Subject: [PATCH] namespaces: fix exit race by splitting exit
Posted by serue on Fri, 26 Jan 2007 05:26:59 GMT
View Forum Message <> Reply to Message

Ok, could you verify that the following patch at least solves
the oopsing?

(I can't reproduce the oops with Daniel's test prog)

thanks,
-serge

From: Serge E. Hallyn <serue@us.ibm.com>
Subject: [PATCH] namespaces: fix exit race by splitting exit

Fix exit race by splitting the nsproxy putting into two pieces.
First piece reduces the nsproxy refcount. If we dropped the last
reference, then it puts the mnt_ns, and returns the nsproxy as a
hint to the caller. Else it returns NULL. The second piece of
exiting task namespaces sets tsk->nsproxy to NULL, and drops the
references to other namespaces and frees the nsproxy only if an
nsproxy was passed in.

A little awkward and should probably be reworked, but hopefully
it fixes the NFS oops.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>

---

 include/linux/nsproxy.h | 30 ++++++++++++++++++++----------
 kernel/exit.c | 6 ++++--
 kernel/fork.c | 4 ++--
 kernel/nsproxy.c | 16 ++++++++++++++-
 4 files changed, 40 insertions(+), 16 deletions(-)

ab969afa3624aba0bc26dc237d27178137c05d46
diff --git a/include/linux/nsproxy.h b/include/linux/nsproxy.h
index 0b9f0dc..678e1d3 100644
--- a/include/linux/nsproxy.h
+++ b/include/linux/nsproxy.h
@@ -35,22 +35,30 @@ struct nsproxy *dup_namespaces(struct ns
 int copy_namespaces(int flags, struct task_struct *tsk);
 void get_task_namespaces(struct task_struct *tsk);
 void free_nsproxy(struct nsproxy *ns);
+struct nsproxy *put_nsproxy(struct nsproxy *ns);

-static inline void put_nsproxy(struct nsproxy *ns)

```
+static inline void finalize_put_nsproxy(struct nsproxy *ns)
 {
- if (atomic_dec_and_test(&ns->count)) {
+ if (ns)
   free_nsproxy(ns);
- }
 }

-static inline void exit_task_namespaces(struct task_struct *p)
+static inline void put_and_finalize_nsproxy(struct nsproxy *ns)
 {
- struct nsproxy *ns = p->nsproxy;
- if (ns) {
-  task_lock(p);
-  p->nsproxy = NULL;
-  task_unlock(p);
-  put_nsproxy(ns);
- }
+ finalize_put_nsproxy(put_nsproxy(ns));
+}
+
+static inline struct nsproxy *preexit_task_namespaces(struct task_struct *p)
+{
+ return put_nsproxy(p->nsproxy);
+}
+
+static inline void exit_task_namespaces(struct task_struct *p,
+       struct nsproxy *ns)
+{
+ task_lock(p);
+ p->nsproxy = NULL;
+ task_unlock(p);
+ finalize_put_nsproxy(ns);
 }
 #endif
diff --git a/kernel/exit.c b/kernel/exit.c
index 3540172..a5bf532 100644
--- a/kernel/exit.c
+++ b/kernel/exit.c
@@ -396,7 +396,7 @@ void daemonize(const char *name, ...)
  current->fs = fs;
  atomic_inc(&fs->count);

- exit_task_namespaces(current);
+ put_and_finalize_nsproxy(current->nsproxy);
  current->nsproxy = init_task.nsproxy;
  get_task_namespaces(current);
```

```
@@ -853,6 +853,7 @@ static void exit_notify(struct task_stru
 fastcall NORET_TYPE void do_exit(long code)
 {
  struct task_struct *tsk = current;
+ struct nsproxy *ns;
  int group_dead;

  profile_task_exit(tsk);
@@ -938,8 +939,9 @@ fastcall NORET_TYPE void do_exit(long co

  tsk->exit_code = code;
  proc_exit_connector(tsk);
+ ns = preexit_task_namespaces(tsk);
  exit_notify(tsk);
- exit_task_namespaces(tsk);
+ exit_task_namespaces(tsk, ns);
 #ifdef CONFIG_NUMA
  mpol_free(tsk->mempolicy);
  tsk->mempolicy = NULL;
diff --git a/kernel/fork.c b/kernel/fork.c
index fc723e5..4cf8684 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1265,7 +1265,7 @@ static struct task_struct *copy_process(
  return p;

 bad_fork_cleanup_namespaces:
- exit_task_namespaces(p);
+ put_and_finalize_nsproxy(p->nsproxy);
 bad_fork_cleanup_keys:
  exit_keys(p);
 bad_fork_cleanup_mm:
@@ -1711,7 +1711,7 @@ asmlinkage long sys_unshare(unsigned lon
  }

  if (new_nsproxy)
-  put_nsproxy(new_nsproxy);
+  put_and_finalize_nsproxy(new_nsproxy);

 bad_unshare_cleanup_ipc:
  if (new_ipc)
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index f5b9ee6..7b05bce 100644
--- a/kernel/nsproxy.c
+++ b/kernel/nsproxy.c
@@ -117,7 +117,7 @@ int copy_namespaces(int flags, struct ta
   goto out_pid;
```

```
 out:
- put_nsproxy(old_ns);
+ put_and_finalize_nsproxy(old_ns);
  return err;

 out_pid:
@@ -135,6 +135,20 @@ out_ns:
  goto out;
 }

+struct nsproxy *put_nsproxy(struct nsproxy *ns)
+{
+ if (ns) {
+  if (atomic_dec_and_test(&ns->count)) {
+   if (ns->mnt_ns) {
+    put_mnt_ns(ns->mnt_ns);
+    ns->mnt_ns = NULL;
+   }
+   return ns;
+  }
+ }
+ return NULL;
+}
+
 void free_nsproxy(struct nsproxy *ns)
 {
  if (ns->mnt_ns)
--
1.1.6
```

