Subject: [PATCH 5/6] userns strict: hook ext2
Posted by serue on Mon, 04 Jun 2007 19:42:00 GMT
View Forum Message <> Reply to Message

>From nobody Mon Sep 17 00:00:00 2001
From: Serge Hallyn <serue@us.ibm.com>
Date: Wed, 28 Mar 2007 15:06:47 -0500
Subject: [PATCH 5/6] userns strict: hook ext2

Add a user namespace pointer to the ext2 superblock and inode.

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

```
 fs/ext2/acl.c                 |    4 +++-
 fs/ext2/balloc.c              |   13 +++++++++++---
 fs/ext2/ialloc.c              |    5 +++++
 fs/ext2/inode.c               |    2 ++
 fs/ext2/ioctl.c               |   12 ++++++++----
 fs/ext2/super.c               |    4 ++++
 fs/ext2/xattr_trusted.c       |    3 ++-
 include/linux/ext2_fs_sb.h    |    1 +
 include/linux/user_namespace.h |   16 ++++++++++++++++
 9 files changed, 51 insertions(+), 9 deletions(-)

04b91190fbcaf388100bc748062724e384f1d868
diff --git a/fs/ext2/acl.c b/fs/ext2/acl.c
index 7c420b8..7bb1ccd 100644
--- a/fs/ext2/acl.c
+++ b/fs/ext2/acl.c
@@ -9,6 +9,7 @@ #include <linux/init.h>
 #include <linux/sched.h>
 #include <linux/slab.h>
 #include <linux/fs.h>
+#include <linux/user_namespace.h>
 #include "ext2.h"
 #include "xattr.h"
 #include "acl.h"
@@ -464,7 +465,8 @@ ext2_xattr_set_acl(struct inode *inode,

  if (!test_opt(inode->i_sb, POSIX_ACL))
   return -EOPNOTSUPP;
- if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))
+ if (!task_inode_same_fsuid(current, inode) &&
+   !task_ino_capable(inode, CAP_FOWNER))
   return -EPERM;
```

```
  if (value) {
diff --git a/fs/ext2/balloc.c b/fs/ext2/balloc.c
index 538c221..2924f0e 100644
--- a/fs/ext2/balloc.c
+++ b/fs/ext2/balloc.c
@@ -16,6 +16,7 @@ #include <linux/quotaops.h>
 #include <linux/sched.h>
 #include <linux/buffer_head.h>
 #include <linux/capability.h>
+#include <linux/user_namespace.h>

 /*
  * balloc.c contains the blocks allocation and deallocation routines
@@ -1127,9 +1128,15 @@ static int ext2_has_free_blocks(struct e

  free_blocks = percpu_counter_read_positive(&sbi->s_freeblocks_counter);
  root_blocks = le32_to_cpu(sbi->s_es->s_r_blocks_count);
- if (free_blocks < root_blocks + 1 && !capable(CAP_SYS_RESOURCE) &&
-  sbi->s_resuid != current->fsuid &&
-  (sbi->s_resgid == 0 || !in_group_p (sbi->s_resgid))) {
+ if (free_blocks < root_blocks + 1) {
+  if (sbi->s_resuidns != task_user_ns(current))
+   return 0;
+  if (capable(CAP_SYS_RESOURCE))
+   return 1;
+  if (sbi->s_resuid == current->fsuid)
+   return 1;
+  if (sbi->s_resgid != 0 && in_group_p (sbi->s_resgid))
+   return 1;
   return 0;
  }
  return 1;
diff --git a/fs/ext2/ialloc.c b/fs/ext2/ialloc.c
index 86a2f3b..34f16ab 100644
--- a/fs/ext2/ialloc.c
+++ b/fs/ext2/ialloc.c
@@ -17,6 +17,7 @@ #include <linux/sched.h>
 #include <linux/backing-dev.h>
 #include <linux/buffer_head.h>
 #include <linux/random.h>
+#include <linux/user_namespace.h>
 #include "ext2.h"
 #include "xattr.h"
 #include "acl.h"
@@ -133,6 +134,9 @@ void ext2_free_inode (struct inode * ino
  /* Do this BEFORE marking the inode not in use or returning an error */
  clear_inode (inode);
```

```
