Subject: [PATCH 2/9] namespaces: incorporate fs namespace into nsproxy
Posted by serue on Thu, 18 May 2006 15:49:01 GMT
View Forum Message <> Reply to Message

This moves the mount namespace into the nsproxy. The mount
namespace count now refers to the number of nsproxies point
to it, rather than the number of tasks. As a result, the
unshare_namespace() function in kernel/fork.c no longer checks
whether it is being shared.

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

```
 fs/namespace.c           |   22 ++++++++----------
 fs/proc/base.c           |    5 +++--
 include/linux/init_task.h |    1 +
 include/linux/namespace.h |    6 ++----
 include/linux/nsproxy.h   |    3 +++
 include/linux/sched.h     |    4 +---
 kernel/exit.c            |    5 -----
 kernel/fork.c            |   19 +++++++------------
 kernel/nsproxy.c         |   40 ++++++++++++++++++++++++++++++++++++++++-----
 9 files changed, 60 insertions(+), 45 deletions(-)

a06a931bb7f9f82df3267dfce14c8a9784ae68e4
diff --git a/fs/namespace.c b/fs/namespace.c
index 2c5f1f8..33330fe 100644
--- a/fs/namespace.c
+++ b/fs/namespace.c
@@ -133,7 +133,7 @@ struct vfsmount *lookup_mnt(struct vfsmo

 static inline int check_mnt(struct vfsmount *mnt)
 {
- return mnt->mnt_namespace == current->namespace;
+ return mnt->mnt_namespace == current->nsproxy->namespace;
 }

 static void touch_namespace(struct namespace *ns)
@@ -832,7 +832,7 @@ static int attach_recursive_mnt(struct v
  if (parent_nd) {
   detach_mnt(source_mnt, parent_nd);
   attach_mnt(source_mnt, nd);
-  touch_namespace(current->namespace);
+  touch_namespace(current->nsproxy->namespace);
  } else {
   mnt_set_mountpoint(dest_mnt, dest_dentry, source_mnt);
   commit_tree(source_mnt);
```

```
@@ -1372,7 +1372,7 @@ dput_out:
  */
 struct namespace *dup_namespace(struct task_struct *tsk, struct fs_struct *fs)
 {
- struct namespace *namespace = tsk->namespace;
+ struct namespace *namespace = tsk->nsproxy->namespace;
  struct namespace *new_ns;
  struct vfsmount *rootmnt = NULL, *pwdmnt = NULL, *altrootmnt = NULL;
  struct vfsmount *p, *q;
@@ -1439,7 +1439,7 @@ struct namespace *dup_namespace(struct t

 int copy_namespace(int flags, struct task_struct *tsk)
 {
- struct namespace *namespace = tsk->namespace;
+ struct namespace *namespace = tsk->nsproxy->namespace;
  struct namespace *new_ns;
  int err = 0;

@@ -1462,7 +1462,7 @@ int copy_namespace(int flags, struct tas
   goto out;
  }

- tsk->namespace = new_ns;
+ tsk->nsproxy->namespace = new_ns;

 out:
  put_namespace(namespace);
@@ -1685,7 +1685,7 @@ asmlinkage long sys_pivot_root(const cha
  detach_mnt(user_nd.mnt, &root_parent);
  attach_mnt(user_nd.mnt, &old_nd);     /* mount old root on put_old */
  attach_mnt(new_nd.mnt, &root_parent); /* mount new_root on / */
- touch_namespace(current->namespace);
+ touch_namespace(current->nsproxy->namespace);
  spin_unlock(&vfsmount_lock);
  chroot_fs_refs(&user_nd, &new_nd);
  security_sb_post_pivotroot(&user_nd, &new_nd);
@@ -1711,7 +1711,6 @@ static void __init init_mount_tree(void)
 {
  struct vfsmount *mnt;
  struct namespace *namespace;
- struct task_struct *g, *p;

  mnt = do_kern_mount("rootfs", 0, "rootfs", NULL);
  if (IS_ERR(mnt))
@@ -1727,13 +1726,8 @@ static void __init init_mount_tree(void)
  namespace->root = mnt;
  mnt->mnt_namespace = namespace;
```

```
- init_task.namespace = namespace;
- read_lock(&tasklist_lock);
- do_each_thread(g, p) {
-  get_namespace(namespace);
-  p->namespace = namespace;
- } while_each_thread(g, p);
- read_unlock(&tasklist_lock);
+ init_task.nsproxy->namespace = namespace;
