

---

Subject: [PATCH 7/8] user ns: handle file sigio  
Posted by [serue](#) on Tue, 19 Dec 2006 23:01:27 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

From: Serge E. Hallyn <[serue@us.ibm.com](mailto:serue@us.ibm.com)>  
Subject: [PATCH 7/8] user ns: handle file sigio

A process in one user namespace could set a fowner and sigio on a file in a shared vfsmount, ending up killing a task in another user namespace.

Prevent this by adding a user namespace pointer to the fown\_struct, and enforcing that a process causing a signal to be sent be in the same user namespace as the file owner.

Signed-off-by: Serge E. Hallyn <[serue@us.ibm.com](mailto:serue@us.ibm.com)>

---

```
fs/fcntl.c      | 14 ++++++++
fs/file_table.c |  2 ++
include/linux/fs.h |  1 +
3 files changed, 14 insertions(+), 3 deletions(-)
```

```
diff --git a/fs/fcntl.c b/fs/fcntl.c
index 8e382a5..6a774c1 100644
```

```
--- a/fs/fcntl.c
+++ b/fs/fcntl.c
@@ -18,6 +18,7 @@ #include <linux/security.h>
#include <linux/ptrace.h>
#include <linux/signal.h>
#include <linux/rcupdate.h>
+#include <linux/user_namespace.h>

#include <asm/poll.h>
#include <asm/siginfo.h>
@@ -250,15 +251,18 @@ static int setfl(int fd, struct file * f
}
```

```
static void f_modown(struct file *filp, struct pid *pid, enum pid_type type,
-      uid_t uid, uid_t euid, int force)
+      uid_t uid, uid_t euid, struct user_namespace *user_ns,
+      int force)
{
    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.user_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.user_ns = get_user_ns(user_ns);
}
write_unlock_irq(&filp->f_owner.lock);
}
@@ -272,7 +276,8 @@ int __f_setown(struct file *filp, struct
if (err)
    return err;

- f_modown(filp, pid, type, current->uid, current->euid, force);
+ f_modown(filp, pid, type, current->uid, current->euid,
+   current->nsproxy->user_ns, force);
    return 0;
}
EXPORT_SYMBOL(__f_setown);
@@ -298,7 +303,7 @@ EXPORT_SYMBOL(f_setown);

void f_delown(struct file *filp)
{
- f_modown(filp, NULL, PIDTYPE_PID, 0, 0, 1);
+ f_modown(filp, NULL, PIDTYPE_PID, 0, 0, NULL, 1);
}

pid_t f_getown(struct file *filp)
@@ -455,6 +460,9 @@ static const long band_table[NSIGPOLL] =
static inline int sigio_perm(struct task_struct *p,
                             struct fown_struct *fown, int sig)
{
+ if (fown->user_ns != init_task.nsproxy->user_ns &&
+   fown->user_ns != p->nsproxy->user_ns)
+ return 0;
    return (((fown->euid == 0) ||
        (fown->euid == p->suid) || (fown->euid == p->uid) ||
        (fown->uid == p->suid) || (fown->uid == p->uid)) &&
diff --git a/fs/file_table.c b/fs/file_table.c
index 4c17a18..6e6632c 100644
--- a/fs/file_table.c
+++ b/fs/file_table.c
@@ -21,6 +21,7 @@ #include <linux/cdev.h>
#include <linux/fsnotify.h>
#include <linux/sysctl.h>
#include <linux/percpu_counter.h>
+#include <linux/user_namespace.h>

#include <asm/atomic.h>

@@ -175,6 +176,7 @@ void fastcall __fput(struct file *file)
if (file->f_mode & FMODE_WRITE)

```

```
    put_write_access(inode);
    put_pid(file->f_owner.pid);
+ put_user_ns(file->f_owner.user_ns);
    file_kill(file);
    file->f_path.dentry = NULL;
    file->f_path.mnt = NULL;
diff --git a/include/linux/fs.h b/include/linux/fs.h
index bb801cb..a5532e1 100644
--- a/include/linux/fs.h
+++ b/include/linux/fs.h
@@ -684,6 +684,7 @@ struct fown_struct {
    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 user_namespace *user_ns; /* namespace to which uid belongs */
    int signum; /* posix.1b rt signal to be delivered on IO */
};

--
1.4.1
```

---

Containers mailing list  
Containers@lists.osdl.org  
<https://lists.osdl.org/mailman/listinfo/containers>

---