

[PATCH] Keys: Add possessor permissions to keys

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-09/6370.html>

From: David Howells (dhowells_at_redhat.com)

Date: 09/21/05

To: torvalds@osdl.org, akpm@osdl.org
Date: Wed, 21 Sep 2005 15:48:10 +0100

The attached patch adds extra permission grants to keys for the possessor of a key in addition to the owner, group and other permissions bits. This makes SUID binaries easier to support without going as far as labelling keys and key targets using the LSM facilities.

This patch adds a second "pointer type" to key structures (struct key_ref *) that can have the bottom bit of the address set to indicate the possession of a key. This is propagated through searches from the keyring to the discovered key. It has been made a separate type so that the compiler can spot attempts to dereference a potentially incorrect pointer.

The "possession" attribute can't be attached to a key structure directly as it's not an intrinsic property of a key.

Pointers to keys have been replaced with struct key_ref *'s wherever possession information needs to be passed through.

This does assume that the bottom bit of the pointer will always be zero on return from `kmem_cache_alloc()`.

Signed-Off-By: David Howells <dhowells@redhat.com>

```
warthog>diffstat -p1 keys-possess-2614rc1mml.diff
```

```
Documentation/keys.txt      | 64 ++++++-----
include/linux/key-ui.h     | 28 +++++-
include/linux/key.h        | 59 ++++++-----
security/keys/internal.h   | 28 +++++-
security/keys/key.c        | 82 ++++++-----
security/keys/keyctl.c     | 172 ++++++-----
security/keys/keyring.c    | 84 ++++++-----
security/keys/proc.c       | 2
security/keys/process_keys.c | 101 ++++++-----
security/keys/request_key.c | 36 +++++-
security/keys/request_key_auth.c | 2
```

```
11 files changed, 401 insertions(+), 257 deletions(-)
```

```
diff -uNrp linux-2.6.14-rc1-mml/Documentation/keys.txt linux-2.6.14-rc1-mml-keys/Documentation/ke
```

```
--- linux-2.6.14-rc1-mml/Documentation/keys.txt 2005-08-30 13:56:09.000000000 +0100
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
+++ linux-2.6.14-rc1-mm1-keys/Documentation/keys.txt      2005-09-21 15:19:22.000000000 +0100
@@ -195,8 +195,8 @@ KEY ACCESS PERMISSIONS
=====
```

Keys have an owner user ID, a group access ID, and a permissions mask. The mask -has up to eight bits each for user, group and other access. Only five of each -set of eight bits are defined. These permissions granted are:
+has up to eight bits each for possessor, user, group and other access. Only +five of each set of eight bits are defined. These permissions granted are:

(*) View

```
@@ -241,16 +241,16 @@ about the status of the key service:
    type, description and permissions. The payload of the key is not available
    this way:
```

-	SERIAL	FLAGS	USAGE	EXPY	PERM	UID	GID	TYPE	DESCRIPTION: SUMMARY
-	00000001	I-----	39	perm	1f0000	0	0	keyring	_uid_ses.0: 1/4
-	00000002	I-----	2	perm	1f0000	0	0	keyring	_uid.0: empty
-	00000007	I-----	1	perm	1f0000	0	0	keyring	_pid.1: empty
-	0000018d	I-----	1	perm	1f0000	0	0	keyring	_pid.412: empty
-	000004d2	I--Q--	1	perm	1f0000	32	-1	keyring	_uid.32: 1/4
-	000004d3	I--Q--	3	perm	1f0000	32	-1	keyring	_uid_ses.32: empty
-	00000892	I--QU-	1	perm	1f0000	0	0	user	metal:copper: 0
-	00000893	I--Q-N	1	35s	1f0000	0	0	user	metal:silver: 0
-	00000894	I--Q--	1	10h	1f0000	0	0	user	metal:gold: 0
+	SERIAL	FLAGS	USAGE	EXPY	PERM	UID	GID	TYPE	DESCRIPTION: SUMMARY
+	00000001	I-----	39	perm	1f1f0000	0	0	keyring	_uid_ses.0: 1/4
+	00000002	I-----	2	perm	1f1f0000	0	0	keyring	_uid.0: empty
+	00000007	I-----	1	perm	1f1f0000	0	0	keyring	_pid.1: empty
+	0000018d	I-----	1	perm	1f1f0000	0	0	keyring	_pid.412: empty
+	000004d2	I--Q--	1	perm	1f1f0000	32	-1	keyring	_uid.32: 1/4
+	000004d3	I--Q--	3	perm	1f1f0000	32	-1	keyring	_uid_ses.32: empty
+	00000892	I--QU-	1	perm	1f000000	0	0	user	metal:copper: 0
+	00000893	I--Q-N	1	35s	1f1f0000	0	0	user	metal:silver: 0
+	00000894	I--Q--	1	10h	001f0000	0	0	user	metal:gold: 0

The flags are:

```
@@ -637,6 +637,34 @@ call, and the key released upon close. H
    two different users opening the same file is left to the filesystem author to
    solve.
```

+Note that there are two different types of pointers to keys that may be +encountered:

```
+
+ (*) struct key *
+
+ This simply points to the key structure itself. Key structures will be at
+ least four-byte aligned.
```

```
+ (*) struct key_ref *
+
+ This is equivalent to a struct key *, but the least significant bit is set
+ if the caller "possesses" the key. By "possession" it is meant that the
+ calling processes has a searchable link to the key from one of its
+ keyrings. Their are three functions for dealing with these:
```

```
+ struct key_ref *key_mkref(const struct key *key,
+                          unsigned long possession);
+
+ struct key *key_deref(const struct key_ref *key_ref);
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
+
+   unsigned long is_key_posessed(const struct key_ref *key_ref);
+
+   The first function constructs a key reference from a key pointer and
+   possession information (which must be 0 or 1 and no other value).
+
+   The second function retrieves the key pointer and the third retrieves the
+   possession flag).
```

When accessing a key's payload contents, certain precautions must be taken to prevent access vs modification races. See the section "Notes on accessing payload contents" for more information.

@@ -689,14 +717,18 @@ payload contents" for more information.

(*) If a keyring was found in the search, this can be further searched by:

```
-   struct key *keyring_search(struct key *keyring,
-                             const struct key_type *type,
-                             const char *description)
+   struct key_ref *keyring_search(struct key_ref *keyring,
+                                  const struct key_type *type,
+                                  const char *description)
```

This searches the keyring tree specified for a matching key. Error ENOKEY is returned upon failure. If successful, the returned key will need to be released.

```
+   The possession attribute from the keyring reference pointer is used to
+   control access through the permissions mask and is propagated to the
+   returned key reference pointer if successful.
```

(*) To check the validity of a key, this function can be called:

