Subject: [RFC] [PATCH 5/8] userns: handle user namespaces in file sigio
Posted by serue on Fri, 07 Dec 2007 19:14:44 GMT
View Forum Message <> Reply to Message

>From 649ee38555a07f5ce052a8483a5f271fc31054d3 Mon Sep 17 00:00:00 2001
From: sergeh@us.ibm.com <sergeh@us.ibm.com>
Date: Tue, 4 Dec 2007 15:37:48 -0800
Subject: [RFC] [PATCH 5/8] userns: handle user namespaces in file sigio

Store user namespaces in fown_struct. _f_modown() sets the
user_ns to current's, or to NULL if current has
capable(CAP_NS_OVERRIDE).

Only allow a signal to be sent through sigio if the recipient
is in the same user namespace or the sender (meaning the user
who did the F_SETOWN, not the one triggering io) has
CAP_NS_OVERRIDE.

Test as follows:

1. log in as user hallyn in two windows
2. in window 1, run vim
3. in window 2, get pid for vim, i.e. PID=`pidof vim`
4. mkfifo ab
5. set_sigio ab $PID

If both logins are in same userns, the vim is killed.
if the logins are in separate user namespaces, vim is not
killed.

****
set_sigio.c
****

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#define __USE_GNU
#include <fcntl.h>

int main(int argc, char *argv[])
{
 int err;
 int pid;
 int fd = open(argv[1], O_RDWR);

 if (!fd) {
```

```
  perror("open");
  exit(1);
 }
 printf("asked to set owner to %s\n", argv[2]);
 pid = atoi(argv[2]);
 printf("setting owner to %d\n", pid);
 err = fcntl(fd, F_SETOWN, pid);
 if (err == -1) {
  perror("fcntl 1\n");
  exit(1);
 }
 err = fcntl(fd, F_SETSIG, 9);
 if (err == -1) {
  perror("fcntl 3\n");
  exit(1);
 }
 err = fcntl(fd, F_SETFL, O_ASYNC);
 if (err == -1) {
  perror("fcntl 2\n");
  exit(1);
 }

 write(fd, "ab", 2);

 close(fd);
 printf("done\n");
}
```

```
***************************************************************

Signed-off-by: Serge Hallyn <serue@us.ibm.com>
---
 fs/fcntl.c         |   53 ++++++++++++++++++++++++++++++++++++++++++++++++++-----
 include/linux/fs.h |    2 +-
 2 files changed, 48 insertions(+), 7 deletions(-)

diff --git a/fs/fcntl.c b/fs/fcntl.c
index 8685263..124408a 100644
--- a/fs/fcntl.c
+++ b/fs/fcntl.c
@@ -19,6 +19,7 @@
 #include <linux/signal.h>
 #include <linux/rcupdate.h>
 #include <linux/pid_namespace.h>
+#include <linux/user_namespace.h>

 #include <asm/poll.h>
 #include <asm/siginfo.h>
@@ -259,10 +260,23 @@ static void f_modown(struct file *filp, struct pid *pid, enum pid_type
```

```
type,
  write_lock_irq(&filp->f_owner.lock);
  if (force || !filp->f_owner.pid) {
   put_pid(filp->f_owner.pid);
+  put_user_ns(filp->f_owner.uid.ns);
   filp->f_owner.pid = get_pid(pid);
   filp->f_owner.pid_type = type;
-  filp->f_owner.uid = uid;
-  filp->f_owner.euid = euid;
+  filp->f_owner.uid.uid = uid;
+  filp->f_owner.euid.uid = euid;
+  if (pid) {
+   if (capable(CAP_NS_OVERRIDE) && capable(CAP_KILL)) {
+    filp->f_owner.uid.ns = NULL;
+    filp->f_owner.euid.ns = NULL;
+   } else {
+    filp->f_owner.uid.ns =
+      current->nsproxy->user_ns;
+    filp->f_owner.euid.ns =
+      current->nsproxy->user_ns;
+    get_user_ns(filp->f_owner.uid.ns);
+   }
+  }
  }
  write_unlock_irq(&filp->f_owner.lock);
 }
@@ -457,13 +471,40 @@ static const long band_table[NSIGPOLL] = {
  POLLHUP | POLLERR   /* POLL_HUP */
 };

+static inline int sigio_userns_check_ok(struct task_struct *p,
+     struct fown_struct *fown)
+{
+ if (!fown->euid.ns)
+  return 1;
+ if (p->nsproxy->user_ns == fown->euid.ns)
+  return 1;
+ return 0;
+}
+
+static inline int sigio_uids_check_ok(struct task_struct *p,
+          struct fown_struct *fown)
+{
+ return ((fown->euid.uid == 0) ||
+  (fown->euid.uid == p->suid) || (fown->euid.uid == p->uid) ||
+  (fown->uid.uid == p->suid) || (fown->uid.uid == p->uid));
+}
+
```

```
+#define lsm_sigiotask_ok(p, fown, sig) \
+ (!security_file_send_sigiotask(p, fown, sig))
+
 static inline int sigio_perm(struct task_struct *p,
                              struct fown_struct *fown, int sig)
 {
- return (((fown->euid == 0) ||
-   (fown->euid == p->suid) || (fown->euid == p->uid) ||
-   (fown->uid == p->suid) || (fown->uid == p->uid)) &&
-  !security_file_send_sigiotask(p, fown, sig));
+ if (!sigio_userns_check_ok(p, fown))
+  return 0;
+
+ if (!sigio_uids_check_ok(p, fown))
+  return 0;
+
+ if (!lsm_sigiotask_ok(p, fown, sig))
+  return 0;
+
+ return 1;
 }

 static void send_sigio_to_task(struct task_struct *p,
diff --git a/include/linux/fs.h b/include/linux/fs.h
index 8323ebf..cb3894d 100644
--- a/include/linux/fs.h
+++ b/include/linux/fs.h
@@ -747,7 +747,7 @@ struct fown_struct {
  rwlock_t lock;          /* protects pid, uid, euid fields */
  struct pid *pid; /* pid or -pgrp where SIGIO should be sent */
  enum pid_type pid_type; /* Kind of process group SIGIO should be sent to */
- uid_t uid, euid; /* uid/euid of process setting the owner */
+ struct k_uid_t uid, euid; /* uid/euid of process setting the owner */
  int signum;  /* posix.1b rt signal to be delivered on IO */
 };

--
1.5.1
```