
Subject: [PATCH v6 03/10] ipc: segment key change helper introduced

Posted by [Stanislav Kinsbursky](#) on Mon, 15 Oct 2012 15:59:51 GMT

[View Forum Message](#) <> [Reply to Message](#)

This patch introduces existent segment key changing infrastructure.

New function `ipc_update_key()` can be used change segment key, cuid, cgid values. It checks for that new key is not used (except `IPC_PRIVATE`) prior to set it on existent.

To make this possible, added copying of this fields from user-space in `__get_compat_ipc_perm()` and `__get_compat_ipc64_perm()` functions. Also segment search by key and lock were splitted into different functions, because `ipc_update_key()` doesn't need to lock the segment during check that new key is not used.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

```
ipc/compat.c | 6 ++++++
ipc/util.c   | 51 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ipc/util.h   | 2 ++
3 files changed, 56 insertions(+), 3 deletions(-)
```

diff --git a/ipc/compat.c b/ipc/compat.c

index ad9518e..af30d13 100644

--- a/ipc/compat.c

+++ b/ipc/compat.c

```
@@ -144,6 +144,9 @@ static inline int __get_compat_ipc64_perm(struct ipc64_perm *p64,
    err = __get_user(p64->uid, &up64->uid);
    err |= __get_user(p64->gid, &up64->gid);
    err |= __get_user(p64->mode, &up64->mode);
+ err |= __get_user(p64->cuid, &up64->cuid);
+ err |= __get_user(p64->cgid, &up64->cgid);
+ err |= __get_user(p64->key, &up64->key);
    return err;
}
```

```
@@ -155,6 +158,9 @@ static inline int __get_compat_ipc_perm(struct ipc64_perm *p,
    err = __get_user(p->uid, &up->uid);
    err |= __get_user(p->gid, &up->gid);
    err |= __get_user(p->mode, &up->mode);
+ err |= __get_user(p->cuid, &up->cuid);
+ err |= __get_user(p->cgid, &up->cgid);
+ err |= __get_user(p->key, &up->key);
    return err;
}
```

diff --git a/ipc/util.c b/ipc/util.c

index 503946e..faae296 100644

--- a/ipc/util.c

```

+++ b/ipc/util.c
@@ -173,7 +173,7 @@ void __init ipc_init_proc_interface(const char *path, const char *header,
 * @key: The key to find
 *
 * Requires ipc_ids.rw_mutex locked.
- * Returns the LOCKED pointer to the ipc structure if found or NULL
+ * Returns the UNLOCKED pointer to the ipc structure if found or NULL
 * if not.
 * If key is found ipc points to the owning ipc structure
 */
@@ -195,7 +195,6 @@ static struct kern_ipc_perm *ipc_findkey(struct ipc_ids *ids, key_t key)
    continue;
}

- ipc_lock_by_ptr(ipc);
    return ipc;
}

@@ -203,6 +202,27 @@ static struct kern_ipc_perm *ipc_findkey(struct ipc_ids *ids, key_t key)
}

/**
+ * ipc_findkey_locked - find and lock a key in an ipc identifier set
+ * @ids: Identifier set
+ * @key: The key to find
+ *
+ * Requires ipc_ids.rw_mutex locked.
+ * Returns the LOCKED pointer to the ipc structure if found or NULL
+ * if not.
+ * If key is found ipc points to the owning ipc structure
+ */
+
+static struct kern_ipc_perm *ipc_findkey_locked(struct ipc_ids *ids, key_t key)
+{
+ struct kern_ipc_perm *ipc;
+
+ ipc = ipc_findkey(ids, key);
+ if (ipc)
+ ipc_lock_by_ptr(ipc);
+ return ipc;
+}
+
+/**
 * ipc_get_maxid - get the last assigned id
 * @ids: IPC identifier set
 *
@@ -388,7 +408,7 @@ retry:
 * a new entry + read locks are not "upgradable"