@@ -732,7 +764,7 @@ More complex payload contents must be al
key->payload.data. One of the following ways must be selected to access the
data:

```
- (1) Unmodifyable key type.  
+ (1) Unmodifiable key type.
```

```
   If the key type does not have a modify method, then the key's payload can
   be accessed without any form of locking, provided that it's known to be
diff -uNr linux-2.6.14-rc1-mm1/include/linux/key.h linux-2.6.14-rc1-mm1-keys/include/linux/key.h
--- linux-2.6.14-rc1-mm1/include/linux/key.h      2005-08-30 13:56:36.000000000 +0100
+++ linux-2.6.14-rc1-mm1-keys/include/linux/key.h  2005-09-21 13:26:39.000000000 +0100
@@ -30,16 +30,24 @@ typedef int32_t key_serial_t;
 typedef uint32_t key_perm_t;

 struct key;
+struct key_ref;          /* pointer to a key with a possession flag */

#ifdef CONFIG_KEYS

#undef KEY_DEBUGGING

-#define KEY_USR_VIEW      0x00010000      /* user can view a key's attributes */
-#define KEY_USR_READ     0x00020000      /* user can read key payload / view keyring */
-#define KEY_USR_WRITE    0x00040000      /* user can update key payload / add link to keyring */
-#define KEY_USR_SEARCH   0x00080000      /* user can find a key in search / search a keyring */
-#define KEY_USR_LINK     0x00100000      /* user can create a link to a key/keyring */
+#define KEY_POS_VIEW     0x01000000      /* possessor can view a key's attributes */
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
+#define KEY_POS_READ    0x02000000    /* possessor can read key payload / view keyring */
+#define KEY_POS_WRITE   0x04000000    /* possessor can update key payload / add link to keyring */
+#define KEY_POS_SEARCH  0x08000000    /* possessor can find a key in search / search a keyring */
+#define KEY_POS_LINK    0x10000000    /* possessor can create a link to a key/keyring */
+#define KEY_POS_ALL     0x1f000000
+
+#define KEY_USR_VIEW    0x00010000    /* user permissions... */
+#define KEY_USR_READ    0x00020000
+#define KEY_USR_WRITE   0x00040000
+#define KEY_USR_SEARCH  0x00080000
+#define KEY_USR_LINK    0x00100000
+  #define KEY_USR_ALL    0x001f0000

  #define KEY_GRP_VIEW    0x00000100    /* group permissions... */
@@ -221,14 +229,14 @@ extern struct key *request_key(struct ke

extern int key_validate(struct key *key);

-extern struct key *key_create_or_update(struct key *keyring,
-                                       const char *type,
-                                       const char *description,
-                                       const void *payload,
-                                       size_t plen,
-                                       int not_in_quota);
+extern struct key_ref *key_create_or_update(struct key_ref *keyring,
+                                           const char *type,
+                                           const char *description,
+                                           const void *payload,
+                                           size_t plen,
+                                           int not_in_quota);

-extern int key_update(struct key *key,
+extern int key_update(struct key_ref *key,
                       const void *payload,
                       size_t plen);

@@ -243,9 +251,9 @@ extern struct key *keyring_alloc(const c

extern int keyring_clear(struct key *keyring);

-extern struct key *keyring_search(struct key *keyring,
-                                  struct key_type *type,
-                                  const char *description);
+extern struct key_ref *keyring_search(struct key_ref *keyring,
+                                     struct key_type *type,
+                                     const char *description);

extern int keyring_add_key(struct key *keyring,
                           struct key *key);
@@ -257,6 +265,24 @@ extern void keyring_replace_payload(stru
  #define key_serial(key) ((key) ? (key)->serial : 0)

  /*
+ * key reference with possession flag handling
+ */
+static inline struct key_ref *key_mkref(const struct key *key, unsigned long possession)
+{
+    return (struct key_ref *) ((unsigned long) key | possession);
+}
+
+static inline struct key *key_deref(const struct key_ref *key)
+{
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
+     return (struct key *) ((unsigned long) key & ~1UL);
+}
+
+static inline unsigned long is_key_possestted(const struct key_ref *key)
+{
+     return (unsigned long) key & 1UL;
+}
+
+/*
+ * the userspace interface
+ */
extern struct key root_user_keyring, root_session_keyring;
@@ -285,6 +311,9 @@ extern void key_init(void);
#define key_serial(k)                0
#define key_get(k)                   ({ NULL; })
#define key_put(k)                   do { } while(0)
+#define key_mkref(k)                ({ NULL; })
+#define key_deref(k)                ({ NULL; })
+#define is_key_possestted(k)        0
#define alloc_uid_keyring(u)         0
#define switch_uid_keyring(u)       do { } while(0)
#define __install_session_keyring(t, k) ({ NULL; })
diff -uNrp linux-2.6.14-rc1-mm1/include/linux/key-ui.h linux-2.6.14-rc1-mm1-keys/include/linux/key-ui.h
--- linux-2.6.14-rc1-mm1/include/linux/key-ui.h 2005-08-30 13:56:36.000000000 +0100
+++ linux-2.6.14-rc1-mm1-keys/include/linux/key-ui.h 2005-09-21 12:00:23.000000000 +0100
@@ -42,11 +42,14 @@ struct keyring_list {
/*
 * check to see whether permission is granted to use a key in the desired way
 */
-static inline int key_permission(const struct key *key, key_perm_t perm)
+static inline int key_permission(const struct key_ref *key_ref, key_perm_t perm)
{
+     struct key *key = key_deref(key_ref);
+     key_perm_t kperm;

-     if (key->uid == current->fsuid)
+     if (is_key_possestted(key_ref))
+         kperm = key->perm >> 24;
+     else if (key->uid == current->fsuid)
+         kperm = key->perm >> 16;
+     else if (key->gid != -1 &&
+             key->perm & KEY_GRP_ALL &&
@@ -65,11 +68,14 @@ static inline int key_permission(const s
 * check to see whether permission is granted to use a key in at least one of
 * the desired ways
 */
-static inline int key_any_permission(const struct key *key, key_perm_t perm)
+static inline int key_any_permission(const struct key_ref *key_ref, key_perm_t perm)
{
+     struct key *key = key_deref(key_ref);
+     key_perm_t kperm;

-     if (key->uid == current->fsuid)
+     if (is_key_possestted(key_ref))
+         kperm = key->perm >> 24;
+     else if (key->uid == current->fsuid)
+         kperm = key->perm >> 16;
+     else if (key->gid != -1 &&
+             key->perm & KEY_GRP_ALL &&
@@ -94,13 +100,17 @@ static inline int key_task_groups_search
return ret;
}
}
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
-static inline int key_task_permission(const struct key *key,
+static inline int key_task_permission(const struct key_ref *key_ref,
                                     struct task_struct *context,
                                     key_perm_t perm)
{