+ get_namespace(namespace);

  set_fs_pwd(current->fs, namespace->root, namespace->root->mnt_root);
  set_fs_root(current->fs, namespace->root, namespace->root->mnt_root);
diff --git a/fs/proc/base.c b/fs/proc/base.c
index 6cc77dc..f74acae 100644
--- a/fs/proc/base.c
+++ b/fs/proc/base.c
@@ -72,6 +72,7 @@
 #include <linux/cpuset.h>
 #include <linux/audit.h>
 #include <linux/poll.h>
+#include <linux/nsproxy.h>
 #include "internal.h"

 /*
@@ -685,7 +686,7 @@ static int mounts_open(struct inode *ino
  int ret = -EINVAL;

  task_lock(task);
- namespace = task->namespace;
+ namespace = task->nsproxy->namespace;
  if (namespace)
   get_namespace(namespace);
  task_unlock(task);
@@ -752,7 +753,7 @@ static int mountstats_open(struct inode
   struct seq_file *m = file->private_data;
   struct namespace *namespace;
   task_lock(task);
-  namespace = task->namespace;
+  namespace = task->nsproxy->namespace;
   if (namespace)
    get_namespace(namespace);
   task_unlock(task);
diff --git a/include/linux/init_task.h b/include/linux/init_task.h
index 79ec4ea..672dc04 100644
--- a/include/linux/init_task.h
+++ b/include/linux/init_task.h
@@ -70,6 +70,7 @@ extern struct nsproxy init_nsproxy;
 #define INIT_NSPROXY(nsproxy) {      \
```

```
  .count  = ATOMIC_INIT(1),    \
  .nslock  = SPIN_LOCK_UNLOCKED,    \
+ .namespace = NULL,      \
 }

 #define INIT_SIGHAND(sighand) {      \
diff --git a/include/linux/namespace.h b/include/linux/namespace.h
index 3abc8e3..d137009 100644
--- a/include/linux/namespace.h
+++ b/include/linux/namespace.h
@@ -4,6 +4,7 @@

 #include <linux/mount.h>
 #include <linux/sched.h>
+#include <linux/nsproxy.h>

 struct namespace {
  atomic_t  count;
@@ -26,11 +27,8 @@ static inline void put_namespace(struct

 static inline void exit_namespace(struct task_struct *p)
 {
- struct namespace *namespace = p->namespace;
+ struct namespace *namespace = p->nsproxy->namespace;
  if (namespace) {
-  task_lock(p);
-  p->namespace = NULL;
-  task_unlock(p);
   put_namespace(namespace);
  }
 }
diff --git a/include/linux/nsproxy.h b/include/linux/nsproxy.h
index 7bdebfa..7ebe666 100644
--- a/include/linux/nsproxy.h
+++ b/include/linux/nsproxy.h
@@ -4,6 +4,8 @@
 #include <linux/spinlock.h>
 #include <linux/sched.h>

+struct namespace;
+
 /*
  * A structure to contain pointers to all per-process
  * namespaces - fs (mount), uts, network, sysvipc, etc.
@@ -19,6 +21,7 @@
 struct nsproxy {
  atomic_t count;
  spinlock_t nslock;
```

```
+ struct namespace *namespace;
 };
 extern struct nsproxy init_nsproxy;

diff --git a/include/linux/sched.h b/include/linux/sched.h
index 4c0bbb3..f2c945b 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -235,7 +235,6 @@ extern signed long schedule_timeout_inte
 extern signed long schedule_timeout_uninterruptible(signed long timeout);
 asmlinkage void schedule(void);

-struct namespace;
 struct nsproxy;

 /* Maximum number of active map areas.. This is a random (large) number */
@@ -807,8 +806,7 @@ struct task_struct {
  struct fs_struct *fs;
 /* open file information */
  struct files_struct *files;
-/* namespace */
- struct namespace *namespace;
+/* namespaces */
  struct nsproxy *nsproxy;
 /* signal handlers */
  struct signal_struct *signal;
diff --git a/kernel/exit.c b/kernel/exit.c
index da2fc84..240d0df 100644
--- a/kernel/exit.c
+++ b/kernel/exit.c
@@ -36,7 +36,6 @@
 #include <linux/compat.h>
 #include <linux/pipe_fs_i.h>
 #include <linux/audit.h> /* for audit_free() */
-#include <linux/nsproxy.h>

 #include <asm/uaccess.h>
 #include <asm/unistd.h>
@@ -415,11 +414,8 @@ void daemonize(const char *name, ...)
