
Subject: Re: [RFC PATCH 2/5] use next syscall data to predefine ipc objects ids
Posted by [Nadia Derby](#) on Tue, 08 Jul 2008 05:30:37 GMT
[View Forum Message](#) <> [Reply to Message](#)

Serge E. Hallyn wrote:

> Quoting Nadia.Derbey@bull.net (Nadia.Derbey@bull.net):

>

>>[PATCH 02/05]

>>

>>This patch uses the value written into the next_syscall_data proc file

>>as a target id for the next IPC object to be created.

>>The following syscalls have a new behavior if next_syscall_data is set:

>>. mssget()

>>. semget()

>>. shmget()

>>

>>Signed-off-by: Nadia Derby <Nadia.Derbey@bull.net>

>>

>>---

>> include/linux/next_syscall_data.h | 17 ++++++++-----

>> ipc/util.c | 38 +++++++++++++++++++++++++++++++++++++-----

>> 2 files changed, 45 insertions(+), 10 deletions(-)

>>

>>Index: linux-2.6.26-rc5-mm3/include/linux/next_syscall_data.h

>>=====

>>--- linux-2.6.26-rc5-mm3.orig/include/linux/next_syscall_data.h 2008-07-01

10:25:48.000000000 +0200

>>+++ linux-2.6.26-rc5-mm3/include/linux/next_syscall_data.h 2008-07-01 11:35:11.000000000

+0200

>>@@ -3,7 +3,8 @@

>> *

>> * Definitions to support fixed data for next syscall to be called. The

>> * following is supported today:

>>- * . object creation with a predefined id.

>>+ * . object creation with a predefined id

>>+ * . for a sysv ipc object

>> *

>> */

>>

>>@@ -16,13 +17,25 @@

>> * If this structure is pointed to by a task_struct, next syscall to be called

>> * by the task will have a non-default behavior.

>> * For example, it can be used to pre-set the id of the object to be created

>>- * by next syscall.

>>+ * by next syscall. The following syscalls support this feature:

>>+ * . msgget(), semget(), shmget()

>> */

>> struct next_syscall_data {

```

>> int ndata;
>> long data[NDATA];
>> };
>>
>>+/*
>>+ * Returns true if tsk has some data set in its next_syscall_data, 0 else
>>+ */
>>+#define next_data_set(tsk) ((tsk)->nsd \
>>+   ? ((tsk)->nsd->ndata ? 1 : 0) \
>>+   : 0)
>>+
>>+#define get_next_data(tsk) ((tsk)->nsd->data[0])
>>+
>>+
>>+
>> extern ssize_t get_next_syscall_data(struct task_struct *, char *, size_t);
>> extern int set_next_syscall_data(struct task_struct *, char *);
>> extern int reset_next_syscall_data(struct task_struct *);
>>Index: linux-2.6.26-rc5-mm3/ipc/util.c
>>=====
>>--- linux-2.6.26-rc5-mm3.orig/ipc/util.c 2008-07-01 10:25:48.000000000 +0200
>>+++ linux-2.6.26-rc5-mm3/ipc/util.c 2008-07-01 10:41:36.000000000 +0200
>>@@ -266,20 +266,42 @@ int ipc_addid(struct ipc_ids* ids, struc
>> if (ids->in_use >= size)
>> return -ENOSPC;
>>
>>- err = idr_get_new(&ids->ipcs_idr, new, &id);
>>- if (err)
>>- return err;
>>+ if (next_data_set(current)) {
>>+ /* There is a target id specified, try to use it */
>>+ int next_id = get_next_data(current);
>>+ int new_lid = next_id % SEQ_MULTIPLIER;
>>+
>>+ if (next_id !=
>>+     (new_lid + (next_id / SEQ_MULTIPLIER) * SEQ_MULTIPLIER))
>>+ return -EINVAL;
>
>
> You're leaving the next_data info set on error. Should we clear it?

```

Well. I presently leave this cleaning up to the calling application. But you're right, it is certainly cleaner to clear everything on error. The only reason for this being that the uncleared data will be taken for next syscall that is "next_syscall_data - sensitive".

>
> I think it seems more reasonable to clear it on error and just expect

```

> the application, if it wants to retry, re-set it to the desired id
> before retry.
>
>
>>+
>>+ err = idr_get_new_above(&ids->ipcs_idr, new, new_lid, &id);
>>+ if (err)
>>+     return err;
>>+ if (id != new_lid) {
>>+     idr_remove(&ids->ipcs_idr, id);
>>+     return -EBUSY;
>>+ }
>>+
>>+ new->id = next_id;
>>+ new->seq = next_id / SEQ_MULTIPLIER;
>>+ reset_next_syscall_data(current);
>>+ } else {
>>+     err = idr_get_new(&ids->ipcs_idr, new, &id);
>>+     if (err)
>>+         return err;
>>+
>>+ new->seq = ids->seq++;
>>+ if (ids->seq > ids->seq_max)
>>+     ids->seq = 0;
>>+ new->id = ipc_buildid(id, new->seq);
>>+ }
>>
>> ids->in_use++;
>>
>> new->cuid = new->uid = current->euid;
>> new->gid = new->cgid = current->egid;
>>
>>- new->seq = ids->seq++;
>>- if(ids->seq > ids->seq_max)
>>-     ids->seq = 0;
>>-
>>- new->id = ipc_buildid(id, new->seq);
>> spin_lock_init(&new->lock);
>> new->deleted = 0;
>> rcu_read_lock();
>>
>>--
>
>
>

```

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