Subject: Re: [PATCH 8/16] Masquerade the siginfo when sending a pid to a foreign namespace
Posted by Sukadev Bhattiprolu on Tue, 10 Jul 2007 04:18:00 GMT
View Forum Message <> Reply to Message

Pavel Emelianov [xemul@openvz.org] wrote:
| When user send signal from (say) init namespace to any task in a sub
| namespace the siginfo struct must not carry the sender's pid value, as
| this value may refer to some task in the destination namespace and thus
| may confuse the application.

Also, do you prevent signals to the child reaper of a container from within its container ? If so, can you show me where you handle it ? I can't seem to find it.

And I guess you do allow signals to the child-reaper of a container from its parent container.

|
| The consensus was to pretend in this case as if it is the kernel who
| sends the signal.
|
| The pid_ns_accessible() call is introduced to check this pid-to-ns
| accessibility.
|
| Signed-off-by: Pavel Emelianov <xemul@openvz.org>
|
| ---
|
| include/linux/pid.h | 10 ++++++++++
| kernel/signal.c | 34 ++++++++++++++++++++++++++++++++------
| 2 files changed, 38 insertions(+), 6 deletions(-)
|
| diff -upr linux-2.6.22-rc4-mm2.orig/include/linux/pid.h linux-2.6.22-rc4-mm2-2/include/linux/pid.h
| --- linux-2.6.22-rc4-mm2.orig/include/linux/pid.h 2007-06-14 12:14:29.000000000 +0400
| +++ linux-2.6.22-rc4-mm2-2/include/linux/pid.h 2007-07-04 19:00:38.000000000 +0400
| @@ -83,6 +89,16 @@ extern void FASTCALL(detach_pid(struct t
| return nr;
| }
|
| +/*
| + * checks whether the pid actually lives in the namespace ns, i.e. it was
| + * created in this namespace or it was moved there.
| + */
| +
| +static inline int pid_ns_accessible(struct pid_namespace *ns, struct pid *pid)
| +{
| + return pid->numbers[pid->level].ns == ns;

```
| +}
| +
|  #define do_each_pid_task(pid, type, task)    \
|   do {        \
|    struct hlist_node *pos___;    \
| diff -upr linux-2.6.22-rc4-mm2.orig/kernel/signal.c linux-2.6.22-rc4-mm2-2/kernel/signal.c
| --- linux-2.6.22-rc4-mm2.orig/kernel/signal.c 2007-07-04 19:00:38.000000000 +0400
| +++ linux-2.6.22-rc4-mm2-2/kernel/signal.c 2007-07-04 19:00:38.000000000 +0400
| @@ -1124,13 +1124,31 @@ EXPORT_SYMBOL_GPL(kill_pid_info_as_uid);
|   * is probably wrong.  Should make it like BSD or SYSV.
|   */
|
| -static int kill_something_info(int sig, struct siginfo *info, int pid)
| +static inline void masquerade_siginfo(struct pid_namespace *src_ns,
| +  struct pid *tgt_pid, struct siginfo *info)
| +{
| + if (tgt_pid != NULL && !pid_ns_accessible(src_ns, tgt_pid)) {
| +  /*
| +   * current namespace is not seen from the taks we
| +   * want to send the signal to, so pretend as if it
| +   * is the kernel who does this to avoid pid messing
| +   * by the target
| +   */
| +
| +  info->si_pid = 0;
| +  info->si_code = SI_KERNEL;
| + }
| +}
| +
| +static int kill_something_info(int sig, struct siginfo *info, int pid_nr)
|  {
|   int ret;
| + struct pid *pid;
| +
|   rcu_read_lock();
| - if (!pid) {
| + if (!pid_nr) {
|    ret = kill_pgrp_info(sig, info, task_pgrp(current));
| - } else if (pid == -1) {
| + } else if (pid_nr == -1) {
|    int retval = 0, count = 0;
|    struct task_struct * p;
```

So what happens if we run "kill -s <sig> -1" from within a container ?
Do you terminate all processes in the system or just the process in
the container ?

_____
Containers mailing list