sk\_filter reassign and sk\_clone() Posted by davem on Fri, 19 Oct 2007 04:55:21 GMT

View Forum Message <> Reply to Message

From: Olof Johansson <olof@lixom.net>

Date: Thu, 18 Oct 2007 21:29:47 -0500

> So, looks like rcu\_dereference() returned NULL. I don't know the 
> filter code at all, but it seems like it might be a valid case? 
> sk\_detach\_filter() seems to handle a NULL sk\_filter, at least. 
> 
> 
> So, this needs review by someone who knows the filter, but it fixes the 
> problem for me: 
> 
> Signed-off-by: Olof Johansson <olof@lixom.net>

I've applied this for now to my net-2.6 tree, thanks Olof for tracking this down.

Pavel please take a look at this and let me know if it should fixed in some other way.

Thanks!

Subject: Re: [PATCH 0/4] Fix race between sk\_filter reassign and sk\_clone() Posted by Pavel Emelianov on Fri, 19 Oct 2007 07:37:02 GMT

View Forum Message <> Reply to Message

David Miller wrote:

```
> From: Olof Johansson <olof@lixom.net>
> Date: Thu, 18 Oct 2007 21:29:47 -0500
>> So, looks like rcu_dereference() returned NULL. I don't know the
>> filter code at all, but it seems like it might be a valid case?
>> sk_detach_filter() seems to handle a NULL sk_filter, at least.
>>
>>
>> So, this needs review by someone who knows the filter, but it fixes the
>> problem for me:
Yes. The NULL filter is a valid case, when there are no
filters attached at all. So this fix is correct.
Thanks, Olof. Sorry, Dave :(
>> Signed-off-by: Olof Johansson <olof@lixom.net>
Acked-by: Pavel Emelyanov <xemul@openvz.org>
> I've applied this for now to my net-2.6 tree, thanks Olof
> for tracking this down.
> Pavel please take a look at this and let me know if it should
> fixed in some other way.
> Thanks!
>
```

Subject: Re: [PATCH 0/4] Fix race between sk\_filter reassign and sk\_clone() Posted by davem on Fri, 19 Oct 2007 07:52:45 GMT

View Forum Message <> Reply to Message

```
From: Pavel Emelyanov <xemul@openvz.org>
Date: Fri, 19 Oct 2007 11:37:02 +0400

> David Miller wrote:
> > From: Olof Johansson <olof@lixom.net>
> > Date: Thu, 18 Oct 2007 21:29:47 -0500
>>
> > So, looks like rcu_dereference() returned NULL. I don't know the
> >> filter code at all, but it seems like it might be a valid case?
> >> sk_detach_filter() seems to handle a NULL sk_filter, at least.
> >>
> >>
```

>>> So, this needs review by someone who knows the filter, but it fixes the

```
> >> problem for me:
> Yes. The NULL filter is a valid case, when there are no
> filters attached at all. So this fix is correct.
> Thanks, Olof. Sorry, Dave :(
```

No worries, thanks for reviewing.