+ put_user_ns(inode->i_userns);
+ inode->i_userns = NULL;
+
  if (ino < EXT2_FIRST_INO(sb) ||
      ino > le32_to_cpu(es->s_inodes_count)) {
   ext2_error (sb, "ext2_free_inode",
@@ -563,6 +567,7 @@ got:
  sb->s_dirt = 1;
  mark_buffer_dirty(bh2);
  inode->i_uid = current->fsuid;
+ inode->i_userns = get_task_user_ns(current);
  if (test_opt (sb, GRPID))
   inode->i_gid = dir->i_gid;
  else if (dir->i_mode & S_ISGID) {
diff --git a/fs/ext2/inode.c b/fs/ext2/inode.c
index 5deb23b..dcca474 100644
--- a/fs/ext2/inode.c
+++ b/fs/ext2/inode.c
@@ -31,6 +31,7 @@ #include <linux/module.h>
 #include <linux/writeback.h>
 #include <linux/buffer_head.h>
 #include <linux/mpage.h>
+#include <linux/user_namespace.h>
 #include "ext2.h"
 #include "acl.h"
 #include "xip.h"
@@ -1168,6 +1169,7 @@ #endif
  inode->i_mode = le16_to_cpu(raw_inode->i_mode);
  inode->i_uid = (uid_t)le16_to_cpu(raw_inode->i_uid_low);
  inode->i_gid = (gid_t)le16_to_cpu(raw_inode->i_gid_low);
+ inode->i_userns = get_task_user_ns(current);
  if (!(test_opt (inode->i_sb, NO_UID32))) {
   inode->i_uid |= le16_to_cpu(raw_inode->i_uid_high) << 16;
   inode->i_gid |= le16_to_cpu(raw_inode->i_gid_high) << 16;
diff --git a/fs/ext2/ioctl.c b/fs/ext2/ioctl.c
index 315a98b..2326804 100644
--- a/fs/ext2/ioctl.c
+++ b/fs/ext2/ioctl.c
@@ -13,6 +13,7 @@ #include <linux/time.h>
 #include <linux/sched.h>
 #include <linux/compat.h>
 #include <linux/smp_lock.h>
+#include <linux/user_namespace.h>
 #include <asm/current.h>
 #include <asm/uaccess.h>

@@ -36,7 +37,8 @@ int ext2_ioctl (struct inode * inode, st
   if (IS_RDONLY(inode))
```

```
    return -EROFS;

-  if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))
+   if (!task_inode_same_fsuid(current, inode) &&
+    !task_ino_capable(inode, CAP_FOWNER))
    return -EACCES;

   if (get_user(flags, (int __user *) arg))
@@ -55,7 +57,7 @@ int ext2_ioctl (struct inode * inode, st
    * This test looks nicer. Thanks to Pauline Middelink
    */
   if ((flags ^ oldflags) & (EXT2_APPEND_FL | EXT2_IMMUTABLE_FL)) {
-   if (!capable(CAP_LINUX_IMMUTABLE)) {
+   if (!task_ino_capable(inode, CAP_LINUX_IMMUTABLE)) {
     mutex_unlock(&inode->i_mutex);
     return -EPERM;
    }
@@ -74,7 +76,8 @@ int ext2_ioctl (struct inode * inode, st
  case EXT2_IOC_GETVERSION:
   return put_user(inode->i_generation, (int __user *) arg);
  case EXT2_IOC_SETVERSION:
-  if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))
+  if (!task_inode_same_fsuid(current, inode) &&
+    !task_ino_capable(inode, CAP_FOWNER))
    return -EPERM;
   if (IS_RDONLY(inode))
    return -EROFS;
@@ -99,7 +102,8 @@ int ext2_ioctl (struct inode * inode, st
   if (IS_RDONLY(inode))
    return -EROFS;

-  if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))
+  if (!task_inode_same_fsuid(current, inode) &&
+    !task_ino_capable(inode, CAP_FOWNER))
    return -EACCES;

   if (get_user(rsv_window_size, (int __user *)arg))
diff --git a/fs/ext2/super.c b/fs/ext2/super.c
index 932579b..75ce9e8 100644
--- a/fs/ext2/super.c
+++ b/fs/ext2/super.c
@@ -29,6 +29,7 @@ #include <linux/smp_lock.h>
 #include <linux/vfs.h>
 #include <linux/seq_file.h>
