
Subject: [PATCH] signal: Introduce kill_pid_ns_info
Posted by [Daniel Hokka Zakrisso](#) on Thu, 17 Jul 2008 21:54:50 GMT
[View Forum Message](#) <> [Reply to Message](#)

I'm sorry Andrew, this should look better.

Author: Eric W. Biederman <ebiederm@xmission.com>

Implement the basic helper function that walks all of the processes in a pid namespace and sends them all a signal.

Both locations that could use this functions are also updated to use this function.

I use find_ge_pid instead of for_each_process because it has a chance of not touching every process in the system.

[daniel@hozac.com: Optimize away nr <= 1 check, against latest Linus tree]
Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
Signed-off-by: Daniel Hokka Zakrisson <daniel@hozac.com>

```
include/linux/sched.h | 2 +
kernel/pid_namespace.c | 17 +-----
kernel/signal.c       | 52 +-----
3 files changed, 43 insertions(+), 28 deletions(-)
```

```
diff --git a/include/linux/sched.h b/include/linux/sched.h
```

```
index ba2f859..83597f8 100644
```

```
--- a/include/linux/sched.h
```

```
+++ b/include/linux/sched.h
```

```
@@ -1778,6 +1778,8 @@ extern void release_task(struct task_struct * p);
```

```
extern int send_sig_info(int, struct siginfo *, struct task_struct *);
```

```
extern int force_sigsegv(int, struct task_struct *);
```

```
extern int force_sig_info(int, struct siginfo *, struct task_struct *);
```

```
+extern int __kill_pid_ns_info(int sig, struct siginfo *info, struct pid_namespace *ns);
```

```
+extern int kill_pid_ns_info(int sig, struct siginfo *info, struct pid_namespace *ns);
```

```
extern int __kill_pgrp_info(int sig, struct siginfo *info, struct pid *pgrp);
```

```
extern int kill_pid_info(int sig, struct siginfo *info, struct pid *pid);
```

```
extern int kill_pid_info_as_uid(int, struct siginfo *, struct pid *, uid_t, uid_t, u32);
```

```
diff --git a/kernel/pid_namespace.c b/kernel/pid_namespace.c
```

```
index 98702b4..9226423 100644
```

```
--- a/kernel/pid_namespace.c
```

```
+++ b/kernel/pid_namespace.c
```

```
@@ -153,29 +153,14 @@ void free_pid_ns(struct kref *kref)
```

```
void zap_pid_ns_processes(struct pid_namespace *pid_ns)
```

```
{
```

```
- int nr;
```

```

int rc;

/*
 * The last thread in the cgroup-init thread group is terminating.
 * Find remaining pid_ts in the namespace, signal and wait for them
 * to exit.
- *
- * Note: This signals each threads in the namespace - even those that
- * belong to the same thread group, To avoid this, we would have
- * to walk the entire tasklist looking a processes in this
- * namespace, but that could be unnecessarily expensive if the
- * pid namespace has just a few processes. Or we need to
- * maintain a tasklist for each pid namespace.
- *
 */
- read_lock(&tasklist_lock);
- nr = next_pidmap(pid_ns, 1);
- while (nr > 0) {
- kill_proc_info(SIGKILL, SEND_SIG_PRIV, nr);
- nr = next_pidmap(pid_ns, nr);
- }
- read_unlock(&tasklist_lock);
+ kill_pid_ns_info(SIGKILL, SEND_SIG_PRIV, pid_ns);

do {
clear_thread_flag(TIF_SIGPENDING);
diff --git a/kernel/signal.c b/kernel/signal.c
index 6c0958e..96d469a 100644
--- a/kernel/signal.c
+++ b/kernel/signal.c
@@ -1118,6 +1118,45 @@ out_unlock:
}
EXPORT_SYMBOL_GPL(kill_pid_info_as_uid);

+int __kill_pid_ns_info(int sig, struct siginfo *info, struct pid_namespace *ns)
+{
+ int retval = 0, count = 0;
+ struct task_struct *p;
+ struct pid *pid;
+ int nr;
+
+ /* Since there isn't a pid namespace list of tasks use the closest
+ * approximation we have: find_ge_pid.
+ */
+ nr = 1;
+ while ((pid = find_ge_pid(nr + 1, ns))) {
+ int err;
+

```

```

+ nr = pid_nr_ns(pid, ns);
+ p = pid_task(pid, PIDTYPE_PID);
+ if (!p || !thread_group_leader(p) ||
+     same_thread_group(p, current))
+     continue;
+
+ err = group_send_sig_info(sig, info, p);
+ ++count;
+ if (err != -EPERM)
+     retval = err;
+ }
+ return count ? retval : -ESRCH;
+}
+
+int kill_pid_ns_info(int sig, struct siginfo *info, struct pid_namespace *ns)
+{
+ int retval;
+
+ read_lock(&tasklist_lock);
+ retval = __kill_pid_ns_info(sig, info, ns);
+ read_unlock(&tasklist_lock);
+
+ return retval;
+}
+
+/*
+ * kill_something_info() interprets pid in interesting ways just like kill(2).
+ */
@@ -1141,18 +1180,7 @@ static int kill_something_info(int sig, struct siginfo *info, int pid)
    ret = __kill_pgrp_info(sig, info,
        pid ? find_vpid(-pid) : task_pgrp(current));
    } else {
- int retval = 0, count = 0;
- struct task_struct * p;
-
- for_each_process(p) {
- if (p->pid > 1 && !same_thread_group(p, current)) {
- int err = group_send_sig_info(sig, info, p);
- ++count;
- if (err != -EPERM)
-     retval = err;
- }
- }
- ret = count ? retval : -ESRCH;
+ ret = __kill_pid_ns_info(sig, info, task_active_pid_ns(current));
    }
    read_unlock(&tasklist_lock);

```

--

1.5.5.1

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