+   struct key *key = key_deref(key_ref);
   key_perm_t kperm;

-   if (key->uid == context->fsuid) {
+   if (is_key_posessed(key_ref)) {
+       kperm = key->perm >> 24;
+   }
+   else if (key->uid == context->fsuid) {
+       kperm = key->perm >> 16;
+   }
+   else if (key->gid != -1 &&
@@ -121,9 +131,9 @@ static inline int key_task_permission(co
}

-extern struct key *lookup_user_key(struct task_struct *context,
-                                  key_serial_t id, int create, int partial,
-                                  key_perm_t perm);
+extern struct key_ref *lookup_user_key(struct task_struct *context,
+                                       key_serial_t id, int create, int partial,
+                                       key_perm_t perm);

extern long join_session_keyring(const char *name);

diff -uNrp linux-2.6.14-rc1-mm1/security/keys/internal.h linux-2.6.14-rc1-mm1-keys/security/keys/
--- linux-2.6.14-rc1-mm1/security/keys/internal.h      2005-08-30 13:56:44.000000000 +0100
+++ linux-2.6.14-rc1-mm1-keys/security/keys/internal.h 2005-09-21 14:59:27.000000000 +0100
@@ -71,26 +71,26 @@ extern void keyring_publish_name(struct

extern int __key_link(struct key *keyring, struct key *key);

-extern struct key *__keyring_search_one(struct key *keyring,
-                                        const struct key_type *type,
-                                        const char *description,
-                                        key_perm_t perm);
+extern struct key_ref *__keyring_search_one(struct key_ref *keyring,
+                                           const struct key_type *type,
+                                           const char *description,
+                                           key_perm_t perm);

extern struct key *keyring_search_instkey(struct key *keyring,
                                         key_serial_t target_id);

typedef int (*key_match_func_t)(const struct key *, const void *);

-extern struct key *keyring_search_aux(struct key *keyring,
-                                     struct task_struct *tsk,
-                                     struct key_type *type,
-                                     const void *description,
-                                     key_match_func_t match);
-
-extern struct key *search_process_keyrings(struct key_type *type,
-                                           const void *description,
-                                           key_match_func_t match,
-                                           struct task_struct *tsk);
+extern struct key_ref *keyring_search_aux(struct key_ref *keyring,
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
+         struct task_struct *tsk,
+         struct key_type *type,
+         const void *description,
+         key_match_func_t match);
+
+extern struct key_ref *search_process_keyrings(struct key_type *type,
+                                             const void *description,
+                                             key_match_func_t match,
+                                             struct task_struct *tsk);
+
extern struct key *find_keyring_by_name(const char *name, key_serial_t bound);

diff -uNrp linux-2.6.14-rc1-mm1/security/keys/key.c linux-2.6.14-rc1-mm1-keys/security/keys/key.c
--- linux-2.6.14-rc1-mm1/security/keys/key.c      2005-08-30 13:56:44.000000000 +0100
+++ linux-2.6.14-rc1-mm1-keys/security/keys/key.c  2005-09-21 13:47:47.000000000 +0100
@@ -693,14 +693,16 @@ void key_type_put(struct key_type *ktype
 * - the key has an incremented refcount
 * - we need to put the key if we get an error
 */
-static inline struct key *__key_update(struct key *key, const void *payload,
-                                     size_t plen)
+static inline struct key_ref *__key_update(struct key_ref *key_ref,
+                                          const void *payload,
+                                          size_t plen)
+{
+    struct key *key = key_deref(key_ref);
+    int ret;

    /* need write permission on the key to update it */
    ret = -EACCES;
-    if (!key_permission(key, KEY_WRITE))
+    if (!key_permission(key_ref, KEY_WRITE))
        goto error;

    ret = -EEXIST;
@@ -719,12 +721,12 @@ static inline struct key *__key_update(s

    if (ret < 0)
        goto error;
- out:
-    return key;
+out:
+    return key_ref;

- error:
+error:
    key_put(key);
-    key = ERR_PTR(ret);
+    key_ref = ERR_PTR(ret);
    goto out;

} /* end __key_update() */
@@ -734,52 +736,56 @@ static inline struct key *__key_update(s
 * search the specified keyring for a key of the same description; if one is
 * found, update it, otherwise add a new one
 */
-struct key *key_create_or_update(struct key *keyring,
-                                const char *type,
-                                const char *description,
-                                const void *payload,
-                                size_t plen,
-                                int not_in_quota)
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
+struct key_ref *key_create_or_update(struct key_ref *keyring_ref,
+                                   const char *type,
+                                   const char *description,
+                                   const void *payload,
+                                   size_t plen,
+                                   int not_in_quota)
{
    struct key_type *ktype;
-   struct key *key = NULL;
+   struct key_ref *key_ref;
+   struct key *keyring, *key = NULL;
    key_perm_t perm;
    int ret;

-   key_check(keyring);
-
    /* look up the key type to see if it's one of the registered kernel
     * types */
    ktype = key_type_lookup(type);
    if (IS_ERR(ktype)) {
-       key = ERR_PTR(-ENODEV);
+       key_ref = ERR_PTR(-ENODEV);
        goto error;
    }

-   ret = -EINVAL;
+   key_ref = ERR_PTR(-EINVAL);
    if (!ktype->match || !ktype->instantiate)
        goto error_2;

+   keyring = key_deref(keyring_ref);
+
+   key_check(keyring);
+
+   down_write(&keyring->sem);
+
+   /* if we're going to allocate a new key, we're going to have
+    * to modify the keyring */
+   key_ref = ERR_PTR(-EACCES);
+   if (!key_permission(keyring_ref, KEY_WRITE))
+       goto error_3;
+
    /* search for an existing key of the same type and description in the
     * destination keyring
     */
-   down_write(&keyring->sem);
-
-   key = __keyring_search_one(keyring, ktype, description, 0);
-   if (!IS_ERR(key))
+   key_ref = __keyring_search_one(keyring_ref, ktype, description, 0);
+   if (!IS_ERR(key_ref))
        goto found_matching_key;

-   /* if we're going to allocate a new key, we're going to have to modify
-    * the keyring */
-   ret = -EACCES;
-   if (!key_permission(keyring, KEY_WRITE))
-       goto error_3;
-
-   /* decide on the permissions we want */
-   perm = KEY_USR_VIEW | KEY_USR_SEARCH | KEY_USR_LINK;
+   perm = KEY_POS_VIEW | KEY_POS_SEARCH | KEY_POS_LINK;
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
