Subject: [PATCH 5/4] Clean up capability.h and capability.c
Posted by serge on Thu, 24 Feb 2011 00:22:49 GMT
View Forum Message <> Reply to Message

Convert macros to functions to let type safety do its thing. Switch some functions from ints to more appropriate bool. Move all forward declarations together to top of the #ifdef __KERNEL__ section. Use kernel-doc format for comments.

Some macros couldn't be converted because they use functions from security.h which sometimes are extern and sometimes static inline, and we don't want to #include security.h in capability.h.

Also add a real current_user_ns function (and convert the existing macro to _current_user_ns() so we can use it in capability.h without #including cred.h.

Signed-off-by: Serge E. Hallyn <serge.hallyn@canonical.com>

```
---
 include/linux/capability.h |   38 ++++++++++++++++++++++++++------------
 include/linux/cred.h       |    4 +++-
 kernel/capability.c        |   20 ++++++++++++--------
 kernel/cred.c              |    5 +++++
 4 files changed, 46 insertions(+), 21 deletions(-)

diff --git a/include/linux/capability.h b/include/linux/capability.h
index bc0f262..688462f 100644
--- a/include/linux/capability.h
+++ b/include/linux/capability.h
@@ -368,6 +368,17 @@ struct cpu_vfs_cap_data {

 #ifdef __KERNEL__

+struct dentry;
+struct user_namespace;
+
+extern struct user_namespace init_user_ns;
+
+struct user_namespace *current_user_ns(void);
+
+extern const kernel_cap_t __cap_empty_set;
+extern const kernel_cap_t __cap_full_set;
+extern const kernel_cap_t __cap_init_eff_set;
+
 /*
  * Internal kernel functions only
  */
@@ -530,10 +541,6 @@ static inline kernel_cap_t cap_raise_nfsd_set(const kernel_cap_t a,
```

```
      cap_intersect(permitted, __cap_nfsd_set));
 }

-extern const kernel_cap_t __cap_empty_set;
-extern const kernel_cap_t __cap_full_set;
-extern const kernel_cap_t __cap_init_eff_set;
-
 /**
  * has_capability - Determine if a task has a superior capability available
  * @t: The task in question
@@ -560,18 +567,25 @@ extern const kernel_cap_t __cap_init_eff_set;
  * Note that this does not set PF_SUPERPRIV on the task.
  */
 #define has_capability_noaudit(t, cap) \
- (security_real_capable_noaudit((t), &init_user_ns, (cap)) == 0)
+       (security_real_capable_noaudit((t), &init_user_ns, (cap)) == 0)

-struct user_namespace;
-extern struct user_namespace init_user_ns;
-extern int capable(int cap);
-extern int ns_capable(struct user_namespace *ns, int cap);
-extern int task_ns_capable(struct task_struct *t, int cap);
+extern bool capable(int cap);
+extern bool ns_capable(struct user_namespace *ns, int cap);
+extern bool task_ns_capable(struct task_struct *t, int cap);

-#define nsown_capable(cap) (ns_capable(current_user_ns(), (cap)))
+/**
+ * nsown_capable - Check superior capability to one's own user_ns
+ * @cap: The capability in question
+ *
+ * Return true if the current task has the given superior capability
+ * targeted at its own user namespace.
+ */
+static inline bool nsown_capable(int cap)
+{
+ return ns_capable(current_user_ns(), cap);
+}

 /* audit system wants to get cap info from files as well */
-struct dentry;
 extern int get_vfs_caps_from_disk(const struct dentry *dentry, struct cpu_vfs_cap_data
*cpu_caps);

 #endif /* __KERNEL__ */
diff --git a/include/linux/cred.h b/include/linux/cred.h
index 4aaeab3..9aeeb0b 100644
--- a/include/linux/cred.h
```

```
+++ b/include/linux/cred.h
@@ -354,9 +354,11 @@ static inline void put_cred(const struct cred *_cred)
 #define current_fsgid()  (current_cred_xxx(fsgid))
 #define current_cap()  (current_cred_xxx(cap_effective))
 #define current_user()  (current_cred_xxx(user))
-#define current_user_ns() (current_cred_xxx(user)->user_ns)
+#define _current_user_ns() (current_cred_xxx(user)->user_ns)
 #define current_security() (current_cred_xxx(security))

+extern struct user_namespace *current_user_ns(void);
+
 #define current_uid_gid(_uid, _gid)  \
 do {      \
  const struct cred *__cred;  \
diff --git a/kernel/capability.c b/kernel/capability.c
index 916658c..0a3d2c8 100644
--- a/kernel/capability.c
+++ b/kernel/capability.c
@@ -300,7 +300,7 @@ error:
  * This sets PF_SUPERPRIV on the task if the capability is available on the
  * assumption that it's about to be used.
  */
-int capable(int cap)
+bool capable(int cap)
 {
  return ns_capable(&init_user_ns, cap);
 }
@@ -317,7 +317,7 @@ EXPORT_SYMBOL(capable);
  * This sets PF_SUPERPRIV on the task if the capability is available on the
  * assumption that it's about to be used.
  */
-int ns_capable(struct user_namespace *ns, int cap)
+bool ns_capable(struct user_namespace *ns, int cap)
 {
  if (unlikely(!cap_valid(cap))) {
   printk(KERN_CRIT "capable() called with invalid cap=%u\n", cap);
@@ -326,17 +326,21 @@ int ns_capable(struct user_namespace *ns, int cap)

  if (security_capable(ns, current_cred(), cap) == 0) {
   current->flags |= PF_SUPERPRIV;
-  return 1;
+  return true;
  }
- return 0;
+ return false;
 }
 EXPORT_SYMBOL(ns_capable);
```

```
-/*
- * does current have capability 'cap' to the user namespace of task
- * 't'.  Return true if it does, false otherwise.
+/**
+ * task_ns_capable - Determine whether current task has a superior
+ * capability targeted at a specific task's user namespace.
+ * @t: The task whose user namespace is targeted.
+ * @cap: The capability in question.
+ *
+ *  Return true if it does, false otherwise.
  */
-int task_ns_capable(struct task_struct *t, int cap)
+bool task_ns_capable(struct task_struct *t, int cap)
 {
  return ns_capable(task_cred_xxx(t, user)->user_ns, cap);
 }
diff --git a/kernel/cred.c b/kernel/cred.c
index 3a9d6dd..e447fa2 100644
--- a/kernel/cred.c
+++ b/kernel/cred.c
@@ -741,6 +741,11 @@ int set_create_files_as(struct cred *new, struct inode *inode)
 }
 EXPORT_SYMBOL(set_create_files_as);

+struct user_namespace *current_user_ns(void)
+{
+ return _current_user_ns();
+}
+
 #ifdef CONFIG_DEBUG_CREDENTIALS

 bool creds_are_invalid(const struct cred *cred)
--
1.7.0.4
```