Subject: Userspace checkpoint/restart hack: cryo
Posted by Dave Hansen on Fri, 25 Apr 2008 17:24:19 GMT
View Forum Message <> Reply to Message

A guy named Marc Vertes wrote this as a little demonstration of
checkpoint/restart. I've been using it to experiment with
checkpoint/restart. I thought it might be of some use as we move
subsystems to being helped by the kernel to checkpoint and restart.

It's ptrace-based, and stuck on i386 for now. It can probably be ported
elsewhere without too much trouble. It doesn't support *anything
fancy like multuiple tasks :). It has the advantage of being very
feature-bare, and I think it is pretty easy to hack on. Whatever c/r
support we add to the kernel could easily be added on and tested.

http://userweb.kernel.org/~daveh/cryo/cryo-001.tar.gz

Usage:
cr -p `pidof task` > checkpoint.cryo
cr -r < checkpoint.cryo

If anyone else has something simpler or easier to hack on, I'm all ears.

-- Dave

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

Subject: Re: Userspace checkpoint/restart hack: cryo
Posted by Cedric Le Goater on Mon, 28 Apr 2008 09:47:09 GMT
View Forum Message <> Reply to Message

Dave Hansen wrote:
> A guy named Marc Vertes wrote this as a little demonstration of
> checkpoint/restart. I've been using it to experiment with
> checkpoint/restart. I thought it might be of some use as we move
> subsystems to being helped by the kernel to checkpoint and restart.
>
> It's ptrace-based, and stuck on i386 for now. It can probably be ported
> elsewhere without too much trouble. It doesn't support *anything
> fancy like multuiple tasks :). It has the advantage of being very
> feature-bare, and I think it is pretty easy to hack on. Whatever c/r
> support we add to the kernel could easily be added on and tested.
>
> http://userweb.kernel.org/~daveh/cryo/cryo-001.tar.gz

>
> Usage:
> cr -p `pidof task` > checkpoint.cryo
> cr -r < checkpoint.cryo
>
> If anyone else has something simpler or easier to hack on, I'm all ears.

Indeed. It looks simple enough.

do you have some kernel requirement ? I run Fedora 8

Here's my first try on a program calculating decimal of PI :

```
 $ ./cr -p `pidof pi1` > pi1.cryo
 attaching to pid: 11082
 [11087 cr.c:243 getfdinfo()] n : 0
 WARNING (sci.c:242) unexpected signal for 11082: 11
 [11087 sci.c:228 ptrace_waitsyscall()] WTERMSIG(status) : 11
 ERROR (sci.c:383) ptrace_getregs(11082, 0xbfe4a3d0) errno=3: No such process
 ./cr[0x8051f10]
 ./cr[0x8049ce9]
 ./cr[0x804b7d2]
 ./cr[0x804f75b]
 ***STOP***
```

other terminal :

```
 $ pi1 20000
 pi1 - 20000 digits, 78.1 kbytes
 Segmentation fault (core dumped)
```

Thanks,

C.

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>

```
--- Makefile~   2008-04-25 19:05:57.000000000 +0200
+++ Makefile    2008-04-28 09:02:50.000000000 +0200
@@ -7,6 +7,3 @@ all : $(BIN) $(MAN1)

 #cr: cr.o utils.o list_hash.o ptrace_linux_x86.o
 cr: cr.o utils.o list_hash.o sci.o injlib.o
-
-include ../version.mk
-include ../c.mk
```

Subject: Re: Userspace checkpoint/restart hack: cryo
Posted by Nadia Derbey on Tue, 29 Apr 2008 14:50:14 GMT
View Forum Message <> Reply to Message

Cedric Le Goater wrote:
> Dave Hansen wrote:
>
>>A guy named Marc Vertes wrote this as a little demonstration of
>>checkpoint/restart. I've been using it to experiment with
>>checkpoint/restart. I thought it might be of some use as we move
>>subsystems to being helped by the kernel to checkpoint and restart.
>>
>>It's ptrace-based, and stuck on i386 for now. It can probably be ported
>>elsewhere without too much trouble. It doesn't support *anything
>>fancy like multuiple tasks :). It has the advantage of being very
>>feature-bare, and I think it is pretty easy to hack on. Whatever c/r
>>support we add to the kernel could easily be added on and tested.
>>
>>http://userweb.kernel.org/~daveh/cryo/cryo-001.tar.gz
>>
>>Usage:
>> cr -p `pidof task` > checkpoint.cryo
>> cr -r < checkpoint.cryo
>>
>>If anyone else has something simpler or easier to hack on, I'm all ears.
>
>
> Indeed. It looks simple enough.
>
> do you have some kernel requirement ? I run Fedora 8
>
> Here's my first try on a program calculating decimal of PI :
>
> $ ./cr -p `pidof pi1` > pi1.cryo
> attaching to pid: 11082
> [11087 cr.c:243 getfdinfo()] n : 0
> WARNING (sci.c:242) unexpected signal for 11082: 11
> [11087 sci.c:228 ptrace_waitsyscall()] WTERMSIG(status) : 11
> ERROR (sci.c:383) ptrace_getregs(11082, 0xbfe4a3d0) errno=3: No such process
> ./cr[0x8051f10]
> ./cr[0x8049ce9]
> ./cr[0x804b7d2]
> ./cr[0x804f75b]
> ***STOP***
>

> other terminal :
>
> $ pi1 20000
> pi1 - 20000 digits, 78.1 kbytes
> Segmentation fault (core dumped)
>
> Thanks,
>
> C.
>

Looks like it has worked for me (msg1 creates 1000 msg queues, sleeps
for a while and then removes the msg queues).

