Subject: Re: Userspace checkpoint/restart hack: cryo
Posted by kathys on Tue, 17 Jun 2008 05:48:27 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn wrote:
> Quoting kathys (kathys@ozlabs.au.ibm.com):
>
>> Serge E. Hallyn wrote:
>>
>>> Quoting Nadia Derbey (Nadia.Derbey@bull.net):
>>>
>>>
>>>> Serge E. Hallyn wrote:
>>>>
>>>>
>>>>> Quoting Nadia Derbey (Nadia.Derbey@bull.net):
>>>>>
>>>>>
>>>>>
>>>>>> Cedric Le Goater wrote:
>>>>>>
>>>>>>
>>>>>>
>>>>>>>
>>>>>>>
>>>>>>>> Looks like it has worked for me (msg1 creates 1000 msg queues, sleeps
>>>>>>>> for a while and then removes the msg queues).
>>>>>>>>
>>>>>>>>
>>>>>>> cool. which kernel are you using ?
>>>>>>>
>>>>>>> C.
>>>>>>>
>>>>>>>
>>>>>>>
>>>>>>>
>>>>>> 2.6.25-mm1 (sorry for the late answer - plenty of days off in
>>>>>> France in May).
>>>>>>
>>>>>> Regards,
>>>>>> Nadia
>>>>>>
>>>>>>
>>>>> I'm playing with features in cryo, and keeping a git tree at:
>>>>>
>>>>> git://git.sr71.net/~hallyn/cryodev.git
>>>>>
>>>>> It's meant to exploit the extras which are in the -lxc kernel at

>>>>> lxc.sf.net. Current latest kernel patch is at
>>>>> http://lxc.sourceforge.net/patches/2.6.26/2.6.26-rc2-mm1-lxc4/ This -lxc
>>>>> tree contains, for instance, Nadia's next_id patches, exploitation for
>>>>> setting ids for sysvipc and for tasks at fork, and updated ipc_setall
>>>>> patches (also using next_id). The version of cryo in my git tree
>>>>> exploits these. If you're root when you restart a task, it will clone a
>>>>> new set of namespaces and recreate your sysvipc objects, and it will
>>>>> reset your pids (even if you're not root if the pids are available).
>>>>>
>>>>>
>>>> Serge,
>>>>
>>>> I noticed that the sys_hijack() has disappeared from the lxc dev
>>>> tree: would you mind putting it back. I think it might be useful if
>>>> we want to start a task in a newly defined cgroup, checkpoint it and
>>>> then try to restart it. We will need to 'join' the restarted
>>>> container to check if everything has correctly be restarted. Or may
>>>> be is there another way to do that?
>>>>
>>>>
>>> Well it *could* be done similar to how cryo itself works, by ptracing
>>> the destination task and making it fork a task which execve()s a process
>>> to do the querying. But yuck.
>>>
>>> Kathy, I'm sorry, I know I asked you to take sys_hijack() out. Could
>>> you please put it back in? Preferably at the end of the queue, as I don't
>>> want other patches having to be ported on top of it since its future is
>>> very suspect... Let me know if you have trouble porting it, but it
>>> should be pretty simple.
>>>
>>>
>> I added sys_hijack() (namespaces-introduce-sys_hijack.patch) back, with
>> a little bit of massaging to get it to port properly (as Cedrics
>> clone64-change-clone_flag-type-to-u64.patch changes unsigned long to
>> u64) but was unable to compile. I received the following errors:
>>
>> /scratch/kathys/containers/kernel_trees/upstream/kernel/fork.c: In
>> function 'do_fork_task':
>> /scratch/kathys/containers/kernel_trees/upstream/kernel/fork.c:1342:
>> warning: format '%1lx' expects type 'long unsigned int', but argument 3
>> has type 'u64'
>> /scratch/kathys/containers/kernel_trees/upstream/kernel/fork.c: At top
>> level:
>> /scratch/kathys/containers/kernel_trees/upstream/kernel/fork.c:1416:
>> error: conflicting types for 'do_fork'
>>
>
>

>> /scratch/kathys/containers/kernel_trees/upstream/include/linux/sched.h:1868:
>> error: previous declaration of 'do_fork' was here
>>
>
> Did you change the do_fork definition in sched.h?
>
> -serge
>
>
I changed definition in sched.h to match do_fork in fork.c
>> I changed the patch as follows so it would apply properly, but I can't
>> work out why it breaks now, but not previously:
>>
>> Original patch from 2.6.26-rc2-mm1-lxc3:
>> @@ -1307,13 +1313,8 @@ static int fork_traceflag(unsigned clone
>> return 0;
>> }
>>
>> -/*
>> - * Ok, this is the main fork-routine.
>> - *
>> - * It copies the process, and if successful kick-starts
>> - * it and waits for it to finish using the VM if required.
>> - */
>> -long do_fork(unsigned long clone_flags,
>> +long do_fork_task(struct cgroup *cgroup,
>> + unsigned long clone_flags,
>> unsigned long stack_start,
>> struct pt_regs *regs,
>> unsigned long stack_size,
>>
>>
>> Changed to:
>>
>> @@ -1308,13 +1314,8 @@ static int fork_traceflag(u64 clone_flag
>> return 0;
>> }
>>
>> -/*
>> - * Ok, this is the main fork-routine.
>> - *
>> - * It copies the process, and if successful kick-starts
>> - * it and waits for it to finish using the VM if required.
>> - */
>> -long do_fork(u64 clone_flags,
>> +long do_fork_task(struct cgroup *cgroup,
>> + u64 clone_flags,
>> unsigned long stack_start,

>> struct pt_regs *regs,
>> unsigned long stack_size,
>>
>>
>>
>>
>>> thanks,
>>> -serge
>>> _____
>>> Containers mailing list
>>> Containers@lists.linux-foundation.org
>>> https://lists.linux-foundation.org/mailman/listinfo/containers
>>>
>>>
>>>
>
>

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