Subject: [PATCH 0/3] uidns: various patches
Posted by serue on Wed, 06 Dec 2006 23:17:43 GMT
View Forum Message <> Reply to Message

Hi,

here is a set of 3 patches on top of the patchset Cedric sent out
yesterday, amending the user namespace a bit.

Patch 1: trivial rename and commenting of the tsk_mnt_same_uid helper.

Patch 2: bugfix at clone_mnt, and updates to get/put_user_ns helpers

Patch 3: implement user namespace equivalence check for file sigio.

thanks,
-serge

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

Subject: [PATCH 1/3] uidns: improve name of tsk_mnt_same_uid
Posted by serue on Wed, 06 Dec 2006 23:18:51 GMT
View Forum Message <> Reply to Message

From: Serge E. Hallyn <serue@us.ibm.com>
Subject: [PATCH 1/3] uidns: improve name of tsk_mnt_same_uid

The helper compares uidns, not uid, so change the name to
tsk_mnt_same_uidns.

Also comment the behavior above the helper.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
---
 fs/namei.c | 4 ++--
 include/linux/sched.h | 9 ++++++++-
 2 files changed, 10 insertions(+), 3 deletions(-)

diff --git a/fs/namei.c b/fs/namei.c
index ab59efe..6c207f1 100644
--- a/fs/namei.c
+++ b/fs/namei.c
@@ -246,7 +246,7 @@ int permission(struct inode *inode, int
    return -EACCES;
  }

```
- if (nd && !task_mnt_same_uid(current, nd->mnt))
+ if (nd && !task_mnt_same_uidns(current, nd->mnt))
   return -EACCES;

  /*
@@ -435,7 +435,7 @@ static int exec_permission_lite(struct i
 {
  umode_t mode = inode->i_mode;

- if (!task_mnt_same_uid(current, nd->mnt))
+ if (!task_mnt_same_uidns(current, nd->mnt))
   return -EACCES;
  if (inode->i_op && inode->i_op->permission)
   return -EAGAIN;
diff --git a/include/linux/sched.h b/include/linux/sched.h
index cd31763..a4b5c77 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -1591,7 +1591,14 @@ extern int cond_resched(void);
 extern int cond_resched_lock(spinlock_t * lock);
 extern int cond_resched_softirq(void);

-static inline int task_mnt_same_uid(struct task_struct *tsk,
+/*
+ * Check whether a task and a vfsmnt belong to the same uidns.
+ * Since the initial namespace is exempt from these checks,
+ * return 1 if so.  Also return 1 if the vfsmnt is exempt from
+ * such checking.  Otherwise, if the uid namespaces are different,
+ * return 0.
+ */
+static inline int task_mnt_same_uidns(struct task_struct *tsk,
       struct vfsmount *mnt)
 {
  if (tsk->nsproxy == init_task.nsproxy)
--
1.4.1
```

Subject: [PATCH 2/3] user_ns: bugfix and cleanup
Posted by serue on Wed, 06 Dec 2006 23:19:04 GMT
View Forum Message <> Reply to Message

From: Serge E. Hallyn <serue@us.ibm.com>
Subject: [PATCH 2/3] user_ns: bugfix and cleanup

The first two updates are used in the next patchset:

1. change get_user_ns to return the namespace
2. change get_user_ns and put_user_ns to handle NULL input

This fixes a bug in the lxc patchset:

3. when clone_mnt uses the user_ns from the original vfsmnt (for
   MNT_PRIV_USERNS mounts), it needs to put the user_ns which was
   gotten in alloc_mnt, and get the new one.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
---
 fs/namespace.c | 9 +++++----
 include/linux/user_namespace.h | 11 +++++++----
 2 files changed, 12 insertions(+), 8 deletions(-)

diff --git a/fs/namespace.c b/fs/namespace.c
index 55fcaa7..fa52e24 100644
--- a/fs/namespace.c
+++ b/fs/namespace.c
@@ -57,8 +57,7 @@ struct vfsmount *alloc_vfsmnt(const char
  struct vfsmount *mnt = kmem_cache_alloc(mnt_cache, GFP_KERNEL);
  if (mnt) {
   memset(mnt, 0, sizeof(struct vfsmount));
- mnt->mnt_user_ns = current->nsproxy->user_ns;
- get_user_ns(mnt->mnt_user_ns);
+ mnt->mnt_user_ns = get_user_ns(current->nsproxy->user_ns);
   atomic_set(&mnt->mnt_count, 1);
   atomic_set(&mnt->mnt_writers, 0);
   INIT_LIST_HEAD(&mnt->mnt_hash);
@@ -261,8 +260,10 @@ static struct vfsmount *clone_mnt(struct

  if (mnt) {
   mnt->mnt_flags = old->mnt_flags;
- if (mnt->mnt_flags & MNT_PRIV_USERNS)
-   mnt->mnt_user_ns = old->mnt_user_ns;
+ if (mnt->mnt_flags & MNT_PRIV_USERNS) {
+   put_user_ns(mnt->mnt_user_ns);
+   mnt->mnt_user_ns = get_user_ns(old->mnt_user_ns);
+ }
   atomic_inc(&sb->s_active);
   mnt->mnt_sb = sb;
   mnt->mnt_root = dget(root);
diff --git a/include/linux/user_namespace.h b/include/linux/user_namespace.h

```
index f25c54c..f9d76de 100644
--- a/include/linux/user_namespace.h
+++ b/include/linux/user_namespace.h
@@ -17,9 +17,11 @@ extern struct user_namespace init_user_n

 #ifdef CONFIG_USER_NS

-static inline void get_user_ns(struct user_namespace *ns)
+static inline struct user_namespace *get_user_ns(struct user_namespace *ns)
 {
- kref_get(&ns->kref);
+ if (ns)
+  kref_get(&ns->kref);
+ return ns;
 }

 extern int unshare_user_ns(unsigned long unshare_flags,
@@ -29,12 +31,13 @@ extern void free_user_ns(struct kref *kr

 static inline void put_user_ns(struct user_namespace *ns)
 {
- kref_put(&ns->kref, free_user_ns);
+ if (ns)
+  kref_put(&ns->kref, free_user_ns);
 }

 #else

-static inline void get_user_ns(struct user_namespace *ns)
+static inline struct user_namespace *get_user_ns(struct user_namespace *ns)
 {
 }

--
1.4.1
```


Subject: [PATCH 3/3] user_ns: handle file sigio
Posted by serue on Wed, 06 Dec 2006 23:19:15 GMT
View Forum Message <> Reply to Message

From: Serge E. Hallyn <serue@us.ibm.com>
Subject: [PATCH 3/3] user_ns: handle file sigio

Close a hole where 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.

Signed-off-by: Serge E. Hallyn <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 4d2de47..725b0fe 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 454fa4b..818b735 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>

@@ -212,6 +213,7 @@ void fastcall __fput(struct file *file)
     mnt_drop_write(mnt);
   }
   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 eeda2ee..9c95d36 100644
--- a/include/linux/fs.h
```

```
+++ b/include/linux/fs.h
@@ -688,6 +688,7 @@ 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
```

Subject: Re: [PATCH 2/3] user_ns: bugfix and cleanup
Posted by serue on Mon, 11 Dec 2006 21:40:42 GMT
View Forum Message <> Reply to Message

This patch needs the following fixup:

From: Serge E. Hallyn <serue@us.ibm.com>
Subject: [PATCH 1/1] userns: fix compile warning if !CONFIG_USER_NS

previous patch updated get_user_ns() to return a struct
user_namespace, but the empty default for !CONFIG_USER_NS
did not return anything.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
---
 include/linux/user_namespace.h | 1 +
 1 files changed, 1 insertions(+), 0 deletions(-)

```
diff --git a/include/linux/user_namespace.h b/include/linux/user_namespace.h
index f9d76de..53ad920 100644
--- a/include/linux/user_namespace.h
+++ b/include/linux/user_namespace.h
@@ -39,6 +39,7 @@ #else

 static inline struct user_namespace *get_user_ns(struct user_namespace *ns)
 {
+ return NULL;
 }

 static inline int unshare_user_ns(unsigned long unshare_flags,
```

--
1.4.1

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