+     perm |= KEY_USR_VIEW | KEY_USR_SEARCH | KEY_USR_LINK;

    if (ktype->read)
-         perm |= KEY_USR_READ;
+         perm |= KEY_POS_READ | KEY_USR_READ;

    if (ktype == &key_type_keyring || ktype->update)
        perm |= KEY_USR_WRITE;
@@ -788,7 +794,7 @@ struct key *key_create_or_update(struct
    key = key_alloc(ktype, description, current->fsuid, current->fsgid,
                    perm, not_in_quota);
    if (IS_ERR(key)) {
-        ret = PTR_ERR(key);
+        key_ref = ERR_PTR(PTR_ERR(key));
        goto error_3;
    }

@@ -796,15 +802,18 @@ struct key *key_create_or_update(struct
    ret = __key_instantiate_and_link(key, payload, plen, keyring, NULL);
    if (ret < 0) {
        key_put(key);
-        key = ERR_PTR(ret);
+        key_ref = ERR_PTR(ret);
+        goto error_3;
    }

+    key_ref = key_mkref(key, is_key_posessed(keyring_ref));
+
error_3:
    up_write(&keyring->sem);
error_2:
    key_type_put(ktype);
error:
-    return key;
+    return key_ref;

found_matching_key:
    /* we found a matching key, so we're going to try to update it
@@ -813,7 +822,7 @@ struct key *key_create_or_update(struct
    up_write(&keyring->sem);
    key_type_put(ktype);

-    key = __key_update(key, payload, plen);
+    key_ref = __key_update(key_ref, payload, plen);
    goto error;

} /* end key_create_or_update() */
@@ -824,15 +833,16 @@ EXPORT_SYMBOL(key_create_or_update);
/*
 * update a key
 */
-int key_update(struct key *key, const void *payload, size_t plen)
+int key_update(struct key_ref *key_ref, const void *payload, size_t plen)
{
+    struct key *key = key_deref(key_ref);
    int ret;

    key_check(key);

    /* the key must be writable */
    ret = -EACCES;
-    if (!key_permission(key, KEY_WRITE))
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
+     if (!key_permission(key_ref, KEY_WRITE))
+         goto error;

        /* attempt to update it if supported */
diff -uNrp linux-2.6.14-rc1-mm1/security/keys/keyctl.c linux-2.6.14-rc1-mm1-keys/security/keys/ke
--- linux-2.6.14-rc1-mm1-keys/keyctl.c 2005-08-30 13:56:44.000000000 +0100
+++ linux-2.6.14-rc1-mm1-keys/security/keys/keyctl.c 2005-09-21 13:44:18.000000000 +0100
@@ -34,7 +34,7 @@ asmlinkage long sys_add_key(const char _
                size_t plen,
                key_serial_t ringid)
{
-     struct key *keyring, *key;
+     struct key_ref *keyring, *key;
     char type[32], *description;
     void *payload;
     long dlen, ret;
@@ -97,14 +97,14 @@ asmlinkage long sys_add_key(const char _
     key = key_create_or_update(keyring, type, description,
                               payload, plen, 0);

     if (!IS_ERR(key)) {
-         ret = key->serial;
-         key_put(key);
+         ret = key_deref(key)->serial;
+         key_put(key_deref(key));
     }
     else {
         ret = PTR_ERR(key);
     }

-     key_put(keyring);
+     key_put(key_deref(keyring));
error3:
    kfree(payload);
error2:
@@ -131,7 +131,8 @@ asmlinkage long sys_request_key(const ch
                key_serial_t destringid)
{
    struct key_type *ktype;
-     struct key *key, *dest;
+     struct key_ref *dest;
+     struct key *key;
    char type[32], *description, *callout_info;
    long dlen, ret;

@@ -204,7 +205,8 @@ asmlinkage long sys_request_key(const ch
}

        /* do the search */
-     key = request_key_and_link(ktype, description, callout_info, dest);
+     key = request_key_and_link(ktype, description, callout_info,
+                               key_deref(dest));
+
    if (IS_ERR(key)) {
        ret = PTR_ERR(key);
        goto error5;
@@ -216,7 +218,7 @@ asmlinkage long sys_request_key(const ch
error5:
    key_type_put(ktype);
error4:
-     key_put(dest);
+     key_put(key_deref(dest));
error3:
    kfree(callout_info);
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
error2:
@@ -234,7 +236,7 @@ asmlinkage long sys_request_key(const ch
    */
    long keyctl_get_keyring_ID(key_serial_t id, int create)
    {
-       struct key *key;
+       struct key_ref *key;
        long ret;

        key = lookup_user_key(NULL, id, create, 0, KEY_SEARCH);
@@ -243,8 +245,8 @@ long keyctl_get_keyring_ID(key_serial_t
        goto error;
    }

-       ret = key->serial;
-       key_put(key);
+       ret = key_deref(key)->serial;
+       key_put(key_deref(key));
    error:
        return ret;

@@ -302,7 +304,7 @@ long keyctl_update_key(key_serial_t id,
        const void __user *_payload,
        size_t plen)
    {
-       struct key *key;
+       struct key_ref *key;
        void *payload;
        long ret;

@@ -333,7 +335,7 @@ long keyctl_update_key(key_serial_t id,
        /* update the key */
        ret = key_update(key, payload, plen);

-       key_put(key);
+       key_put(key_deref(key));
    error2:
        kfree(payload);
    error:
@@ -349,7 +351,7 @@ long keyctl_update_key(key_serial_t id,
    */
    long keyctl_revoke_key(key_serial_t id)
    {
-       struct key *key;
+       struct key_ref *key;
        long ret;

        key = lookup_user_key(NULL, id, 0, 0, KEY_WRITE);
@@ -358,10 +360,10 @@ long keyctl_revoke_key(key_serial_t id)
        goto error;
    }

-       key_revoke(key);
+       key_revoke(key_deref(key));
        ret = 0;

-       key_put(key);
+       key_put(key_deref(key));
    error:
        return ret;

@@ -375,7 +377,7 @@ long keyctl_revoke_key(key_serial_t id)
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
*/
long keyctl_keyring_clear(key_serial_t ringid)
{
-     struct key *keyring;
+     struct key_ref *keyring;
     long ret;

     keyring = lookup_user_key(NULL, ringid, 1, 0, KEY_WRITE);
@@ -384,9 +386,9 @@ long keyctl_keyring_clear(key_serial_t r
         goto error;
     }

-     ret = keyring_clear(keyring);
+     ret = keyring_clear(key_deref(keyring));

-     key_put(keyring);
+     key_put(key_deref(keyring));
error:
    return ret;

@@ -401,7 +403,7 @@ long keyctl_keyring_clear(key_serial_t r
*/
long keyctl_keyring_link(key_serial_t id, key_serial_t ringid)
{
-     struct key *keyring, *key;
+     struct key_ref *keyring, *key;
     long ret;

     keyring = lookup_user_key(NULL, ringid, 1, 0, KEY_WRITE);
@@ -416,11 +418,11 @@ long keyctl_keyring_link(key_serial_t id
        goto error2;
    }

-     ret = key_link(keyring, key);
+     ret = key_link(key_deref(keyring), key_deref(key));

-     key_put(key);
+     key_put(key_deref(key));
error2:
-     key_put(keyring);
+     key_put(key_deref(keyring));
error:
    return ret;

@@ -435,7 +437,7 @@ long keyctl_keyring_link(key_serial_t id
*/
long keyctl_keyring_unlink(key_serial_t id, key_serial_t ringid)
{
-     struct key *keyring, *key;
+     struct key_ref *keyring, *key;
     long ret;

     keyring = lookup_user_key(NULL, ringid, 0, 0, KEY_WRITE);
@@ -450,11 +452,11 @@ long keyctl_keyring_unlink(key_serial_t
        goto error2;
    }

-     ret = key_unlink(keyring, key);
+     ret = key_unlink(key_deref(keyring), key_deref(key));