Output attached.

Regards,
Nadia

[root@akt cryo-001]# ./cr -p `pidof msg1` > msg1.cryo
attaching to pid: 4689
[4695 cr.c:243 getfdinfo()] n : 0
[4695 cr.c:243 getfdinfo()] n : 1
[4695 cr.c:243 getfdinfo()] n : 2
DEBUG (utils.c::25) write_item() writing item named: 'pid' datasize: 4 to positi
on: 0
DEBUG (utils.c::25) write_item() writing item named: 'ppid' datasize: 4 to posit
ion: 10
DEBUG (utils.c::25) write_item() writing item named: 'exitsig' datasize: 4 to po
sition: 21
DEBUG (utils.c::25) write_item() writing item named: 'fpregs' datasize: 108 to p
osition: 35
DEBUG (utils.c::25) write_item() writing item named: 'regs' datasize: 68 to posi
tion: 154
DEBUG (utils.c::25) write_item() writing item named: 'exe' datasize: 32 to posit
ion: 230
DEBUG (utils.c::25) write_item() writing item named: 'argv' datasize: 7 to posit
ion: 269
DEBUG (utils.c::25) write_item() writing item named: 'env' datasize: 1137 to pos
ition: 283
DEBUG (utils.c::25) write_item() writing item named: 'cwd' datasize: 27 to posit
ion: 1429
DEBUG (utils.c::25) write_item() writing item named: 'sigact' datasize: 9240 to
position: 1463
DEBUG (utils.c::25) write_item() writing item named: 'sigmask' datasize: 128 to

position: 10715
DEBUG (utils.c::25) write_item() writing item named: 'sigpend' datasize: 128 to
position: 10855
DEBUG (utils.c::25) write_item() writing item named: 'FD' datasize: 0 to positio
n: 10995
[4695 cr.c:512 checkpoint()] pi->nf : 3
DEBUG (utils.c::25) write_item() writing item named: 'fdinfo' datasize: 148 to p
osition: 11000
DEBUG (utils.c::25) write_item() writing item named: 'fdinfo' datasize: 148 to p
osition: 11159
DEBUG (utils.c::25) write_item() writing item named: 'fdinfo' datasize: 148 to p
osition: 11318
DEBUG (utils.c::25) write_item() writing item named: 'END FD' datasize: 0 to pos
ition: 11477
DEBUG (utils.c::25) write_item() writing item named: 'SOCK' datasize: 0 to posit
ion: 11486
DEBUG (utils.c::25) write_item() writing item named: 'END SOCK' datasize: 0 to p
osition: 11493
DEBUG (utils.c::25) write_item() writing item named: 'MEM' datasize: 0 to positi
on: 11504
getmaps() '' is anonymous (old test)
getmaps() '' is anonymous (new test)
getmaps() '' is anonymous (old test)
getmaps() '' is anonymous (new test)
getmaps() '' is anonymous (old test)
getmaps() '' is anonymous (new test)
getmaps() '[stack]' is anonymous (new test)
getmaps() '[vdso]' is anonymous (new test)
DEBUG (utils.c::25) write_item() writing item named: 'memseg' datasize: 148 to p
osition: 11510
DEBUG (cr.c::528) mem i=0 0x850000 -> 0x869000 /lib/ld-2.5.so
DEBUG (utils.c::25) write_item() writing item named: 'membuf' datasize: 0 to pos
ition: 11669
DEBUG (utils.c::25) write_item() writing item named: 'memseg' datasize: 148 to p
osition: 11678
DEBUG (cr.c::528) mem i=1 0x869000 -> 0x86a000 /lib/ld-2.5.so
DEBUG (utils.c::25) write_item() writing item named: 'membuf' datasize: 0 to pos
ition: 11837
DEBUG (utils.c::25) write_item() writing item named: 'memseg' datasize: 148 to p
osition: 11846
DEBUG (cr.c::528) mem i=2 0x86a000 -> 0x86b000 /lib/ld-2.5.so
DEBUG (cr.c::530) mem i=2 saved size = 4 KB
DEBUG (utils.c::25) write_item() writing item named: 'membuf' datasize: 4096 to
position: 12005
DEBUG (utils.c::25) write_item() writing item named: 'memseg' datasize: 148 to p
osition: 16113
DEBUG (cr.c::528) mem i=3 0x86d000 -> 0x9a4000 /lib/libc-2.5.so
DEBUG (utils.c::25) write_item() writing item named: 'membuf' datasize: 0 to pos