 #include <linux/mount.h>
+#include <linux/user_namespace.h>
 #include <asm/uaccess.h>
 #include "ext2.h"
```

```
 #include "xattr.h"
@@ -125,6 +126,7 @@ static void ext2_put_super (struct super
    brelse (sbi->s_group_desc[i]);
  kfree(sbi->s_group_desc);
  kfree(sbi->s_debts);
+ put_user_ns(sbi->s_resuidns);
  percpu_counter_destroy(&sbi->s_freeblocks_counter);
  percpu_counter_destroy(&sbi->s_freeinodes_counter);
  percpu_counter_destroy(&sbi->s_dirs_counter);
@@ -742,6 +744,7 @@ #endif

  sbi->s_resuid = le16_to_cpu(es->s_def_resuid);
  sbi->s_resgid = le16_to_cpu(es->s_def_resgid);
+ sbi->s_resuidns = get_task_user_ns(current);

  set_opt(sbi->s_mount_opt, RESERVATION);

@@ -990,6 +993,7 @@ failed_mount_group_desc:
  kfree(sbi->s_group_desc);
  kfree(sbi->s_debts);
 failed_mount:
+ put_user_ns(sbi->s_resuidns);
  brelse(bh);
 failed_sbi:
  sb->s_fs_info = NULL;
diff --git a/fs/ext2/xattr_trusted.c b/fs/ext2/xattr_trusted.c
index f28a6a4..d24729a 100644
--- a/fs/ext2/xattr_trusted.c
+++ b/fs/ext2/xattr_trusted.c
@@ -11,6 +11,7 @@ #include <linux/capability.h>
 #include <linux/fs.h>
 #include <linux/smp_lock.h>
 #include <linux/ext2_fs.h>
+#include <linux/user_namespace.h>
 #include "xattr.h"

 #define XATTR_TRUSTED_PREFIX "trusted."
@@ -22,7 +23,7 @@ ext2_xattr_trusted_list(struct inode *in
  const int prefix_len = sizeof(XATTR_TRUSTED_PREFIX)-1;
  const size_t total_len = prefix_len + name_len + 1;

- if (!capable(CAP_SYS_ADMIN))
+ if (!task_ino_capable(inode, CAP_SYS_ADMIN))
   return 0;

  if (list && total_len <= list_size) {
diff --git a/include/linux/ext2_fs_sb.h b/include/linux/ext2_fs_sb.h
index b1b69bc..bdbc7cd 100644
```

```
--- a/include/linux/ext2_fs_sb.h
+++ b/include/linux/ext2_fs_sb.h
@@ -85,6 +85,7 @@ struct ext2_sb_info {
  unsigned long  s_mount_opt;
  uid_t s_resuid;
  gid_t s_resgid;
+ struct user_namespace *s_resuidns;
  unsigned short s_mount_state;
  unsigned short s_pad;
  int s_addr_per_block_bits;
diff --git a/include/linux/user_namespace.h b/include/linux/user_namespace.h
index 073f3e0..e87d6f5 100644
--- a/include/linux/user_namespace.h
+++ b/include/linux/user_namespace.h
@@ -27,6 +27,8 @@ static inline struct user_namespace *get
  return ns;
 }

+extern struct user_namespace *get_task_user_ns(struct task_struct *tsk);
+
 extern struct user_namespace *copy_user_ns(int flags,
        struct user_namespace *old_ns);
 extern void free_user_ns(struct kref *kref);
@@ -76,6 +78,10 @@ task_inode_same_fsuid(struct task_struct
  return 1;
 }

+/* the set of helpers is really getting out of hand.  consolidate
+ * in thr next release
+ */
+
 /*
  * task_ino_capable:
  * Again, when userns keys exist, we will need to check for a
@@ -95,6 +101,16 @@ static inline struct user_namespace *get
  return &init_user_ns;
 }

+static inline struct user_namespace *task_user_ns(struct task_struct *tsk)
+{
+ return &init_user_ns;
+}
+
+static inline struct user_namespace *get_task_user_ns(struct task_struct *tsk)
+{
+ return &init_user_ns;
+}
+
```

```
static inline struct user_namespace *copy_user_ns(int flags,
        struct user_namespace *old_ns)
{
--
1.3.2
```