-     key_put(key);
+     key_put(key_deref(key));
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
error2:
-   key_put(keyring);
+   key_put(key_deref(keyring));
error:
    return ret;

@@ -475,7 +477,8 @@ long keyctl_describe_key(key_serial_t ke
        char __user *buffer,
        size_t buflen)
{
-   struct key *key, *instkey;
+   struct key_ref *key;
+   struct key *instkey;
    char *tmpbuf;
    long ret;

@@ -505,12 +508,12 @@ okay:
        goto error2;

    ret = snprintf(tmpbuf, PAGE_SIZE - 1,
-               "%s;%d;%d;%06x;%s",
-               key->type->name,
-               key->uid,
-               key->gid,
-               key->perm,
-               key->description ? key->description : ""
+               "%s;%d;%d;%08x;%s",
+               key_deref(key)->type->name,
+               key_deref(key)->uid,
+               key_deref(key)->gid,
+               key_deref(key)->perm,
+               key_deref(key)->description ? key_deref(key)->description : ""
    );

    /* include a NUL char at the end of the data */
@@ -530,7 +533,7 @@ okay:

    kfree(tmpbuf);
error2:
-   key_put(key);
+   key_put(key_deref(key));
error:
    return ret;

@@ -552,7 +555,7 @@ long keyctl_keyring_search(key_serial_t
        key_serial_t destringid)
{
    struct key_type *ktype;
-   struct key *keyring, *key, *dest;
+   struct key_ref *keyring, *key, *dest;
    char type[32], *description;
    long dlen, ret;

@@ -621,21 +624,21 @@ long keyctl_keyring_search(key_serial_t
        if (!key_permission(key, KEY_LINK))
            goto error6;

-       ret = key_link(dest, key);
+       ret = key_link(key_deref(dest), key_deref(key));
        if (ret < 0)
            goto error6;
    }
}
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
-     ret = key->serial;
+     ret = key_deref(key)->serial;

error6:
-     key_put(key);
+     key_put(key_deref(key));
error5:
     key_type_put(ktype);
error4:
-     key_put(dest);
+     key_put(key_deref(dest));
error3:
-     key_put(keyring);
+     key_put(key_deref(keyring));
error2:
     kfree(description);
error:
@@ -645,16 +648,6 @@ long keyctl_keyring_search(key_serial_t

/*****
/*
- * see if the key we're looking at is the target key
- */
-static int keyctl_read_key_same(const struct key *key, const void *target)
-{
-     return key == target;
-
-} /* end keyctl_read_key_same() */
-
-/**
* read a user key's payload
* - the keyring must be readable or the key must be searchable from the
*   process's keyrings
@@ -665,38 +658,34 @@ static int keyctl_read_key_same(const st
*/
long keyctl_read_key(key_serial_t keyid, char __user *buffer, size_t buflen)
{
-     struct key *key, *skey;
+     struct key_ref *key_ref;
+     struct key *key;
     long ret;

     /* find the key first */
-     key = lookup_user_key(NULL, keyid, 0, 0, 0);
-     if (!IS_ERR(key)) {
-         /* see if we can read it directly */
-         if (key_permission(key, KEY_READ))
-             goto can_read_key;
-
-         /* we can't; see if it's searchable from this process's
-          * keyrings
-          * - we automatically take account of the fact that it may be
-          *   dangling off an instantiation key
-          */
-         skey = search_process_keyrings(key->type, key,
-                                       keyctl_read_key_same, current);
-         if (!IS_ERR(skey))
-             goto can_read_key2;
+     key_ref = lookup_user_key(NULL, keyid, 0, 0, 0);
+     if (IS_ERR(key_ref)) {
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
+         ret = -ENOKEY;
+         goto error;
+     }

-         ret = PTR_ERR(skey);
-         if (ret == -EAGAIN)
-             ret = -EACCES;
+     key = key_deref(key_ref);
+
+     /* see if we can read it directly */
+     if (key_permission(key_ref, KEY_READ))
+         goto can_read_key;
+
+     /* we can't; see if it's searchable from this process's
+      * keyrings
+      * - we automatically take account of the fact that it may be
+      *   dangling off an instantiation key
+      */
+     if (!is_key_posessed(key_ref)) {
+         ret = -EACCES;
+         goto error2;
+     }

-     ret = -ENOKEY;
-     goto error;
-
-     /* the key is probably readable - now try to read it */
- can_read_key2:
-     key_put(skey);
- can_read_key:
-     ret = key_validate(key);
-     if (ret == 0) {
@@ -726,6 +715,7 @@ long keyctl_read_key(key_serial_t keyid,
-     */
-     long keyctl_chown_key(key_serial_t id, uid_t uid, gid_t gid)
-     {
+         struct key_ref *key_ref;
+         struct key *key;
+         long ret;

@@ -733,12 +723,14 @@ long keyctl_chown_key(key_serial_t id, u
-     if (uid == (uid_t) -1 && gid == (gid_t) -1)
-         goto error;

-     key = lookup_user_key(NULL, id, 1, 1, 0);
-     if (IS_ERR(key)) {
-         ret = PTR_ERR(key);
+     key_ref = lookup_user_key(NULL, id, 1, 1, 0);
+     if (IS_ERR(key_ref)) {
+         ret = PTR_ERR(key_ref);
+         goto error;
+     }

+     key = key_deref(key_ref);
+
+     /* make the changes with the locks held to prevent chown/chown races */
+     ret = -EACCES;
+     down_write(&key->sem);
@@ -783,19 +775,22 @@ long keyctl_chown_key(key_serial_t id, u
-     */
-     long keyctl_setperm_key(key_serial_t id, key_perm_t perm)
-     {
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
+ struct key_ref *key_ref;
+ struct key *key;
+ long ret;

- ret = -EINVAL;
- if (perm & ~(KEY_USR_ALL | KEY_GRP_ALL | KEY_OTH_ALL))
+ if (perm & ~(KEY_POS_ALL | KEY_USR_ALL | KEY_GRP_ALL | KEY_OTH_ALL))
+     goto error;

- key = lookup_user_key(NULL, id, 1, 1, 0);
- if (IS_ERR(key)) {
-     ret = PTR_ERR(key);
+ key_ref = lookup_user_key(NULL, id, 1, 1, 0);
+ if (IS_ERR(key_ref)) {
+     ret = PTR_ERR(key_ref);
+     goto error;
+ }

+ key = key_deref(key_ref);
+
+ /* make the changes with the locks held to prevent chown/chmod races */
+ ret = -EACCES;
+ down_write(&key->sem);
@@ -824,7 +819,8 @@ long keyctl_instantiate_key(key_serial_t
+     key_serial_t ringid)
+ {
+     struct request_key_auth *rka;
-     struct key *instkey, *keyring;
+     struct key_ref *keyring;
+     struct key *instkey;
+     void *payload;
+     long ret;

@@ -869,9 +865,9 @@ long keyctl_instantiate_key(key_serial_t
+
+     /* instantiate the key and link it into a keyring */
+     ret = key_instantiate_and_link(rka->target_key, payload, plen,
-         keyring, instkey);
+         key_deref(keyring), instkey);

-     key_put(keyring);
+     key_put(key_deref(keyring));
+ error3:
+     key_put(instkey);
+ error2:
@@ -889,7 +885,8 @@ long keyctl_instantiate_key(key_serial_t
+ long keyctl_negate_key(key_serial_t id, unsigned timeout, key_serial_t ringid)
+ {
+     struct request_key_auth *rka;
-     struct key *instkey, *keyring;
+     struct key_ref *keyring;
+     struct key *instkey;
+     long ret;

+     /* find the instantiation authorisation key */
@@ -913,9 +910,10 @@ long keyctl_negate_key(key_serial_t id,
