Subject: [PATCH 2/6] user namespace : add unshare
Posted by serue on Mon, 04 Jun 2007 19:40:42 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 13:00:47 -0400
Subject: [PATCH 2/6] user namespace : add unshare

Changelog: Fix !CONFIG_USER_NS clone with CLONE_NEWUSER so it returns -EINVAL
 rather than 0, so that userspace knows they didn't get a new user
 namespace.

Cc: Serge E. Hallyn <serue@us.ibm.com>
Cc: Herbert Poetzl <herbert@13thfloor.at>
Cc: Kirill Korotaev <dev@sw.ru>
Cc: "Eric W. Biederman" <ebiederm@xmission.com>
Signed-off-by: Andrew Morton <akpm@osdl.org>

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

---

 include/linux/sched.h | 1 +
 include/linux/user_namespace.h | 5 +++-
 kernel/fork.c | 2 +-
 kernel/nsproxy.c | 10 +++++++--
 kernel/user_namespace.c | 46 ++-
 5 files changed, 59 insertions(+), 5 deletions(-)

4ddd1e8f0539d843e998ea26d0c6c9cb751cbf3e
diff --git a/include/linux/sched.h b/include/linux/sched.h
index 259d0a5..ad19efa 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 c2debd8..9044456 100644
--- a/include/linux/user_namespace.h
+++ b/include/linux/user_namespace.h
@@ -45,7 +45,10 @@ static inline struct user_namespace *get

```
 static inline struct user_namespace *copy_user_ns(int flags,
        struct user_namespace *old_ns)
 {
- return 0;
+ if (flags & CLONE_NEWUSER)
+  return ERR_PTR(-EINVAL);
+
+ return NULL;
 }

 static inline void put_user_ns(struct user_namespace *ns)
diff --git a/kernel/fork.c b/kernel/fork.c
index c20137e..123bc93 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1581,7 +1581,7 @@ 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)))
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index 1ea8a73..fb00706 100644
--- a/kernel/nsproxy.c
+++ b/kernel/nsproxy.c
@@ -121,7 +121,7 @@ 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;

  if (!capable(CAP_SYS_ADMIN)) {
@@ -168,9 +168,15 @@ int unshare_nsproxy_namespaces(unsigned
  struct nsproxy *old_ns = current->nsproxy;
  int err = 0;

- if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC)))
+ if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
+         CLONE_NEWUSER)))
   return 0;

+#ifndef CONFIG_USER_NS
+ if (unshare_flags & CLONE_NEWUSER)
```

```
+  return -EINVAL;
+#endif
+
  if (!capable(CAP_SYS_ADMIN))
   return -EPERM;

diff --git a/kernel/user_namespace.c b/kernel/user_namespace.c
index 3d79642..89a27e8 100644
--- a/kernel/user_namespace.c
+++ b/kernel/user_namespace.c
@@ -21,6 +21,45 @@ 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;
+}
+
 struct user_namespace * copy_user_ns(int flags, struct user_namespace *old_ns)
 {
  struct user_namespace *new_ns;
@@ -28,7 +67,12 @@ struct user_namespace * copy_user_ns(int
  BUG_ON(!old_ns);
  get_user_ns(old_ns);

- new_ns = old_ns;
+ if (!(flags & CLONE_NEWUSER))
+  return old_ns;
+
+ new_ns = clone_user_ns(old_ns);
+
+ put_user_ns(old_ns);
  return new_ns;
 }

--
1.3.2
```

Subject: Re: [PATCH 2/6] user namespace : add unshare
Posted by Pavel Emelianov on Fri, 08 Jun 2007 08:37:52 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn wrote:
>>From nobody Mon Sep 17 00:00:00 2001
> From: Serge E. Hallyn <serue@us.ibm.com>
> Date: Thu, 5 Apr 2007 13:00:47 -0400
> Subject: [PATCH 2/6] user namespace : add unshare
>
> Changelog: Fix !CONFIG_USER_NS clone with CLONE_NEWUSER so it returns -EINVAL
> rather than 0, so that userspace knows they didn't get a new user
> namespace.
>
> Cc: Serge E. Hallyn <serue@us.ibm.com>
> Cc: Herbert Poetzl <herbert@13thfloor.at>
> Cc: Kirill Korotaev <dev@sw.ru>
> Cc: "Eric W. Biederman" <ebiederm@xmission.com>
> Signed-off-by: Andrew Morton <akpm@osdl.org>
>
> Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>

Acked-by: Pavel Emelianov <xemul@openvz.org>

> ---
>
> include/linux/sched.h | 1 +
> include/linux/user_namespace.h | 5 +++-
> kernel/fork.c | 2 +-
> kernel/nsproxy.c | 10 +++++++--
> kernel/user_namespace.c | 46 ++
> 5 files changed, 59 insertions(+), 5 deletions(-)
>
> 4ddd1e8f0539d843e998ea26d0c6c9cb751cbf3e
> diff --git a/include/linux/sched.h b/include/linux/sched.h
> index 259d0a5..ad19efa 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 c2debd8..9044456 100644
> --- a/include/linux/user_namespace.h
> +++ b/include/linux/user_namespace.h
> @@ -45,7 +45,10 @@ static inline struct user_namespace *get
> static inline struct user_namespace *copy_user_ns(int flags,
>   struct user_namespace *old_ns)
> {
> - return 0;
> + if (flags & CLONE_NEWUSER)
> + return ERR_PTR(-EINVAL);
> +
> + return NULL;
> }
>
> static inline void put_user_ns(struct user_namespace *ns)
> diff --git a/kernel/fork.c b/kernel/fork.c
> index c20137e..123bc93 100644
> --- a/kernel/fork.c
> +++ b/kernel/fork.c
> @@ -1581,7 +1581,7 @@ 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)))
> diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
> index 1ea8a73..fb00706 100644
> --- a/kernel/nsproxy.c
> +++ b/kernel/nsproxy.c
> @@ -121,7 +121,7 @@ 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;
>
>    if (!capable(CAP_SYS_ADMIN)) {
> @@ -168,9 +168,15 @@ int unshare_nsproxy_namespaces(unsigned
>    struct nsproxy *old_ns = current->nsproxy;
>    int err = 0;
>
> - if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC)))
> + if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
> +          CLONE_NEWUSER)))
>     return 0;
>
> +#ifndef CONFIG_USER_NS
> + if (unshare_flags & CLONE_NEWUSER)
> +  return -EINVAL;
> +#endif
> +
>    if (!capable(CAP_SYS_ADMIN))
>     return -EPERM;
>
> diff --git a/kernel/user_namespace.c b/kernel/user_namespace.c
> index 3d79642..89a27e8 100644
> --- a/kernel/user_namespace.c
> +++ b/kernel/user_namespace.c
> @@ -21,6 +21,45 @@ 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;
> +}
> +
>  struct user_namespace * copy_user_ns(int flags, struct user_namespace *old_ns)
>  {
>   struct user_namespace *new_ns;
> @@ -28,7 +67,12 @@ struct user_namespace * copy_user_ns(int
>   BUG_ON(!old_ns);
>   get_user_ns(old_ns);
>
> - new_ns = old_ns;
> + if (!(flags & CLONE_NEWUSER))
> +  return old_ns;
> +
> + new_ns = clone_user_ns(old_ns);
> +
```

> + put_user_ns(old_ns);
> return new_ns;
> }
>