
Subject: [PATCH] Masquerade sender information
Posted by [Sukadev Bhattiprolu](#) on Sat, 27 Oct 2007 19:02:16 GMT
[View Forum Message](#) <> [Reply to Message](#)

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [PATCH] Masquerade sender information

With multiple pid namespaces, sender of a signal could be in an ancestor namespace of the receiver and so the sender will not have a valid 'pid_t' in the receiver's namespace.

In this case, masquerade the 'siginfo' for the signal to pretend that the signal originated from the kernel.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

include/linux/pid.h | 8 +++++++
include/linux/pid_namespace.h | 1 +
kernel/pid.c | 13 ++++++++
kernel/signal.c | 15 ++++++++
4 files changed, 37 insertions(+)

Index: 2.6.23-mm1/include/linux/pid.h

```
=====
--- 2.6.23-mm1.orig/include/linux/pid.h 2007-10-27 08:48:56.000000000 -0700
+++ 2.6.23-mm1/include/linux/pid.h 2007-10-27 09:56:53.000000000 -0700
@@ -123,6 +123,14 @@ extern struct pid *alloc_pid(struct pid_
extern void FASTCALL(free_pid(struct pid *pid));
extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);

+static inline struct pid_namespace *pid_active_ns(struct pid *pid)
+{
+    if (!pid)
+        return NULL;
+
+    return pid->numbers[pid->level].ns;
+}
+
+/*
+ * the helpers to get the pid's id seen from different namespaces
+ */
```

Index: 2.6.23-mm1/kernel/pid.c

```
=====
--- 2.6.23-mm1.orig/kernel/pid.c 2007-10-27 08:50:51.000000000 -0700
+++ 2.6.23-mm1/kernel/pid.c 2007-10-27 10:03:28.000000000 -0700
@@ -430,6 +430,19 @@ struct pid *find_get_pid(pid_t nr)
    return pid;
}
```

```

+/*
+ * Return TRUE if the active pid namespace of @tsk is same as active
+ * pid namespace of 'current'.
+ *
+ * Note the difference between this and the task_in_pid_ns() below.
+ * task_in_pid_ns() includes processes in descendant pid name spaces
+ * but pid_ns_equal() only matches _active_ pid namespaces.
+ */
+int pid_ns_equal(struct task_struct *tsk)
+{
+ return pid_active_ns(task_pid(current)) == pid_active_ns(task_pid(tsk));
+}
+
+static int pid_in_pid_ns(struct pid *pid, struct pid_namespace *ns)
+{
+ return pid && (ns->level <= pid->level) &&
Index: 2.6.23-mm1/kernel/signal.c

```

```

=====
--- 2.6.23-mm1.orig/kernel/signal.c 2007-10-27 08:50:51.000000000 -0700
+++ 2.6.23-mm1/kernel/signal.c 2007-10-27 10:02:04.000000000 -0700
@@ -679,6 +679,20 @@ static void handle_stop_signal(int sig,
 }
}

```

```

+static void masquerade_sender(struct task_struct *t, struct sigqueue *q)
+{
+ /*
+ * If the sender does not have a pid_t in the receiver's active
+ * pid namespace, set si_pid to 0 and pretend signal originated
+ * from the kernel.
+ */
+ if (!pid_ns_equal(t)) {
+ q->info.si_pid = 0;
+ q->info.si_uid = 0;
+ q->info.si_code = SI_KERNEL;
+ }
+}
+
+static int send_signal(int sig, struct siginfo *info, struct task_struct *t,
+ struct sigpending *signals)
+{
@@ -730,6 +744,7 @@ static int send_signal(int sig, struct s
+ copy_siginfo(&q->info, info);
+ break;
+ }
+ masquerade_sender(t, q);
+ } else if (!is_si_special(info)) {

```

```
if (sig >= SIGRTMIN && info->si_code != SI_USER)
/*
```

Index: 2.6.23-mm1/include/linux/pid_namespace.h

```
=====
--- 2.6.23-mm1.orig/include/linux/pid_namespace.h 2007-10-27 09:44:25.000000000 -0700
+++ 2.6.23-mm1/include/linux/pid_namespace.h 2007-10-27 10:04:20.000000000 -0700
@@ -56,6 +56,7 @@ static inline struct task_struct *task_c
return tsk->nsproxy->pid_ns->child_reaper;
}

+extern int pid_ns_equal(struct task_struct *tsk);
extern int task_in_pid_ns(struct task_struct *tsk, struct pid_namespace *ns);

#endif /* _LINUX_PID_NS_H */
```

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [PATCH] Masquerade sender information
Posted by [ebiederm](#) on Mon, 29 Oct 2007 20:06:20 GMT
[View Forum Message](#) <> [Reply to Message](#)

sukadev@us.ibm.com writes:

```
> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> Subject: [PATCH] Masquerade sender information
>
> With multiple pid namespaces, sender of a signal could be in an ancestor
> namespace of the receiver and so the sender will not have a valid 'pid_t'
> in the receiver's namespace.
>
> In this case, masquerade the 'siginfo' for the signal to pretend that the
> signal originated from the kernel.
```

At first glance this looks ok. I think the only case where we can be sending a signal from inside a pid namespace to something not in a child pid namespace is if we are the kernel. In which case we also want si_pid = 0.

If that holds this problem is easier then I was thinking it would be.

Eric

Subject: Re: [PATCH] Masquerade sender information

Posted by [serue](#) on Thu, 01 Nov 2007 16:50:26 GMT

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

Quoting Eric W. Biederman (ebiederm@xmission.com):

> sukadev@us.ibm.com writes:

>

> > From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

> > Subject: [PATCH] Masquerade sender information

> >

> > With multiple pid namespaces, sender of a signal could be in an ancestor

> > namespace of the receiver and so the sender will not have a valid 'pid_t'

> > in the receiver's namespace.

> >

> > In this case, masquerade the 'signifo' for the signal to pretend that the

> > signal originated from the kernel.

>

> At first glance this looks ok. I think the only case where we can

> be sending a signal from inside a pid namespace to something not

> in a child pid namespace is if we are the kernel. In which case

Are we now blocking F_SETOWN|F_SETSIG signals to outside our pid namespace? mq_notify? (I didn't think we were)

> we also want si_pid = 0.

>

> If that holds this problem is easier then I was thinking it would

> be.

>

> Eric

>

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [PATCH] Masquerade sender information

Posted by [ebiederm](#) on Thu, 01 Nov 2007 16:59:31 GMT

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

sukadev@us.ibm.com writes:

```

> +static void masquerade_sender(struct task_struct *t, struct sigqueue *q)
> +{
> +    /*
> +     * If the sender does not have a pid_t in the receiver's active
> +     * pid namespace, set si_pid to 0 and pretend signal originated
> +     * from the kernel.
> +     */
> +    if (!pid_ns_equal(t)) {
> +        q->info.si_pid = 0;
> +        q->info.si_uid = 0;
> +        q->info.si_code = SI_KERNEL;
> +    }
> +}

```

It looks like we are hooked in the right place. However the way we are handling this appears wrong.

First. If we have an si_code that does not use si_pid then we should not be changing si_pid, because the structure is a union and that field is not always a pid value.

My gut feel says the code should be something like:

```

switch (q->info->si_code & __SI_MASK) {
case __SI_KILL:
case __SI_CHILD:
case __SI_RT:
case __MESQ:
    q->info->si_pid = task_pid_nr_ns(current, t->nsproxy->pid_ns);
    break;
}

```

Eric

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [PATCH] Masquerade sender information
Posted by [Pavel Emelianov](#) on Thu, 01 Nov 2007 17:03:27 GMT
[View Forum Message](#) <> [Reply to Message](#)

sukadev@us.ibm.com wrote:
> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> Subject: [PATCH] Masquerade sender information

>
> With multiple pid namespaces, sender of a signal could be in an ancestor
> namespace of the receiver and so the sender will not have a valid 'pid_t'
> in the receiver's namespace.
>
> In this case, masquerade the 'siginfo' for the signal to pretend that the
> signal originated from the kernel.
>
> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

Hmm... Definitely looks good. I thought that the problem was that
the siginfo structure is not always allocated, but hey!, if we
failed to allocate it, then it doesn't really matter whether we
masquerade something or not! Good point!

```
> ---
> include/linux/pid.h          |  8 ++++++++
> include/linux/pid_namespace.h |  1 +
> kernel/pid.c                 | 13 ++++++++
> kernel/signal.c              | 15 ++++++++
> 4 files changed, 37 insertions(+)
>
> Index: 2.6.23-mm1/include/linux/pid.h
> =====
> --- 2.6.23-mm1.orig/include/linux/pid.h 2007-10-27 08:48:56.000000000 -0700
> +++ 2.6.23-mm1/include/linux/pid.h 2007-10-27 09:56:53.000000000 -0700
> @@ -123,6 +123,14 @@ extern struct pid *alloc_pid(struct pid_
> extern void FASTCALL(free_pid(struct pid *pid));
> extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);
>
> +static inline struct pid_namespace *pid_active_ns(struct pid *pid)
> +{
> +    if (!pid)
> +        return NULL;
> +
> +    return pid->numbers[pid->level].ns;
> +}
> +
> /*
>  * the helpers to get the pid's id seen from different namespaces
>  *
> Index: 2.6.23-mm1/kernel/pid.c
> =====
> --- 2.6.23-mm1.orig/kernel/pid.c 2007-10-27 08:50:51.000000000 -0700
> +++ 2.6.23-mm1/kernel/pid.c 2007-10-27 10:03:28.000000000 -0700
> @@ -430,6 +430,19 @@ struct pid *find_get_pid(pid_t nr)
>     return pid;
> }
```

```

>
> +/*
> + * Return TRUE if the active pid namespace of @tsk is same as active
> + * pid namespace of 'current'.
> + *
> + * Note the difference between this and the task_in_pid_ns() below.
> + * task_in_pid_ns() includes processes in descendant pid name spaces
> + * but pid_ns_equal() only matches _active_ pid namespaces.
> + */
> +int pid_ns_equal(struct task_struct *tsk)
> +{
> + return pid_active_ns(task_pid(current)) == pid_active_ns(task_pid(tsk));
> +}
> +
> static int pid_in_pid_ns(struct pid *pid, struct pid_namespace *ns)
> {
> return pid && (ns->level <= pid->level) &&
> Index: 2.6.23-mm1/kernel/signal.c
> =====
> --- 2.6.23-mm1.orig/kernel/signal.c 2007-10-27 08:50:51.000000000 -0700
> +++ 2.6.23-mm1/kernel/signal.c 2007-10-27 10:02:04.000000000 -0700
> @@ -679,6 +679,20 @@ static void handle_stop_signal(int sig,
> }
> }
>
> +static void masquerade_sender(struct task_struct *t, struct sigqueue *q)
> +{
> + /*
> + * If the sender does not have a pid_t in the receiver's active
> + * pid namespace, set si_pid to 0 and pretend signal originated
> + * from the kernel.
> + */
> + if (!pid_ns_equal(t)) {
> +     q->info.si_pid = 0;
> +     q->info.si_uid = 0;
> +     q->info.si_code = SI_KERNEL;
> + }
> +}
> +
> static int send_signal(int sig, struct siginfo *info, struct task_struct *t,
> struct sigpending *signals)
> {
> @@ -730,6 +744,7 @@ static int send_signal(int sig, struct s
> copy_siginfo(&q->info, info);
> break;
> }
> + masquerade_sender(t, q);
> } else if (!is_si_special(info)) {

```

```

> if (sig >= SIGRTMIN && info->si_code != SI_USER)
> /*
> Index: 2.6.23-mm1/include/linux/pid_namespace.h
> =====
> --- 2.6.23-mm1.orig/include/linux/pid_namespace.h 2007-10-27 09:44:25.000000000 -0700
> +++ 2.6.23-mm1/include/linux/pid_namespace.h 2007-10-27 10:04:20.000000000 -0700
> @@ -56,6 +56,7 @@ static inline struct task_struct *task_c
> return tsk->nsproxy->pid_ns->child_reaper;
> }
>
> +extern int pid_ns_equal(struct task_struct *tsk);
> extern int task_in_pid_ns(struct task_struct *tsk, struct pid_namespace *ns);
>
> #endif /* _LINUX_PID_NS_H */
>

```

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [PATCH] Masquerade sender information
Posted by [Cedric Le Goater](#) on Fri, 02 Nov 2007 13:40:24 GMT
[View Forum Message](#) <> [Reply to Message](#)

Serge E. Hallyn wrote:

```

> Quoting Eric W. Biederman (ebiederm@xmission.com):
>> sukadev@us.ibm.com writes:
>>
>>> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
>>> Subject: [PATCH] Masquerade sender information
>>>
>>> With multiple pid namespaces, sender of a signal could be in an ancestor
>>> namespace of the receiver and so the sender will not have a valid 'pid_t'
>>> in the receiver's namespace.
>>>
>>> In this case, masquerade the 'siginfo' for the signal to pretend that the
>>> signal originated from the kernel.
>> At first glance this looks ok. I think the only case where we can
>> be sending a signal from inside a pid namespace to something not
>> in a child pid namespace is if we are the kernel. In which case
>
> Are we now blocking F_SETOWN|F_SETSIG signals to outside our pid
> namespace? mq_notify? (I didn't think we were)

```

My understanding is that we're not blocking and that a process killing another process in a sibling pid namespace will have a si_pid = 0.

C.

