Subject: [PATCH 8/8] user ns: implement user ns unshare
Posted by serue on Tue, 19 Dec 2006 23:01:47 GMT
View Forum Message <> Reply to Message

From: Serge E. Hallyn <serue@us.ibm.com>
Subject: [PATCH 8/8] user ns: implement user ns unshare

Implement CLONE_NEWUSER flag useable at clone and unshare.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
---
 include/linux/sched.h | 1 +
 include/linux/user_namespace.h | 10 +++++
 kernel/fork.c | 22 ++++++++++--
 kernel/nsproxy.c | 2 +
 kernel/user_namespace.c | 74 ++-
 5 files changed, 102 insertions(+), 7 deletions(-)

diff --git a/include/linux/sched.h b/include/linux/sched.h
index 73df38c..55ecf81 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -26,6 +26,7 @@ #define CLONE_CHILD_SETTID 0x01000000 /*
 #define CLONE_STOPPED 0x02000000 /* Start in stopped state */
 #define CLONE_NEWUTS 0x04000000 /* New utsname group? */
 #define CLONE_NEWIPC 0x08000000 /* New ipcs */
+#define CLONE_NEWUSER 0x10000000 /* New user namespace */

 /*
  * Scheduling policies
diff --git a/include/linux/user_namespace.h b/include/linux/user_namespace.h
index 4ad4c0d..d577ede 100644
--- a/include/linux/user_namespace.h
+++ b/include/linux/user_namespace.h
@@ -25,6 +25,7 @@ static inline struct user_namespace *get
 }

 extern int copy_user_ns(int flags, struct task_struct *tsk);
+extern int unshare_user_ns(unsigned long flags, struct user_namespace **new_user);
 extern void free_user_ns(struct kref *kref);

 static inline void put_user_ns(struct user_namespace *ns)
@@ -40,6 +41,15 @@ static inline struct user_namespace *get
  return NULL;
 }