+ }

+     /* instantiate the key and link it into a keyring */
-     ret = key_negate_and_link(rka->target_key, timeout, keyring, instkey);
+     ret = key_negate_and_link(rka->target_key, timeout, key_deref(keyring),
+         instkey);
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
-     key_put(keyring);
+     key_put(key_deref(keyring));
error2:
    key_put(instkey);
error:
diff -uNrp linux-2.6.14-rc1-mm1/security/keys/keyring.c linux-2.6.14-rc1-mm1-keys/security/keys/k
--- linux-2.6.14-rc1-mm1/security/keys/keyring.c      2005-08-30 13:56:44.000000000 +0100
+++ linux-2.6.14-rc1-mm1-keys/security/keys/keyring.c  2005-09-21 14:34:02.000000000 +0100
@@ -309,7 +309,7 @@ struct key *keyring_alloc(const char *de
    int ret;

    keyring = key_alloc(&key_type_keyring, description,
-                       uid, gid, KEY_USR_ALL, not_in_quota);
+                       uid, gid, KEY_POS_ALL | KEY_USR_ALL, not_in_quota);

    if (!IS_ERR(keyring)) {
        ret = key_instantiate_and_link(keyring, NULL, 0, dest, NULL);
@@ -333,12 +333,13 @@ struct key *keyring_alloc(const char *de
 * - we rely on RCU to prevent the keyring lists from disappearing on us
 * - we return -EAGAIN if we didn't find any matching key
 * - we return -ENOKEY if we only found negative matching keys
+ * - we propagate the possession attribute from the keyring ref to the key ref
 */
-struct key *keyring_search_aux(struct key *keyring,
-                               struct task_struct *context,
-                               struct key_type *type,
-                               const void *description,
-                               key_match_func_t match)
+struct key_ref *keyring_search_aux(struct key_ref *keyring_ref,
+                                   struct task_struct *context,
+                                   struct key_type *type,
+                                   const void *description,
+                                   key_match_func_t match)
    {
        struct {
            struct keyring_list *keylist;
@@ -347,29 +348,33 @@ struct key *keyring_search_aux(struct ke
        struct timespec now;
-       struct key *key;
+       struct key_ref *key_ref;
+       unsigned long possessed;
+       struct key *keyring, *key;
        long err;
        int sp, kix;

+       keyring = key_deref(keyring_ref);
+       possessed = is_key_posessed(keyring_ref);
        key_check(keyring);

-       rcu_read_lock();
-
        /* top keyring must have search permission to begin the search */
-       key = ERR_PTR(-EACCES);
-       if (!key_task_permission(keyring, context, KEY_SEARCH))
+       key_ref = ERR_PTR(-EACCES);
+       if (!key_task_permission(keyring_ref, context, KEY_SEARCH))
            goto error;

-       key = ERR_PTR(-ENOTDIR);
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
+     key_ref = ERR_PTR(-ENOTDIR);
+     if (keyring->type != &key_type_keyring)
+         goto error;

+     rcu_read_lock();
+
+     now = current_kernel_time();
+     err = -EAGAIN;
+     sp = 0;

+     /* start processing a new keyring */
- descend:
+descend:
+     if (test_bit(KEY_FLAG_REVOKED, &keyring->flags))
+         goto not_this_keyring;

@@ -397,7 +402,8 @@ struct key *keyring_search_aux(struct ke
+         continue;

+         /* key must have search permissions */
-         if (!key_task_permission(key, context, KEY_SEARCH))
+         if (!key_task_permission(key_mkref(key, possessed),
+             context, KEY_SEARCH))
+             continue;

+         /* we set a different error code if we find a negative key */
@@ -411,7 +417,7 @@ struct key *keyring_search_aux(struct ke
+         /* search through the keyrings nested in this one */
+         kix = 0;
- ascend:
+ascend:
+         for (; kix < keylist->nkeys; kix++) {
+             key = keylist->keys[kix];
+             if (key->type != &key_type_keyring)
@@ -423,7 +429,8 @@ struct key *keyring_search_aux(struct ke
+                 if (sp >= KEYRING_SEARCH_MAX_DEPTH)
+                     continue;

-                 if (!key_task_permission(key, context, KEY_SEARCH))
+                 if (!key_task_permission(key_mkref(key, possessed),
+                     context, KEY_SEARCH))
+                     continue;

+                 /* stack the current position */
@@ -438,7 +445,7 @@ struct key *keyring_search_aux(struct ke
+                 /* the keyring we're looking at was disqualified or didn't contain a
+                  * matching key */
- not_this_keyring:
+not_this_keyring:
+                 if (sp > 0) {
+                     /* resume the processing of a keyring higher up in the tree */
+                     sp--;
@@ -447,16 +454,18 @@ struct key *keyring_search_aux(struct ke
+                     goto ascend;
+                 }

-         key = ERR_PTR(err);
-         goto error;
+         key_ref = ERR_PTR(err);
+         goto error_2;
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```

        /* we found a viable match */
- found:
+found:
        atomic_inc(&key->usage);
        key_check(key);
- error:
+
        key_ref = key_mkref(key, possessed);
+error_2:
        rcu_read_unlock();
-
        return key;
+error:
+
        return key_ref;

} /* end keyring_search_aux() */

@@ -469,9 +478,9 @@ struct key *keyring_search_aux(struct ke
 * - we return -EAGAIN if we didn't find any matching key
 * - we return -ENOKEY if we only found negative matching keys
 */
-struct key *keyring_search(struct key *keyring,
-
-        struct key_type *type,
-        const char *description)
+struct key_ref *keyring_search(struct key_ref *keyring,
+
+        struct key_type *type,
+        const char *description)
{
        if (!type->match)
                return ERR_PTR(-ENOKEY);
@@ -488,15 +497,19 @@ EXPORT_SYMBOL(keyring_search);
 * search the given keyring only (no recursion)
 * - keyring must be locked by caller
 */
-struct key *__keyring_search_one(struct key *keyring,
-
-        const struct key_type *ktype,
-        const char *description,
-        key_perm_t perm)
+struct key_ref *__keyring_search_one(struct key_ref *keyring_ref,
+
+        const struct key_type *ktype,
+        const char *description,
+        key_perm_t perm)
{
        struct keyring_list *klist;
-
        struct key *key;
+
        unsigned long possessed;
+
        struct key *keyring, *key;
        int loop;

+
        keyring = key_deref(keyring_ref);
+
        possessed = is_key_possestted(keyring_ref);
+
        rcu_read_lock();

        klist = rcu_dereference(keyring->payload.subscriptions);
@@ -507,21 +520,20 @@ struct key *__keyring_search_one(struct
        if (key->type == ktype &&
            (!key->type->match ||
             key->type->match(key, description)) &&
-
            key_permission(key, perm) &&
+
            key_permission(key_mkref(key, possessed), perm) &&
            !test_bit(KEY_FLAG_REVOKED, &key->flags)
        )