>
>> we also want si_pid = 0.
>>
>> If that holds this problem is easier then I was thinking it would
>> be.
>>
>> Eric
>>

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [PATCH] Masquerade sender information
Posted by [serue](#) on Fri, 02 Nov 2007 13:45:45 GMT
[View Forum Message](#) <> [Reply to Message](#)

Quoting Cedric Le Goater (clg@fr.ibm.com):

> Serge E. Hallyn wrote:
> > Quoting Eric W. Biederman (ebiederm@xmission.com):
> >> sukadev@us.ibm.com writes:
> >>
> >>> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> >>> Subject: [PATCH] Masquerade sender information
> >>>
> >>> With multiple pid namespaces, sender of a signal could be in an ancestor
> >>> namespace of the receiver and so the sender will not have a valid 'pid_t'
> >>> in the receiver's namespace.
> >>>
> >>> In this case, masquerade the 'siginfo' for the signal to pretend that the
> >>> signal originated from the kernel.
> >> At first glance this looks ok. I think the only case where we can
> >> be sending a signal from inside a pid namespace to something not
> >> in a child pid namespace is if we are the kernel. In which case
> >
> > Are we now blocking F_SETOWN|F_SETSIG signals to outside our pid
> > namespace? mq_notify? (I didn't think we were)
>
> My understanding is that we're not blocking and that a process killing
> another process in a sibling pid namespace will have a si_pid = 0.

And I think I'm fine with that, I was just wondering about Eric's claim
that only the kernel can send signals from inside a pidns to something

not in a child pidns. We can treat these cases as being from the kernel, but it's not in fact the case that the signals came from the kernel.

-serge

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [PATCH] Masquerade sender information

Posted by [Cedric Le Goater](#) on Fri, 02 Nov 2007 14:05:48 GMT

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

Eric W. Biederman wrote:

> sukadev@us.ibm.com writes:

>

>> +static void masquerade_sender(struct task_struct *t, struct sigqueue *q)

>> +{

>> + /*

>> + * If the sender does not have a pid_t in the receiver's active

>> + * pid namespace, set si_pid to 0 and pretend signal originated

>> + * from the kernel.

>> + */

>> + if (!pid_ns_equal(t)) {

>> + q->info.si_pid = 0;

>> + q->info.si_uid = 0;

>> + q->info.si_code = SI_KERNEL;

>> + }

>> +}

>

> It looks like we are hooked in the right place. However the way we

> are handling this appears wrong.

>

> First. If we have an si_code that does not use si_pid then we should

> not be changing si_pid, because the structure is a union and that field

> is not always a pid value.

>

>

> My gut feel says the code should be something like:

>

> switch (q->info->si_code & __SI_MASK) {

> case __SI_KILL:

> case __SI_CHILD:

> case __SI_RT:

> case __SI_MESQ:

> q->info->si_pid = task_pid_nr_ns(current, t->nsproxy->pid_ns);

```
> break;
> }
```

IMHO, it should be

```
q->info->si_pid = 0.
```

we're trying to cover the case where the sender does not have a pid_t in the receiver's active pid namespace.

C.

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [PATCH] Masquerade sender information
Posted by [ebiederm](#) on Sun, 04 Nov 2007 04:12:53 GMT
[View Forum Message](#) <> [Reply to Message](#)

Cedric Le Goater <clg@fr.ibm.com> writes:

```
> Eric W. Biederman wrote:
>> sukadev@us.ibm.com writes:
>>
>>> +static void masquerade_sender(struct task_struct *t, struct sigqueue *q)
>>> +{
>>> +    /*
>>> +     * If the sender does not have a pid_t in the receiver's active
>>> +     * pid namespace, set si_pid to 0 and pretend signal originated
>>> +     * from the kernel.
>>> +     */
>>> +    if (!pid_ns_equal(t)) {
>>> +        q->info.si_pid = 0;
>>> +        q->info.si_uid = 0;
>>> +        q->info.si_code = SI_KERNEL;
>>> +    }
>>> +}
>>
>> It looks like we are hooked in the right place. However the way we
>> are handling this appears wrong.
>>
>> First. If we have an si_code that does not use si_pid then we should
>> not be changing si_pid, because the structure is a union and that field
>> is not always a pid value.
>>
>>
```

```
>> My gut feel says the code should be something like:
>>
>> switch (q->info->si_code & __SI_MASK) {
>> case __SI_KILL:
>> case __SI_CHILD:
>> case __SI_RT:
>> case __MESQ:
>>     q->info->si_pid = task_pid_nr_ns(current, t->nspoxy->pid_ns);
>>     break;
>> }
>
> IMHO, it should be
>
> q->info->si_pid = 0.
>
> we're trying to cover the case where the sender does not have a pid_t in
> the receiver's active pid namespace.
```

Yes. However you are currently missing the case where the target pid namespace is a parent pid namespace. So besides applying the change to liberally we also missed the case when sending to a parent pid namespace. `task_tgid_nr_ns(current, t->nspoxy->pid_ns)` handles that.

Technically I think that is safe right now, but I think I would like to pass in the `task_struct` of the sender because we have a few odd instances where `current` is not the sender although every case I have traced we do continue to be in the same process group.

Eric

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
