Subject: Re: [PATCH] namespaces: introduce sys_hijack (v4)
Posted by serue on Thu, 11 Oct 2007 22:15:34 GMT
View Forum Message <> Reply to Message

Quoting Serge E. Hallyn (serue@us.ibm.com):
> >From 945fe66259cd0cfdc2fe846287b7821e329a558c Mon Sep 17 00:00:00 2001
> From: sergeh@us.ibm.com <hallyn@kernel.(none)>
> Date: Tue, 9 Oct 2007 08:30:30 -0700
> Subject: [PATCH] namespaces: introduce sys_hijack (v4)
>
> Move most of do_fork() into a new do_fork_task() which acts on
> a new argument, task, rather than on current. do_fork() becomes
> a call to do_fork_task(current, ...).
>
> Introduce sys_hijack (for x86 only so far). It is like clone, but
> in place of a stack pointer (which is assumed null) it accepts a
> pid. The process identified by that pid is the one which is
> actually cloned. Some state - include the file table, the signals
> and sighand (and hence tty), and the ->parent are taken from the
> calling process.
>
> The effect is a sort of namespace enter. The following program
> uses sys_hijack to 'enter' all namespaces of the specified pid.
> For instance in one terminal, do
>
> mount -t cgroup -ons /cgroup
> hostname
> qemu
> ns_exec -u /bin/sh
> hostname serge
> echo $$
> 1073
> cat /proc/$$/cgroup
> ns:/node_1073
>
> In another terminal then do
>
> hostname
> qemu
> cat /proc/$$/cgroup
> ns:/
> hijack 1073
> hostname
> serge
> cat /proc/$$/cgroup
> ns:/node_1073
>
> sys_hijack is arch-dependent and is only implemented for i386 so far.

>
> Changelog:
> Aug 23: send a stop signal to the hijacked process
>   (like ptrace does).
> Oct 09: Update for 2.6.23-rc8-mm2 (mainly pidns)
>   Don't take task_lock under rcu_read_lock
>   Send hijacked process to cgroup_fork() as
>   the first argument.
>   Removed some unneeded task_locks.

Thanks to Cedric for finding an oops when using pid namespaces. The
following patch fixes the problem.

In addition, to hijack a process in another pid namespace, the
hijack.c test program needs to be updated to do waitpid as
 while(waitpid(-1, &status, __WALL) != -1)
  ;

as shown below:

==========
hijack.c
==========

```
int do_clone_task(void)
{
 execl("/bin/sh", "/bin/sh", NULL);
}

int main(int argc, char *argv[])
{
 int pid;
 int ret;
 int status;

 if (argc < 2)
  return 1;
 pid = atoi(argv[1]);

 ret = syscall(327, SIGCHLD, pid, NULL, NULL);

 if  (ret == 0) {
  return do_clone_task();
 } else if (ret < 0) {
  perror("sys_hijack");
 } else {
  printf("waiting on cloned process %d\n", ret);
  while(waitpid(-1, &status, __WALL) != -1)
```

```
   ;
  printf("cloned process %d exited with %d\n", ret, status);
 }

 return ret;
}
============================================================
```

>From f1d9621e8325471e3ccde7f5fc2ed5a7be582524 Mon Sep 17 00:00:00 2001
From: sergeh@us.ibm.com <hallyn@kernel.(none)>
Date: Thu, 11 Oct 2007 14:26:05 -0700
Subject: [PATCH 2/2] hijack: pidns bugfix

My change to alloc_pid was bogus and introduced a pidns bug. Fix.

Signed-off-by: sergeh@us.ibm.com <hallyn@kernel.(none)>
---
 include/linux/pid.h | 2 +-
 kernel/fork.c | 2 +-
 kernel/pid.c | 5 ++---
 3 files changed, 4 insertions(+), 5 deletions(-)

```
diff --git a/include/linux/pid.h b/include/linux/pid.h
index 145dce7..e29a900 100644
--- a/include/linux/pid.h
+++ b/include/linux/pid.h
@@ -119,7 +119,7 @@ extern struct pid *find_pid(int nr);
 extern struct pid *find_get_pid(int nr);
 extern struct pid *find_ge_pid(int nr, struct pid_namespace *);

-extern struct pid *alloc_pid(struct task_struct *task);
+extern struct pid *alloc_pid(struct pid_namespace *ns);
 extern void FASTCALL(free_pid(struct pid *pid));
 extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);

diff --git a/kernel/fork.c b/kernel/fork.c
index ac73f3e..c1d4672 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1154,7 +1154,7 @@ static struct task_struct *copy_process(struct task_struct *task,

  if (pid != &init_struct_pid) {
   retval = -ENOMEM;
-  pid = alloc_pid(task);
+  pid = alloc_pid(task_active_pid_ns(p));
   if (!pid)
    goto bad_fork_cleanup_namespaces;
```

```
diff --git a/kernel/pid.c b/kernel/pid.c
index b887a6a..d7388d7 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
@@ -238,15 +238,14 @@ fastcall void free_pid(struct pid *pid)
  call_rcu(&pid->rcu, delayed_put_pid);
 }

-struct pid *alloc_pid(struct task_struct *srctsk)
+struct pid *alloc_pid(struct pid_namespace *ns)
 {
  struct pid *pid;
  enum pid_type type;
  int i, nr;
- struct pid_namespace *tmp, *ns;
+ struct pid_namespace *tmp;
  struct upid *upid;

- ns = task_active_pid_ns(srctsk);
  pid = kmem_cache_alloc(ns->pid_cachep, GFP_KERNEL);
  if (!pid)
   goto out;
--
1.5.1
```

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