```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
                goto found;
            }
        }

-       key = ERR_PTR(-ENOKEY);
-       goto error;
+       rcu_read_unlock();
+       return ERR_PTR(-ENOKEY);

found:
    atomic_inc(&key->usage);
- error:
    rcu_read_unlock();
-    return key;
+    return key_mkref(key, possessed);

} /* end __keyring_search_one() */

@@ -603,7 +615,7 @@ struct key *find_keyring_by_name(const c
                if (strcmp(keyring->description, name) != 0)
                    continue;

-                if (!key_permission(keyring, KEY_SEARCH))
+                if (!key_permission(key_mkref(keyring, 0), KEY_SEARCH))
                    continue;

                /* found a potential candidate, but we still need to
diff -uNrp linux-2.6.14-rc1-mml/security/keys/proc.c linux-2.6.14-rc1-mml-keys/security/keys/proc.c
--- linux-2.6.14-rc1-mml/security/keys/proc.c    2005-08-30 13:56:44.000000000 +0100
+++ linux-2.6.14-rc1-mml-keys/security/keys/proc.c    2005-09-21 13:35:36.000000000 +0100
@@ -167,7 +167,7 @@ static int proc_keys_show(struct seq_fil
#define showflag(KEY, LETTER, FLAG) \
    (test_bit(FLAG, &(KEY)->flags) ? LETTER : '-')

-    seq_printf(m, "%08x %c%c%c%c%c%c %5d %4s %06x %5d %5d %-9.9s ",
+    seq_printf(m, "%08x %c%c%c%c%c%c %5d %4s %08x %5d %5d %-9.9s ",
                key->serial,
                showflag(key, 'I', KEY_FLAG_INSTANTIATED),
                showflag(key, 'R', KEY_FLAG_REVOKED),
diff -uNrp linux-2.6.14-rc1-mml/security/keys/process_keys.c linux-2.6.14-rc1-mml-keys/security/keys/process_keys.c
--- linux-2.6.14-rc1-mml/security/keys/process_keys.c    2005-08-30 13:56:44.000000000 +0100
+++ linux-2.6.14-rc1-mml-keys/security/keys/process_keys.c    2005-09-21 13:45:22.000000000 +0100
@@ -39,7 +39,7 @@ struct key root_user_keyring = {
    .type          = &key_type_keyring,
    .user          = &root_key_user,
    .sem          = __RWSEM_INITIALIZER(root_user_keyring.sem),
-   .perm         = KEY_USR_ALL,
+   .perm         = KEY_POS_ALL | KEY_USR_ALL,
    .flags        = 1 << KEY_FLAG_INSTANTIATED,
    .description   = "_uid.0",
#ifdef KEY_DEBUGGING
@@ -54,7 +54,7 @@ struct key root_session_keyring = {
    .type          = &key_type_keyring,
    .user          = &root_key_user,
    .sem          = __RWSEM_INITIALIZER(root_session_keyring.sem),
-   .perm         = KEY_USR_ALL,
+   .perm         = KEY_POS_ALL | KEY_USR_ALL,
    .flags        = 1 << KEY_FLAG_INSTANTIATED,
    .description   = "_uid_ses.0",
#ifdef KEY_DEBUGGING
@@ -98,7 +98,7 @@ int alloc_uid_keyring(struct user_struct
    user->session_keyring = session_keyring;
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
ret = 0;

- error:
+error:
    return ret;

} /* end alloc_uid_keyring() */
@@ -156,7 +156,7 @@ int install_thread_keyring(struct task_s
    ret = 0;

    key_put(old);
- error:
+error:
    return ret;

} /* end install_thread_keyring() */
@@ -193,7 +193,7 @@ int install_process_keyring(struct task_
}

ret = 0;
- error:
+error:
    return ret;

} /* end install_process_keyring() */
@@ -236,7 +236,7 @@ static int install_session_keyring(struc
/* we're using RCU on the pointer */
synchronize_rcu();
key_put(old);
- error:
+error:
    return ret;

} /* end install_session_keyring() */
@@ -376,13 +376,13 @@ void key_fsgid_changed(struct task_struct
* - we return -EAGAIN if we didn't find any matching key
* - we return -ENOKEY if we found only negative matching keys
*/
-struct key *search_process_keyrings(struct key_type *type,
-                                   const void *description,
-                                   key_match_func_t match,
-                                   struct task_struct *context)
+struct key_ref *search_process_keyrings(struct key_type *type,
+                                       const void *description,
+                                       key_match_func_t match,
+                                       struct task_struct *context)
{
    struct request_key_auth *rka;
-   struct key *key, *ret, *err, *instkey;
+   struct key_ref *key, *ret, *err, *instkey;

    /* we want to return -EAGAIN or -ENOKEY if any of the keyrings were
     * searchable, but we failed to find a key or we found a negative key;
@@ -397,7 +397,7 @@ struct key *search_process_keyrings(stru

    /* search the thread keyring first */
    if (context->thread_keyring) {
-       key = keyring_search_aux(context->thread_keyring,
+       key = keyring_search_aux(key_mkref(context->thread_keyring, 1),
                                context, type, description, match);
        if (!IS_ERR(key))
            goto found;
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
@@ -417,8 +417,9 @@ struct key *search_process_keyrings(stru
    /* search the process keyring second */
    if (context->signal->process_keyring) {
-       key = keyring_search_aux(context->signal->process_keyring,
-                               context, type, description, match);
+       key = keyring_search_aux(
+           key_mkref(context->signal->process_keyring, 1),
+           context, type, description, match);
        if (!IS_ERR(key))
            goto found;

@@ -439,7 +440,7 @@ struct key *search_process_keyrings(stru
    if (context->signal->session_keyring) {
        rcu_read_lock();
        key = keyring_search_aux(
-           rcu_dereference(context->signal->session_keyring),
+           key_mkref(rcu_dereference(context->signal->session_keyring), 1),
            context, type, description, match);
        rcu_read_unlock();

@@ -466,18 +467,18 @@ struct key *search_process_keyrings(stru
        rcu_read_lock();
        instkey = __keyring_search_one(
-           rcu_dereference(context->signal->session_keyring),
+           key_mkref(rcu_dereference(context->signal->session_keyring), 1),
            &key_type_request_key_auth, NULL, 0);
        rcu_read_unlock();

        if (IS_ERR(instkey))
            goto no_key;

-       rka = instkey->payload.data;
+       rka = key_deref(instkey)->payload.data;

        key = search_process_keyrings(type, description, match,
                                     rka->context);
-       key_put(instkey);
+       key_put(key_deref(instkey));

        if (!IS_ERR(key))
            goto found;
@@ -496,8 +497,9 @@ struct key *search_process_keyrings(stru
    }
    /* or search the user-session keyring */
    else {
-       key = keyring_search_aux(context->user->session_keyring,
-                               context, type, description, match);
+       key = keyring_search_aux(
+           key_mkref(context->user->session_keyring, 1),
+           context, type, description, match);
        if (!IS_ERR(key))
            goto found;