ition: 16272
DEBUG (utils.c::25) write_item() writing item named: 'memseg' datasize: 148 to p
osition: 16281
DEBUG (cr.c::528) mem i=4 0x9a4000 -> 0x9a6000 /lib/libc-2.5.so
DEBUG (utils.c::25) write_item() writing item named: 'membuf' datasize: 0 to pos
ition: 16440
DEBUG (utils.c::25) write_item() writing item named: 'memseg' datasize: 148 to p
osition: 16449
DEBUG (cr.c::528) mem i=5 0x9a6000 -> 0x9a7000 /lib/libc-2.5.so
DEBUG (cr.c::530) mem i=5 saved size = 4 KB
DEBUG (utils.c::25) write_item() writing item named: 'membuf' datasize: 4096 to position: 16608
DEBUG (utils.c::25) write_item() writing item named: 'memseg' datasize: 148 to position: 20716
DEBUG (cr.c::528) mem i=6 0x9a7000 -> 0x9aa000
DEBUG (cr.c::530) mem i=6 saved size = 12 KB
DEBUG (utils.c::25) write_item() writing item named: 'membuf' datasize: 12288 to position: 20875
DEBUG (utils.c::25) write_item() writing item named: 'memseg' datasize: 148 to position: 33176
DEBUG (cr.c::528) mem i=7 0x8048000 -> 0x8049000 /home/lkernel/src_ref/tests/msg1
DEBUG (utils.c::25) write_item() writing item named: 'membuf' datasize: 0 to position: 33335
DEBUG (utils.c::25) write_item() writing item named: 'memseg' datasize: 148 to position: 33344
DEBUG (cr.c::528) mem i=8 0x8049000 -> 0x804a000 /home/lkernel/src_ref/tests/msg1
DEBUG (cr.c::530) mem i=8 saved size = 4 KB
DEBUG (utils.c::25) write_item() writing item named: 'membuf' datasize: 4096 to position: 33503
DEBUG (utils.c::25) write_item() writing item named: 'memseg' datasize: 148 to position: 37611
DEBUG (cr.c::528) mem i=9 0xb7f98000 -> 0xb7f99000
DEBUG (cr.c::530) mem i=9 saved size = 4 KB
DEBUG (utils.c::25) write_item() writing item named: 'membuf' datasize: 4096 to position: 37770
DEBUG (utils.c::25) write_item() writing item named: 'memseg' datasize: 148 to position: 41878
DEBUG (cr.c::528) mem i=10 0xb7fa6000 -> 0xb7fa8000
DEBUG (cr.c::530) mem i=10 saved size = 8 KB
DEBUG (utils.c::25) write_item() writing item named: 'membuf' datasize: 8192 to position: 42037
DEBUG (utils.c::25) write_item() writing item named: 'memseg' datasize: 148 to position: 50241
DEBUG (cr.c::528) mem i=11 0xbfc93000 -> 0xbfca8000 [stack]
DEBUG (cr.c::530) mem i=11 saved size = 84 KB
DEBUG (utils.c::25) write_item() writing item named: 'membuf' datasize: 86016 to position: 50400
DEBUG (utils.c::25) write_item() writing item named: 'memseg' datasize: 148 to position: 136429
DEBUG (cr.c::528) mem i=12 0xffffe000 -> 0xfffff000 [vdso]
DEBUG (utils.c::25) write_item() writing item named: 'membuf' datasize: 0 to position: 136588
DEBUG (utils.c::25) write_item() writing item named: 'END MEM' datasize: 0 to position: 136597

_____

Subject: Re: Userspace checkpoint/restart hack: cryo

Posted by Cedric Le Goater on Tue, 29 Apr 2008 15:21:05 GMT
View Forum Message <> Reply to Message

> 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.

Subject: Re: Userspace checkpoint/restart hack: cryo
Posted by Nadia Derbey on Mon, 05 May 2008 08:50:10 GMT
View Forum Message <> Reply to Message

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

Subject: Re: Userspace checkpoint/restart hack: cryo

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).

Cryo is still rather touchy - I'm trying to track this down. Older
distros seem to actually have the most success (I'm told FC6 is somewhat
useful, while my FC8 kvm image is practically useless with cryo).

So if you want to play with it, please clone and do so, and send me any
patches you think should go in.

thanks,
-serge

_____
Containers mailing list
Containers@lists.linux-foundation.org

Subject: Re: Userspace checkpoint/restart hack: cryo
Posted by Nadia Derbey on Mon, 09 Jun 2008 15:02:42 GMT
View Forum Message <> Reply to Message

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?

Regards,
Nadia

>
> Cryo is still rather touchy - I'm trying to track this down. Older
> distros seem to actually have the most success (I'm told FC6 is somewhat
> useful, while my FC8 kvm image is practically useless with cryo).
>
> So if you want to play with it, please clone and do so, and send me any
> patches you think should go in.
>
> 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

---

Subject: Re: Userspace checkpoint/restart hack: cryo
Posted by serue on Mon, 09 Jun 2008 15:23:36 GMT
View Forum Message <> Reply to Message

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.

thanks,
-serge

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

---

Subject: Re: Userspace checkpoint/restart hack: cryo
Posted by serue on Tue, 10 Jun 2008 18:17:56 GMT
View Forum Message <> Reply to Message

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 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

---

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