Subject: [PATCH 2/2] Fix user namespace exiting OOPs
Posted by Pavel Emelianov on Fri, 14 Sep 2007 13:12:26 GMT
View Forum Message <> Reply to Message

It turned out, that the user namespace is released during
the do_exit() in exit_task_namespaces(), but the struct
user_struct is released only during the put_task_struct(),
i.e. MUCH later.

On debug kernels with poisoned slabs this will cause the
oops in uid_hash_remove() because the head of the chain,
which resides inside the struct user_namespace, will be
already freed and poisoned.

Since the uid hash itself is required only when someone
can search it, i.e. when the namespace is alive, we can
safely unhash all the user_struct-s from it during the
namespace exiting. The subsequent free_uid() will complete
the user_struct destruction.

For example simple program

```
#include <sched.h>

char stack[2 * 1024 * 1024];

int f(void *foo)
{
 return 0;
}

int main(void)
{
 clone(f, stack + 1 * 1024 * 1024, 0x10000000, 0);
 return 0;
}
```

run on kernel with CONFIG_USER_NS turned on will oops the
kernel immediately.

This was spotted during OpenVZ kernel testing.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
Signed-off-by: Alexey Dobriyan <adobriyan@openvz.org>

---

include/linux/sched.h | 1 +

```
 kernel/user.c          |   26 ++++++++++++++++++++++++-
 kernel/user_namespace.c |    2 +-
 3 files changed, 27 insertions(+), 2 deletions(-)

diff --git a/include/linux/sched.h b/include/linux/sched.h
index a2afa88..b4a4211 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -1530,6 +1530,7 @@ static inline struct user_struct *get_ui
 }
 extern void free_uid(struct user_struct *);
 extern void switch_uid(struct user_struct *);
+extern void release_uids(struct user_namespace *ns);

 #include <asm/current.h>

diff --git a/kernel/user.c b/kernel/user.c
index add57c7..e1f2d32 100644
--- a/kernel/user.c
+++ b/kernel/user.c
@@ -62,7 +62,7 @@ static inline void uid_hash_insert(struc

 static inline void uid_hash_remove(struct user_struct *up)
 {
- hlist_del(&up->uidhash_node);
+ hlist_del_init(&up->uidhash_node);
 }

 static inline struct user_struct *uid_hash_find(uid_t uid, struct hlist_head *hashent)
@@ -199,6 +199,30 @@ void switch_uid(struct user_struct *new_
  suid_keys(current);
 }

+void release_uids(struct user_namespace *ns)
+{
+ int i;
+ unsigned long flags;
+ struct hlist_head *head;
+ struct hlist_node *nd;
+
+ spin_lock_irqsave(&uidhash_lock, flags);
+ /*
+  * collapse the chains so that the user_struct-s will
+  * be still alive, but not in hashes. subsequent free_uid()
+  * will free them.
+  */
+ for (i = 0; i < UIDHASH_SZ; i++) {
+  head = ns->uidhash_table + i;
```

```
+  while (!hlist_empty(head)) {
+   nd = head->first;
+   hlist_del_init(nd);
+  }
+ }
+ spin_unlock_irqrestore(&uidhash_lock, flags);
+
+ free_uid(ns->root_user);
+}

 static int __init uid_cache_init(void)
 {
diff --git a/kernel/user_namespace.c b/kernel/user_namespace.c
index 85af942..df1d2cf 100644
--- a/kernel/user_namespace.c
+++ b/kernel/user_namespace.c
@@ -81,7 +81,7 @@ void free_user_ns(struct kref *kref)
  struct user_namespace *ns;

  ns = container_of(kref, struct user_namespace, kref);
- free_uid(ns->root_user);
+ release_uids(ns);
  kfree(ns);
 }
```