
Subject: [PATCH] Masquerade sender information
Posted by [Sukadev Bhattiprolu](#) on Sat, 27 Oct 2007 19:02:16 GMT
[View Forum Message](#) <> [Reply to Message](#)

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [PATCH] Masquerade sender information

With multiple pid namespaces, sender of a signal could be in an ancestor namespace of the receiver and so the sender will not have a valid 'pid_t' in the receiver's namespace.

In this case, masquerade the 'siginfo' for the signal to pretend that the signal originated from the kernel.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

include/linux/pid.h | 8 +++++++
include/linux/pid_namespace.h | 1 +
kernel/pid.c | 13 ++++++++
kernel/signal.c | 15 ++++++++
4 files changed, 37 insertions(+)

Index: 2.6.23-mm1/include/linux/pid.h

```
=====
--- 2.6.23-mm1.orig/include/linux/pid.h 2007-10-27 08:48:56.000000000 -0700
+++ 2.6.23-mm1/include/linux/pid.h 2007-10-27 09:56:53.000000000 -0700
@@ -123,6 +123,14 @@ extern struct pid *alloc_pid(struct pid_
extern void FASTCALL(free_pid(struct pid *pid));
extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);

+static inline struct pid_namespace *pid_active_ns(struct pid *pid)
+{
+    if (!pid)
+        return NULL;
+
+    return pid->numbers[pid->level].ns;
+}
+
+/*
+ * the helpers to get the pid's id seen from different namespaces
+ */
```

Index: 2.6.23-mm1/kernel/pid.c

```
=====
--- 2.6.23-mm1.orig/kernel/pid.c 2007-10-27 08:50:51.000000000 -0700
+++ 2.6.23-mm1/kernel/pid.c 2007-10-27 10:03:28.000000000 -0700
@@ -430,6 +430,19 @@ struct pid *find_get_pid(pid_t nr)
    return pid;
}
```

```

+/*
+ * Return TRUE if the active pid namespace of @tsk is same as active
+ * pid namespace of 'current'.
+ *
+ * Note the difference between this and the task_in_pid_ns() below.
+ * task_in_pid_ns() includes processes in descendant pid name spaces
+ * but pid_ns_equal() only matches _active_ pid namespaces.
+ */
+int pid_ns_equal(struct task_struct *tsk)
+{
+ return pid_active_ns(task_pid(current)) == pid_active_ns(task_pid(tsk));
+}
+
+ static int pid_in_pid_ns(struct pid *pid, struct pid_namespace *ns)
+ {
+ return pid && (ns->level <= pid->level) &&
Index: 2.6.23-mm1/kernel/signal.c

```

```

=====
--- 2.6.23-mm1.orig/kernel/signal.c 2007-10-27 08:50:51.000000000 -0700
+++ 2.6.23-mm1/kernel/signal.c 2007-10-27 10:02:04.000000000 -0700
@@ -679,6 +679,20 @@ static void handle_stop_signal(int sig,
 }
}

```

```

+static void masquerade_sender(struct task_struct *t, struct sigqueue *q)
+{
+ /*
+ * If the sender does not have a pid_t in the receiver's active
+ * pid namespace, set si_pid to 0 and pretend signal originated
+ * from the kernel.
+ */
+ if (!pid_ns_equal(t)) {
+ q->info.si_pid = 0;
+ q->info.si_uid = 0;
+ q->info.si_code = SI_KERNEL;
+ }
+}
+
+ static int send_signal(int sig, struct siginfo *info, struct task_struct *t,
+ struct sigpending *signals)
+ {
@@ -730,6 +744,7 @@ static int send_signal(int sig, struct s
+ copy_siginfo(&q->info, info);
+ break;
+ }
+ masquerade_sender(t, q);
+ } else if (!is_si_special(info)) {

```

```
if (sig >= SIGRTMIN && info->si_code != SI_USER)
```

```
/*
```

```
Index: 2.6.23-mm1/include/linux/pid_namespace.h
```

```
=====  
--- 2.6.23-mm1.orig/include/linux/pid_namespace.h 2007-10-27 09:44:25.000000000 -0700
```

```
+++ 2.6.23-mm1/include/linux/pid_namespace.h 2007-10-27 10:04:20.000000000 -0700
```

```
@@ -56,6 +56,7 @@ static inline struct task_struct *task_c
```

```
return tsk->nsproxy->pid_ns->child_reaper;
```

```
}
```

```
+extern int pid_ns_equal(struct task_struct *tsk);
```

```
extern int task_in_pid_ns(struct task_struct *tsk, struct pid_namespace *ns);
```

```
#endif /* _LINUX_PID_NS_H */
```

Containers mailing list

Containers@lists.linux-foundation.org

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