+static inline int unshare_user_ns(unsigned long flags,
+   struct user_namespace **new_user)

```
+{
+ if (flags & CLONE_NEWUSER)
+  return -EINVAL;
+
+ return 0;
+}
+
 static inline int copy_user_ns(int flags, struct task_struct *tsk)
 {
  return 0;
diff --git a/kernel/fork.c b/kernel/fork.c
index deafa6e..eead517 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -49,6 +49,7 @@ #include <linux/cn_proc.h>
 #include <linux/delayacct.h>
 #include <linux/taskstats_kern.h>
 #include <linux/random.h>
+#include <linux/user_namespace.h>

 #include <asm/pgtable.h>
 #include <asm/pgalloc.h>
@@ -1620,6 +1621,7 @@ asmlinkage long sys_unshare(unsigned lon
  struct nsproxy *new_nsproxy = NULL, *old_nsproxy = NULL;
  struct uts_namespace *uts, *new_uts = NULL;
  struct ipc_namespace *ipc, *new_ipc = NULL;
+ struct user_namespace *user, *new_user = NULL;

  check_unshare_flags(&unshare_flags);

@@ -1627,7 +1629,7 @@ asmlinkage long sys_unshare(unsigned lon
  err = -EINVAL;
  if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_NEWNS|CLONE_SIGHAND|
     CLONE_VM|CLONE_FILES|CLONE_SYSVSEM|
-    CLONE_NEWUTS|CLONE_NEWIPC))
+    CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWUSER))
   goto bad_unshare_out;

  if ((err = unshare_thread(unshare_flags)))
@@ -1648,18 +1650,20 @@ asmlinkage long sys_unshare(unsigned lon
   goto bad_unshare_cleanup_semundo;
  if ((err = unshare_ipcs(unshare_flags, &new_ipc)))
   goto bad_unshare_cleanup_uts;
+ if ((err = unshare_user_ns(unshare_flags, &new_user)))
+  goto bad_unshare_cleanup_ipc;

- if (new_ns || new_uts || new_ipc) {
+ if (new_ns || new_uts || new_ipc || new_user) {
```

```
   old_nsproxy = current->nsproxy;
   new_nsproxy = dup_namespaces(old_nsproxy);
   if (!new_nsproxy) {
    err = -ENOMEM;
-   goto bad_unshare_cleanup_ipc;
+   goto bad_unshare_cleanup_user;
   }
  }

  if (new_fs || new_ns || new_mm || new_fd || new_ulist ||
-    new_uts || new_ipc) {
+    new_uts || new_ipc || new_user) {

   task_lock(current);

@@ -1707,12 +1711,22 @@ asmlinkage long sys_unshare(unsigned lon
    new_ipc = ipc;
   }

+  if (new_user) {
+   user = current->nsproxy->user_ns;
+   current->nsproxy->user_ns = new_user;
+   new_user = user;
+  }
+
   task_unlock(current);
  }

  if (new_nsproxy)
   put_nsproxy(new_nsproxy);

+bad_unshare_cleanup_user:
+ if (new_user)
+  put_user_ns(new_user);
+
 bad_unshare_cleanup_ipc:
  if (new_ipc)
   put_ipc_ns(new_ipc);
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index cf43e06..2d9bafb 100644
--- a/kernel/nsproxy.c
+++ b/kernel/nsproxy.c
@@ -102,7 +102,7 @@ int copy_namespaces(int flags, struct ta

  get_nsproxy(old_ns);

- if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC)))
+ if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC | CLONE_NEWUSER)))
```

```
   return 0;

  new_ns = clone_namespaces(old_ns);
diff --git a/kernel/user_namespace.c b/kernel/user_namespace.c
index 368f8da..d1ce4c1 100644
--- a/kernel/user_namespace.c
+++ b/kernel/user_namespace.c
@@ -20,17 +20,87 @@ struct user_namespace init_user_ns = {
 EXPORT_SYMBOL_GPL(init_user_ns);

 #ifdef CONFIG_USER_NS
+/*
+ * Clone a new ns copying an original user ns, setting refcount to 1
+ * @old_ns: namespace to clone
+ * Return NULL on error (failure to kmalloc), new ns otherwise
+ */
+static struct user_namespace *clone_user_ns(struct user_namespace *old_ns)
+{
+       struct user_namespace *ns;
+       struct user_struct *new_user;
+       int n;
+
+       ns = kmalloc(sizeof(struct user_namespace), GFP_KERNEL);
+       if (!ns)
+               return NULL;
+
+       kref_init(&ns->kref);
+
+       for(n = 0; n < UIDHASH_SZ; ++n)
+               INIT_LIST_HEAD(ns->uidhash_table + n);
+
+       /* Insert new root user.  */
+       ns->root_user = alloc_uid(ns, 0);
+       if (!ns->root_user) {
+               kfree(ns);
+               return NULL;
+       }
+
+       /* Reset current->user with a new one */
+       new_user = alloc_uid(ns, current->uid);
+       if (!new_user) {
+               free_uid(ns->root_user);
+               kfree(ns);
+               return NULL;
+       }
+
+       switch_uid(new_user);
+       return ns;
```

```
+}
+
+/*
+ * unshare the current process' user namespace.
+ */
+int unshare_user_ns(unsigned long flags,
+   struct user_namespace **new_user)
+{
+       if (flags & CLONE_NEWUSER) {
+               if (!capable(CAP_SYS_ADMIN))
+                       return -EPERM;
+
+               *new_user = clone_user_ns(current->nsproxy->user_ns);
+               if (!*new_user)
+                       return -ENOMEM;
+       }
+
+       return 0;
+}

 int copy_user_ns(int flags, struct task_struct *tsk)
 {
- struct user_namespace *old_ns = tsk->nsproxy->user_ns;
+ struct user_namespace *new_ns, *old_ns = tsk->nsproxy->user_ns;
  int err = 0;

  if (!old_ns)
   return 0;

  get_user_ns(old_ns);
- return err;
+ if (!(flags & CLONE_NEWUSER))
+  return 0;
+ err = -EPERM;
+ if (!capable(CAP_SYS_ADMIN))
+  goto out;
+ err = -ENOMEM;
+ new_ns = clone_user_ns(old_ns);
+ if (!new_ns)
+  goto out;
+
+ tsk->nsproxy->user_ns = new_ns;
+ err = 0;
+out:
+ put_user_ns(old_ns);
+ return 0;
 }
```

void free_user_ns(struct kref *kref)
--
1.4.1

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