@@ -526,20 +528,31 @@ found:
    /*
+ * see if the key we're looking at is the target key
+ */
+static int lookup_user_key_possestted(const struct key *key, const void *target)
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
+{
+    return key == target;
+
+} /* end lookup_user_key_possestted() */
+
+/*
+*****
+
+ * lookup a key given a key ID from userspace with a given permissions mask
+ * - don't create special keyrings unless so requested
+ * - partially constructed keys aren't found unless requested
+ */
-struct key *lookup_user_key(struct task_struct *context, key_serial_t id,
-                             int create, int partial, key_perm_t perm)
+struct key_ref *lookup_user_key(struct task_struct *context, key_serial_t id,
+                                int create, int partial, key_perm_t perm)
+
+{
+    struct key_ref *key_ref, *skey;
+    struct key *key;
+    int ret;
+
+    if (!context)
+        context = current;
+
+    key = ERR_PTR(-ENOKEY);
+    key_ref = ERR_PTR(-ENOKEY);
+
+    switch (id) {
+    case KEY_SPEC_THREAD_KEYRING:
+@@ -556,6 +569,7 @@ struct key *lookup_user_key(struct task_
+
+        key = context->thread_keyring;
+        atomic_inc(&key->usage);
+    +
+        key_ref = key_mkref(key, 1);
+        break;
+
+    case KEY_SPEC_PROCESS_KEYRING:
+@@ -572,6 +586,7 @@ struct key *lookup_user_key(struct task_
+
+        key = context->signal->process_keyring;
+        atomic_inc(&key->usage);
+    +
+        key_ref = key_mkref(key, 1);
+        break;
+
+    case KEY_SPEC_SESSION_KEYRING:
+@@ -579,7 +594,7 @@ struct key *lookup_user_key(struct task_
+        /* always install a session keyring upon access if one
+         * doesn't exist yet */
+        ret = install_session_keyring(
+    -
+            context, context->user->session_keyring);
+    +
+            context, context->user->session_keyring);
+        if (ret < 0)
+            goto error;
+    }
+@@ -588,16 +603,19 @@ struct key *lookup_user_key(struct task_
+        key = rcu_dereference(context->signal->session_keyring);
+        atomic_inc(&key->usage);
+        rcu_read_unlock();
+    +
+        key_ref = key_mkref(key, 1);
+        break;
+
+    case KEY_SPEC_USER_KEYRING:
+        key = context->user->uid_keyring;
```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
        atomic_inc(&key->usage);
+       key_ref = key_mkref(key, 1);
        break;

    case KEY_SPEC_USER_SESSION_KEYRING:
        key = context->user->session_keyring;
        atomic_inc(&key->usage);
+       key_ref = key_mkref(key, 1);
        break;

    case KEY_SPEC_GROUP_KEYRING:
@@ -606,13 +624,28 @@ struct key *lookup_user_key(struct task_
        goto error;

    default:
-       key = ERR_PTR(-EINVAL);
+       key_ref = ERR_PTR(-EINVAL);
        if (id < 1)
            goto error;

        key = key_lookup(id);
-       if (IS_ERR(key))
+       if (IS_ERR(key)) {
+           key_ref = ERR_PTR(PTR_ERR(key));
            goto error;
+       }

+       key_ref = key_mkref(key, 0);
+
+       /* check to see if we possess the key */
+       skey = search_process_keyrings(key->type, key,
+                                     lookup_user_key_posessed,
+                                     current);
+
+       if (!IS_ERR(skey)) {
+           key_put(key);
+           key_ref = skey;
+       }
+
        break;
    }

@@ -630,15 +663,15 @@ struct key *lookup_user_key(struct task_
    /* check the permissions */
    ret = -EACCES;

-    if (!key_task_permission(key, context, perm))
+    if (!key_task_permission(key_ref, context, perm))
        goto invalid_key;

- error:
-     return key;
+error:
+     return key_ref;

- invalid_key:
-     key_put(key);
-     key = ERR_PTR(ret);
+invalid_key:
+     key_put(key_deref(key_ref));
+     key_ref = ERR_PTR(ret);
    goto error;

```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```

    } /* end lookup_user_key() */
@@ -694,9 +727,9 @@ long join_session_keyring(const char *na
    ret = keyring->serial;
    key_put(keyring);

- error2:
+error2:
    up(&key_session_sem);
- error:
+error:
    return ret;

} /* end join_session_keyring() */
diff -uNrp linux-2.6.14-rc1-mm1/security/keys/request_key_auth.c linux-2.6.14-rc1-mm1-keys/security/keys/request_key_auth.c
--- linux-2.6.14-rc1-mm1/security/keys/request_key_auth.c      2005-08-30 13:56:44.000000000 +0100
+++ linux-2.6.14-rc1-mm1-keys/security/keys/request_key_auth.c 2005-09-21 13:48:24.000000000 +0100
@@ -126,7 +126,7 @@ struct key *request_key_auth_new(struct

    rkakey = key_alloc(&key_type_request_key_auth, desc,
                      current->fsuid, current->fsgid,
-                      KEY_USR_VIEW, 1);
+                      KEY_POS_VIEW | KEY_USR_VIEW, 1);
    if (IS_ERR(rkakey)) {
        key_put(keyring);
        kleave("= %ld", PTR_ERR(rkakey));
diff -uNrp linux-2.6.14-rc1-mm1/security/keys/request_key.c linux-2.6.14-rc1-mm1-keys/security/keys/request_key.c
--- linux-2.6.14-rc1-mm1/security/keys/request_key.c          2005-08-30 13:56:44.000000000 +0100
+++ linux-2.6.14-rc1-mm1-keys/security/keys/request_key.c     2005-09-21 15:22:05.000000000 +0100
@@ -129,7 +129,7 @@ static struct key *__request_key_constru

    /* create a key and add it to the queue */
    key = key_alloc(type, description,
-                  current->fsuid, current->fsgid, KEY_USR_ALL, 0);
+                  current->fsuid, current->fsgid, KEY_POS_ALL, 0);
    if (IS_ERR(key))
        goto alloc_failed;

@@ -364,15 +364,25 @@ struct key *request_key_and_link(struct
                                struct key *dest_keyring)
    {
        struct key_user *user;
+
        struct key_ref *key_ref;
        struct key *key;

        kenter("%s,%s,%s,%p",
              type->name, description, callout_info, dest_keyring);

        /* search all the process keyrings for a key */
-       key = search_process_keyrings(type, description, type->match, current);
+       key_ref = search_process_keyrings(type, description, type->match,
+                                         current);
+
+       if (PTR_ERR(key) == -EAGAIN) {
+           kdebug("search 1: %p", key_ref);
+
+           if (!IS_ERR(key_ref)) {
+               key = key_deref(key_ref);
+           }
+           else if (PTR_ERR(key_ref) != -EAGAIN) {
+               key = ERR_PTR(PTR_ERR(key_ref));
+           }
        }
    }

```

Linux-Kernel: [PATCH] Keys: Add possessor permissions to keys

```
+     else {
+         /* the search failed, but the keyrings were searchable, so we
+          * should consult userspace if we can */
+         key = ERR_PTR(-ENOKEY);
@@ -384,7 +394,7 @@ struct key *request_key_and_link(struct
+         if (!user)
+             goto nomem;
-         do {
+         for (;;) {
+             if (signal_pending(current))
+                 goto interrupted;
@@ -397,10 +407,20 @@ struct key *request_key_and_link(struct
+
+         /* someone else made the key we want, so we need to
+          * search again as it might now be available to us */
-         key = search_process_keyrings(type, description,
-                                     type->match, current);
-
-         } while (PTR_ERR(key) == -EAGAIN);
+         key_ref = search_process_keyrings(type, description,
+                                     type->match,
+                                     current);
+
+         kdebug("search 2: %p", key_ref);
+
+         if (PTR_ERR(key_ref) != -EAGAIN) {
+             if (IS_ERR(key_ref))
+                 key = key_deref(key_ref);
+             else
+                 key = ERR_PTR(PTR_ERR(key_ref));
+             break;
+         }
+
+         key_user_put(user);
```

-
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@vger.kernel.org

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>