```

```

    */
    down_write(&ids->rw_mutex);
- ipc = ipc_findkey(ids, params->key);
+ ipc = ipc_findkey_locked(ids, params->key);
    if (ipc == NULL) {
        /* key not used */
        if (!(flag & IPC_CREAT))
@@ -755,6 +775,31 @@ int ipcget(struct ipc_namespace *ns, struct ipc_ids *ids,
    }

/**
+ * ipc_update_key - update the key of an IPC.
+ * @in: the permission given as input.
+ * @out: the permission of the ipc to set.
+ *
+ * Common routine called by sys_shmctl(), sys_semctl(), sys_msgctl().
+ */
+int ipc_update_key(struct ipc_ids *ids, struct ipc64_perm *in,
+    struct kern_ipc_perm *out)
+{
+
+ if (in->key && out->key != in->key) {
+     /*
+      * Check for existent segment with the same key.
+      * Note: ipc_ids.rw_mutex is taken for write already.
+      */
+     if (ipc_findkey(ids, in->key))
+         return -EEXIST;
+     }
+     out->cuid = in->cuid;
+     out->cgid = in->cgid;
+     out->key = in->key;
+     return 0;
+ }
+
+/**
+ * ipc_update_perm - update the permissions of an IPC.
+ * @in: the permission given as input.
+ * @out: the permission of the ipc to set.
diff --git a/ipc/util.h b/ipc/util.h
index 3a9e558..271bded 100644
--- a/ipc/util.h
+++ b/ipc/util.h
@@ -126,6 +126,8 @@ struct kern_ipc_perm *ipc_lock(struct ipc_ids *, int);

void kernel_to_ipc64_perm(struct kern_ipc_perm *in, struct ipc64_perm *out);
void ipc64_perm_to_ipc_perm(struct ipc64_perm *in, struct ipc_perm *out);
+int ipc_update_key(struct ipc_ids *ids, struct ipc64_perm *in,

```

```
+ struct kern_ipc_perm *out);
int ipc_update_perm(struct ipc64_perm *in, struct kern_ipc_perm *out);
struct kern_ipc_perm *ipcctl_pre_down(struct ipc_namespace *ns,
    struct ipc_ids *ids, int id, int cmd,
```

Subject: Re: [PATCH v6 03/10] ipc: segment key change helper introduced
Posted by [Serge E. Hallyn](#) on Tue, 23 Oct 2012 16:19:58 GMT
[View Forum Message](#) <> [Reply to Message](#)

Quoting Stanislav Kinsbursky (skinsbursky@parallels.com):

```
> This patch introduces existent segment key changing infrastructure.
> New function ipc_update_key() can be used change segment key, cuid, cgid
> values. It checks for that new key is not used (except IPC_PRIVATE) prior to
> set it on existent.
> To make this possible, added copying of this fields from user-space in
> __get_compat_ipc_perm() and __get_compat_ipc64_perm() functions. Also segment
> search by key and lock were splitted into different functions, because
> ipc_update_key() doesn't need to lock the segment during check that new key is
> not used.
>
> Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
```

Acked-by: Serge E. Hallyn <serge.hallyn@ubuntu.com>

```
> ---
> ipc/compat.c | 6 ++++++
> ipc/util.c | 51 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
> ipc/util.h | 2 ++
> 3 files changed, 56 insertions(+), 3 deletions(-)
>
> diff --git a/ipc/compat.c b/ipc/compat.c
> index ad9518e..af30d13 100644
> --- a/ipc/compat.c
> +++ b/ipc/compat.c
> @@ -144,6 +144,9 @@ static inline int __get_compat_ipc64_perm(struct ipc64_perm *p64,
> err = __get_user(p64->uid, &up64->uid);
> err |= __get_user(p64->gid, &up64->gid);
> err |= __get_user(p64->mode, &up64->mode);
> + err |= __get_user(p64->cuid, &up64->cuid);
> + err |= __get_user(p64->cgid, &up64->cgid);
> + err |= __get_user(p64->key, &up64->key);
> return err;
> }
>
> @@ -155,6 +158,9 @@ static inline int __get_compat_ipc_perm(struct ipc64_perm *p,
> err = __get_user(p->uid, &up->uid);
> err |= __get_user(p->gid, &up->gid);
```

```

> err |= __get_user(p->mode, &up->mode);
> + err |= __get_user(p->cuid, &up->cuid);
> + err |= __get_user(p->cgid, &up->cgid);
> + err |= __get_user(p->key, &up->key);
> return err;
> }
>
> diff --git a/ipc/util.c b/ipc/util.c
> index 503946e..faae296 100644
> --- a/ipc/util.c
> +++ b/ipc/util.c
> @@ -173,7 +173,7 @@ void __init ipc_init_proc_interface(const char *path, const char
*header,
> * @key: The key to find
> *
> * Requires ipc_ids.rw_mutex locked.
> - * Returns the LOCKED pointer to the ipc structure if found or NULL
> + * Returns the UNLOCKED pointer to the ipc structure if found or NULL
> * if not.
> * If key is found ipc points to the owning ipc structure
> */
> @@ -195,7 +195,6 @@ static struct kern_ipc_perm *ipc_findkey(struct ipc_ids *ids, key_t key)
> continue;
> }
>
> - ipc_lock_by_ptr(ipc);
> return ipc;
> }
>
> @@ -203,6 +202,27 @@ static struct kern_ipc_perm *ipc_findkey(struct ipc_ids *ids, key_t key)
> }
>
> /**
> + * ipc_findkey_locked - find and lock a key in an ipc identifier set
> + * @ids: Identifier set
> + * @key: The key to find
> + *
> + * Requires ipc_ids.rw_mutex locked.
> + * Returns the LOCKED pointer to the ipc structure if found or NULL
> + * if not.
> + * If key is found ipc points to the owning ipc structure
> + */
> +
> +static struct kern_ipc_perm *ipc_findkey_locked(struct ipc_ids *ids, key_t key)
> +{
> + struct kern_ipc_perm *ipc;
> +
> + ipc = ipc_findkey(ids, key);