  current->fs = fs;
  atomic_inc(&fs->count);

- exit_namespace(current);
  exit_task_namespaces(current);
- current->namespace = init_task.namespace;
  current->nsproxy = init_task.nsproxy;
- get_namespace(current->namespace);
  get_task_namespaces(current);
```

```
   exit_files(current);
@@ -924,7 +920,6 @@ fastcall NORET_TYPE void do_exit(long co
  exit_sem(tsk);
  __exit_files(tsk);
  __exit_fs(tsk);
- exit_namespace(tsk);
  exit_task_namespaces(tsk);
  exit_thread();
  cpuset_exit(tsk);
diff --git a/kernel/fork.c b/kernel/fork.c
index 60303c3..ddab7e7 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1063,11 +1063,9 @@ static task_t *copy_process(unsigned lon
   goto bad_fork_cleanup_mm;
  if ((retval = copy_namespaces(clone_flags, p)))
   goto bad_fork_cleanup_keys;
- if ((retval = copy_namespace(clone_flags, p)))
-  goto bad_fork_cleanup_namespaces;
  retval = copy_thread(0, clone_flags, stack_start, stack_size, p, regs);
  if (retval)
-  goto bad_fork_cleanup_namespace;
+  goto bad_fork_cleanup_namespaces;

  p->set_child_tid = (clone_flags & CLONE_CHILD_SETTID) ? child_tidptr : NULL;
  /*
@@ -1154,7 +1152,7 @@ static task_t *copy_process(unsigned lon
   spin_unlock(&current->sighand->siglock);
   write_unlock_irq(&tasklist_lock);
   retval = -ERESTARTNOINTR;
-  goto bad_fork_cleanup_namespace;
+  goto bad_fork_cleanup_namespaces;
  }

  if (clone_flags & CLONE_THREAD) {
@@ -1167,7 +1165,7 @@ static task_t *copy_process(unsigned lon
    spin_unlock(&current->sighand->siglock);
    write_unlock_irq(&tasklist_lock);
    retval = -EAGAIN;
-   goto bad_fork_cleanup_namespace;
+   goto bad_fork_cleanup_namespaces;
   }

   p->group_leader = current->group_leader;
@@ -1219,8 +1217,6 @@ static task_t *copy_process(unsigned lon
  proc_fork_connector(p);
  return p;
```

```
-bad_fork_cleanup_namespace:
- exit_namespace(p);
 bad_fork_cleanup_namespaces:
  exit_task_namespaces(p);
 bad_fork_cleanup_keys:
@@ -1472,10 +1468,9 @@ static int unshare_fs(unsigned long unsh
  */
 static int unshare_namespace(unsigned long unshare_flags, struct namespace **new_nsp, struct
fs_struct *new_fs)
 {
- struct namespace *ns = current->namespace;
+ struct namespace *ns = current->nsproxy->namespace;

- if ((unshare_flags & CLONE_NEWNS) &&
-     (ns && atomic_read(&ns->count) > 1)) {
+ if ((unshare_flags & CLONE_NEWNS) && ns) {
   if (!capable(CAP_SYS_ADMIN))
    return -EPERM;

@@ -1608,8 +1603,8 @@ asmlinkage long sys_unshare(unsigned lon
   }

   if (new_ns) {
-   ns = current->namespace;
-   current->namespace = new_ns;
+   ns = current->nsproxy->namespace;
+   current->nsproxy->namespace = new_ns;
    new_ns = ns;
   }

diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index 6f53df1..4103f58 100644
--- a/kernel/nsproxy.c
+++ b/kernel/nsproxy.c
@@ -1,18 +1,19 @@
 /*
  *  Copyright (C) 2006 IBM Corporation
- *
- *  Author: Serge Hallyn <serue@us.ibm.com>
- *
+ *
+ *  Author: Serge Hallyn <serue@us.ibm.com>
+ *
  *  This program is free software; you can redistribute it and/or
  *  modify it under the terms of the GNU General Public License as
  *  published by the Free Software Foundation, version 2 of the
  *  License.
```

```
- */
+ */

 #include <linux/compile.h>
 #include <linux/module.h>
 #include <linux/version.h>
 #include <linux/nsproxy.h>
+#include <linux/namespace.h>

 static inline void get_nsproxy(struct nsproxy *ns)
 {
@@ -53,6 +54,11 @@ struct nsproxy *dup_namespaces(struct ns
 {
  struct nsproxy *ns = clone_namespaces(orig);

+ if (ns) {
+  if (ns->namespace)
+   get_namespace(ns->namespace);
+ }
+
  return ns;
 }

@@ -63,16 +69,40 @@ struct nsproxy *dup_namespaces(struct ns
 int copy_namespaces(int flags, struct task_struct *tsk)
 {
  struct nsproxy *old_ns = tsk->nsproxy;
+ struct nsproxy *new_ns;
+ int err = 0;

  if (!old_ns)
   return 0;

  get_nsproxy(old_ns);

- return 0;
+ if (!(flags & CLONE_NEWNS))
+  return 0;
+
+ new_ns = clone_namespaces(old_ns);
+ if (!new_ns) {
+  err = -ENOMEM;
+  goto out;
+ }
+
+ tsk->nsproxy = new_ns;
+
+ err = copy_namespace(flags, tsk);
```

```
+ if (err) {
+  tsk->nsproxy = old_ns;
+  put_nsproxy(new_ns);
+  goto out;
+ }
+
+out:
+ put_nsproxy(old_ns);
+ return err;
 }

 void free_nsproxy(struct nsproxy *ns)
 {
+  if (ns->namespace)
+   put_namespace(ns->namespace);
   kfree(ns);
 }
--
1.1.6
```