Subject: Re: [RFC PATCH 5/5] use next syscall data to predefine the file descriptor value
Posted by kathys on Wed, 09 Jul 2008 04:59:00 GMT
View Forum Message <> Reply to Message

Hi Nadia,

I am trying with great difficulty to incorporate these patches into the
existing lxc-tree on 2.6.26-rc8-mm1-lxc1, they are conflicting with a
number of other patches from checkpoint/. Serge has asked me to include
them in the next lxc release so I need to know how to make them fit.

I will put out 2.6.26-rc8-mm1-lxc1 without your patches because its
taking me too long, I will endeavor to include them in the
2.6.26-rc8-mm1-lxc2, so if you could have a look at them against the
next release of lxc which I hope to get out by tomorrow (Thursday)
afternoon.

Thanks,

Kathy

Serge E. Hallyn wrote:
> Quoting Nadia.Derbey@bull.net (Nadia.Derbey@bull.net):
>
>> [PATCH 05/05]
>>
>> This patch uses the value written into the next_syscall_data proc file
>> as a target file descriptor for the next file to be opened.
>>
>> This makes it easy to restart a process with the same fds as the ones it was
>> using during the checkpoint phase, instead of 1. opening the file, 2. dup2'ing
>> the open file descriptor.
>>
>> The following syscalls are impacted if next_syscall_data is set:
>> . open()
>> . openat()
>>
>
> Oh, neat, I somehow missed the fact that you had this in your previous
> posting :)
>
>
>> Signed-off-by: Nadia Derbey <Nadia.Derbey@bull.net>
>>
>
> It'd be nice if the get_predefined_fd_flags() could share a helper
> with get_unused_fd_flags() (in particular because the "/* snaity check */

```
> at the end is between a '#if 1' which sounds like it may one day be
> removed), but I'm not sure offhand the best way to do that.  So for now
>
> Acked-by: Serge Hallyn <serue@us.ibm.com>
>
> Thanks, Nadia.
>
> Kathy, I'd love to see a -lxc release with this patchset so we can test
> it with cryo.
>
> Suka, the open with specified id here might help your simplify your pipe
> c/r patches for cryo?
>
> -serge
>
>
>> ---
>>  fs/open.c |   62
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++-
>>  1 file changed, 61 insertions(+), 1 deletion(-)
>>
>> Index: linux-2.6.26-rc8-mm1/fs/open.c
>> ===================================================================
>> --- linux-2.6.26-rc8-mm1.orig/fs/open.c 2008-07-08 12:12:34.000000000 +0200
>> +++ linux-2.6.26-rc8-mm1/fs/open.c 2008-07-08 13:23:03.000000000 +0200
>> @@ -974,6 +974,59 @@ struct file *dentry_open(struct dentry *
>>  EXPORT_SYMBOL(dentry_open);
>>
>>  /*
>> + * Marks a given file descriptor entry as busy (should not be busy when this
>> + * routine is called.
>> + *
>> + * files->next_fd is not updated: this lets the potentially created hole be
>> + * filled up on next calls to get_unused_fd_flags.
>> + *
>> + * Returns the specified fd if successful, -errno else.
>> + */
>> +static int get_predefined_fd_flags(int fd, int flags)
>> +{
>> + struct files_struct *files = current->files;
>> + int error;
>> + struct fdtable *fdt;
>> +
>> + error = -EINVAL;
>> + if (fd < 0)
>> +  goto out;
>> +
>> + error = -EMFILE;
>> + if (fd >= current->signal->rlim[RLIMIT_NOFILE].rlim_cur)
>> +  goto out;
>> +
>> + spin_lock(&files->file_lock);
>> + fdt = files_fdtable(files);
>> +
>> + error = expand_files(files, fd);
>> + if (error < 0)
>> +  goto out_unlock;
>> +
>> + error = -EBUSY;
>> + if (FD_ISSET(fd, fdt->open_fds))
>> +  goto out_unlock;
>> +
>> + FD_SET(fd, fdt->open_fds);
>> + if (flags & O_CLOEXEC)
>> +  FD_SET(fd, fdt->close_on_exec);
>> + else
>> +  FD_CLR(fd, fdt->close_on_exec);
>> +
>> + /* Sanity check */
>> + if (fdt->fd[fd] != NULL) {
>> +  printk(KERN_WARNING "get_unused_fd: slot %d not NULL!\n", fd);
>> +  fdt->fd[fd] = NULL;
>> + }
>> +
>> + error = fd;
>> +out_unlock:
>> + spin_unlock(&files->file_lock);
>> +out:
>> + return error;
>> +}
>> +
>> +/*
>>   * Find an empty file descriptor entry, and mark it busy.
>>   */
>>  int get_unused_fd_flags(int flags)
>> @@ -1088,7 +1141,14 @@ long do_sys_open(int dfd, const char __u
>>   int fd = PTR_ERR(tmp);
>>
>>   if (!IS_ERR(tmp)) {
>> -  fd = get_unused_fd_flags(flags);
>> +  if (unlikely(next_data_set(current))) {
>> +   int next_fd = get_next_data(current);
>> +
>> +   fd = get_predefined_fd_flags(next_fd, flags);
>> +   reset_next_syscall_data(current);
>> +  } else
>> +   fd = get_unused_fd_flags(flags);
>> +
>>    if (fd >= 0) {
>>     struct file *f = do_filp_open(dfd, tmp, flags, mode);
>>     if (IS_ERR(f)) {
>>
>> --
>>
>
>
```