```

```

> + if (ipc)
> + ipc_lock_by_ptr(ipc);
> + return ipc;
> +}
> +
> +/**
>  * ipc_get_maxid - get the last assigned id
>  * @ids: IPC identifier set
>  *
>  * @@ -388,7 +408,7 @@ retry:
>  * a new entry + read locks are not "upgradable"
>  */
> down_write(&ids->rw_mutex);
> - ipcp = ipc_findkey(ids, params->key);
> + ipcp = ipc_findkey_locked(ids, params->key);
> if (ipcp == NULL) {
>  /* key not used */
> if (!(flg & IPC_CREAT))
> @@ -755,6 +775,31 @@ int ipcget(struct ipc_namespace *ns, struct ipc_ids *ids,
> }
>
> /**
> + * ipc_update_key - update the key of an IPC.
> + * @in: the permission given as input.
> + * @out: the permission of the ipc to set.
> + *
> + * Common routine called by sys_shmctl(), sys_semctl(), sys_msgctl().
> + */
> +int ipc_update_key(struct ipc_ids *ids, struct ipc64_perm *in,
> + struct kern_ipc_perm *out)
> +{
> +
> + if (in->key && out->key != in->key) {
> + /*
> +  * Check for existent segment with the same key.
> +  * Note: ipc_ids.rw_mutex is taken for write already.
> +  */
> + if (ipc_findkey(ids, in->key))
> + return -EEXIST;
> + }
> + out->cuid = in->cuid;
> + out->cgid = in->cgid;
> + out->key = in->key;
> + return 0;
> +}
> +
> +/**
>  * ipc_update_perm - update the permissions of an IPC.

```

```
> * @in: the permission given as input.
> * @out: the permission of the ipc to set.
> diff --git a/ipc/util.h b/ipc/util.h
> index 3a9e558..271bded 100644
> --- a/ipc/util.h
> +++ b/ipc/util.h
> @@ -126,6 +126,8 @@ struct kern_ipc_perm *ipc_lock(struct ipc_ids *, int);
>
> void kernel_to_ipc64_perm(struct kern_ipc_perm *in, struct ipc64_perm *out);
> void ipc64_perm_to_ipc_perm(struct ipc64_perm *in, struct ipc_perm *out);
> +int ipc_update_key(struct ipc_ids *ids, struct ipc64_perm *in,
> + struct kern_ipc_perm *out);
> int ipc_update_perm(struct ipc64_perm *in, struct kern_ipc_perm *out);
> struct kern_ipc_perm *ipcctl_pre_down(struct ipc_namespace *ns,
> struct ipc_ids *ids, int id, int cmd,
>
> --
> To unsubscribe from this list: send the line "unsubscribe linux-security-module" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html
```