Subject: Re: [PATCH] namespaces: fix exit race by splitting exit
Posted by Daniel Hokka Zakrisso on Fri, 26 Jan 2007 10:09:46 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn wrote:
> Ok, could you verify that the following patch at least solves
> the oopsing?
>
> (I can't reproduce the oops with Daniel's test prog)
>
> thanks,
> -serge

Indeed, this patch solves the oopsing, but so did the last one. I think I
finally managed to figure out why too, as release_task would be called
with current referring to the parent process, or whoever is reaping the
process with the last reference to the namespace. Right?

Regardless, I have to say I prefer this patch.

--
Daniel Hokka Zakrisson

Subject: Re: [PATCH] namespaces: fix exit race by splitting exit
Posted by Oleg Nesterov on Fri, 26 Jan 2007 10:23:15 GMT
View Forum Message <> Reply to Message

On 01/26, Daniel Hokka Zakrisson wrote:
>
> Serge E. Hallyn wrote:
> > Ok, could you verify that the following patch at least solves
> > the oopsing?
> >
> > (I can't reproduce the oops with Daniel's test prog)
> >
> > thanks,
> > -serge
>
> Indeed, this patch solves the oopsing, but so did the last one. I think I
> finally managed to figure out why too, as release_task would be called
> with current referring to the parent process, or whoever is reaping the
> process with the last reference to the namespace. Right?

Not always. The task can do release_task() on itself if it is sub-thread,
or its parent ignores SIGCHLD.

Anyway I think your explanation is correct for this particular test-case.

Oleg.

Subject: Re: [PATCH] namespaces: fix exit race by splitting exit
Posted by serue on Tue, 30 Jan 2007 03:00:39 GMT
View Forum Message <> Reply to Message

Quoting Herbert Poetzl (herbert@13thfloor.at):
> On Thu, Jan 25, 2007 at 10:30:56PM -0800, Andrew Morton wrote:
> > On Thu, 25 Jan 2007 23:26:59 -0600
> > "Serge E. Hallyn" <serue@us.ibm.com> wrote:
> >
> > > Fix exit race by splitting the nsproxy putting into two pieces.
> > > First piece reduces the nsproxy refcount. If we dropped the last
> > > reference, then it puts the mnt_ns, and returns the nsproxy as a
> > > hint to the caller. Else it returns NULL. The second piece of
> > > exiting task namespaces sets tsk->nsproxy to NULL, and drops the
> > > references to other namespaces and frees the nsproxy only if an
> > > nsproxy was passed in.
> > >
> > > A little awkward and should probably be reworked, but hopefully
> > > it fixes the NFS oops.
> >
> > I'm a bit worried about jamming something like this into 2.6.20.
> > Could the usual culprits please review this carefully with
> > some urgency?
>
> okay, after integrating this into two Linux-VServer
> branches and some testing, I can confirm that it
> _seems_ to fix the nfs and related issues, but still,
> I do not like it :)

I don't either :)

> here my issues with this approach:
>
> - the code is quite hard to read and can easily
> lead to unexpected issues when spaces are
> manipulated

Yes, but I do think fixing the naming will help that.

> - it breaks the basic get/put refcounting for
> nsproxy references outside the task struct
> i.e. we had to add a vs_put_nsproxy() which
> does what the put_nsproxy() did before, to
> keep and handle a reference to the nsproxy
> from the context structure

Was the put_and_finalize_nsproxy() not sufficient?

> - the following scenario might become a problem

> for future spaces (especially the pid space?)
>
> A B
>
> exit_task_namespaces_early()
> exit_task_namespaces_early()
> exit_notify()
> exit_task_namespaces()
> ---
> exit_notify()
> exit_task_namespaces()

Confounded, you're right, the exit_task_namespaces() in B
would see that B had reduced the nsproxy->count to 0, and
free the nsproxy, so that exit_notify() in A would oops.
And that should be triggerable right now.

I'm afraid adding an extra refcount for the mounts is
unavoidable.

> note: I still consider it the best available fix
> for this issues, especially as 2.6.20 is in a
> late rc stage ... but IMHO the nfs threads should
> be modified to handle the nsproxy disposal properly

That *would* lead to much more readable code.

> > And Daniel, if you can find time to runtime test it please?
>
> he did, looks like it works fine with vanilla too
> (even when stressing the described cornercase)
>
> best,
> Herbert

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers