Subject: [PATCH 3/6] user ns: add an inode user_ns pointer
Posted by serue on Mon, 04 Jun 2007 19:41:05 GMT
View Forum Message <> Reply to Message

>From nobody Mon Sep 17 00:00:00 2001
From: Serge E. Hallyn <serue@us.ibm.com>
Date: Thu, 5 Apr 2007 14:02:09 -0400
Subject: [PATCH 3/6] user ns: add an inode user_ns pointer

Add a user namespace pointer to each inode. One user namespace is said
to own each inode. Each filesystem can fill these in for itself. By
default it is NULL, meaning that user namespace checks will be skipped.
At first most filesystems will presumably assign a user namespace to the
fs superblock info, and assign that to each inode on the fs.

If this seems limiting based on this description, see the long-ish
development roadmap to see why it is really the best way to go, and in
no way limiting.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>

```
 fs/inode.c                  |    2 ++
 include/linux/fs.h          |    3 +++
 include/linux/user_namespace.h |   25 ++++++++++++++++++++++++
 kernel/user_namespace.c     |   18 ++++++++++++++++++
 4 files changed, 48 insertions(+), 0 deletions(-)

c76b1d886c98c9661e794fa7f58dbaa4f4613864
diff --git a/fs/inode.c b/fs/inode.c
index 42cb8e5..2eab0f1 100644
--- a/fs/inode.c
+++ b/fs/inode.c
@@ -23,6 +23,7 @@ #include <linux/cdev.h>
 #include <linux/bootmem.h>
 #include <linux/inotify.h>
 #include <linux/mount.h>
+#include <linux/user_namespace.h>

 /*
  * This is needed for the following functions:
@@ -116,6 +117,7 @@ static struct inode *alloc_inode(struct
   struct address_space * const mapping = &inode->i_data;

   inode->i_sb = sb;
+  inode->i_userns = NULL;
   inode->i_blkbits = sb->s_blocksize_bits;
```

```
   inode->i_flags = 0;
   atomic_set(&inode->i_count, 1);
diff --git a/include/linux/fs.h b/include/linux/fs.h
index 2e46ad7..d021658 100644
--- a/include/linux/fs.h
+++ b/include/linux/fs.h
@@ -527,6 +527,8 @@ #else
 #define i_size_ordered_init(inode) do { } while (0)
 #endif

+struct user_namespace;
+
 struct inode {
  struct hlist_node i_hash;
  struct list_head i_list;
@@ -536,6 +538,7 @@ struct inode {
  atomic_t  i_count;
  unsigned int  i_nlink;
  uid_t   i_uid;
+ struct user_namespace *i_userns;
  gid_t   i_gid;
  dev_t   i_rdev;
  unsigned long  i_version;
diff --git a/include/linux/user_namespace.h b/include/linux/user_namespace.h
index 9044456..3b5e040 100644
--- a/include/linux/user_namespace.h
+++ b/include/linux/user_namespace.h
@@ -4,6 +4,7 @@ #define _LINUX_USER_NAMESPACE_H
 #include <linux/kref.h>
 #include <linux/nsproxy.h>
 #include <linux/sched.h>
+#include <linux/err.h>

 #define UIDHASH_BITS (CONFIG_BASE_SMALL ? 3 : 8)
 #define UIDHASH_SZ (1 << UIDHASH_BITS)
@@ -35,6 +36,25 @@ static inline void put_user_ns(struct us
   kref_put(&ns->kref, free_user_ns);
 }

+struct user_namespace *task_user_ns(struct task_struct *tsk);
+struct user_namespace *inode_user_ns(struct inode *inode);
+/*
+ * task_inode_same_userns:
+ * If inode->i_userns is NULL, the fs does not support user namespaces, so any
+ * user namespace may access it as though it owned it.  Otherwise, any user
+ * not in the owning user namespace will be treated as 'nobody'.
+ *
+ * once we implement a user namespace keychain, we will want to augment this
```

```
+ * to also check for the existance of (inode->i_userns, inode->i_uid) as a key
+ * in the keychain for the user running tsk.
+ */
+static inline int
+task_inode_same_userns(struct task_struct *tsk, struct inode *inode)
+{
+ struct user_namespace *ino_userns = inode_user_ns(inode);
+
+ return (!ino_userns || task_user_ns(tsk)== ino_userns);
+}
 #else

 static inline struct user_namespace *get_user_ns(struct user_namespace *ns)
@@ -55,6 +75,11 @@ static inline void put_user_ns(struct us
 {
 }

+static inline int
+task_inode_same_userns(struct task_struct *tsk, struct inode *inode)
+{
+ return 1;
+}
 #endif

 #endif /* _LINUX_USER_H */
diff --git a/kernel/user_namespace.c b/kernel/user_namespace.c
index 89a27e8..f487361 100644
--- a/kernel/user_namespace.c
+++ b/kernel/user_namespace.c
@@ -84,4 +84,22 @@ void free_user_ns(struct kref *kref)
  kfree(ns);
 }

+struct user_namespace *task_user_ns(struct task_struct *tsk)
+{
+ return tsk->nsproxy->user_ns;
+}
+
+struct user_namespace *get_task_user_ns(struct task_struct *tsk)
+{
+ return get_user_ns(task_user_ns(tsk));
+}
+
+/* should this just go into fs.h? */
+#include <linux/fs.h>
+struct user_namespace *inode_user_ns(struct inode *inode)
+{
+ return inode->i_userns;
```

```
+}
+
+
 #endif /* CONFIG_USER_NS */
--
1.3.2
```