
Subject: Re: [PATCH v6 09/10] IPC: message queue copy feature introduced
Posted by [Serge E. Hallyn](#) on Tue, 23 Oct 2012 16:39:48 GMT
[View Forum Message](#) <> [Reply to Message](#)

Quoting Stanislav Kinsbursky (skinsbursky@parallels.com):

> This patch is required for checkpoint/restore in userspace.
> IOW, c/r requires some way to get all pending IPC messages without deleting
> them from the queue (checkpoint can fail and in this case tasks will be resumed,
> so queue have to be valid).
> To achive this, new operation flag MSG_COPY for sys_msgrcv() system call was
> introduced. If this flag was specified, then mtype is interpreted as number of
> the message to copy.
> If MSG_COPY is set, then kernel will allocate dummy message with passed size,
> and then use new copy_msg() helper function to copy desired message (instead of
> unlinking it from the queue).
>
> Notes:
> 1) Return -ENOSYS if MSG_COPY is specified, but CONFIG_CHECKPOINT_RESTORE is
> not set.

How much could it clean things up if a new ipc/cr.c contained

```
#ifdef CONFIG_CHECKPOINT_RESTORE
struct msg_msg *copy_msg(struct msg_msg *src, struct msg_msg *dst)
{
    ...
}
#else
struct msg_msg *copy_msg(struct msg_msg *src, struct msg_msg *dst)
{
    return -ENOSYS;
}
#endif
```

and you went on from there to try to remove all the new #ifdefs from the
existing files?

> Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
> ---
> include/uapi/linux/msg.h | 1 +
> ipc/msg.c | 50 ++++++
> ipc/msgutil.c | 38 ++++++
> ipc/util.h | 1 +
> 4 files changed, 88 insertions(+), 2 deletions(-)
>
> diff --git a/include/uapi/linux/msg.h b/include/uapi/linux/msg.h
> index 76999c9..c1af84a 100644

```

> --- a/include/uapi/linux/msg.h
> +++ b/include/uapi/linux/msg.h
> @@ -11,6 +11,7 @@
> /* msgrcv options */
> #define MSG_NOERROR    010000 /* no error if message is too big */
> #define MSG_EXCEPT    020000 /* recv any msg except of specified type.*/
> +#define MSG_COPY        040000 /* copy (not remove) all queue messages */
>
> /* Obsolete, used only for backwards compatibility and libc5 compiles */
> struct msqid_ds {
> diff --git a/ipc/msg.c b/ipc/msg.c
> index 9808da8..6f52c6b 100644
> --- a/ipc/msg.c
> +++ b/ipc/msg.c
> @@ -788,19 +788,48 @@ long do_msgrcv(int msqid, void __user *buf, size_t bufsz, long
msgtyp,
> struct msg_msg *msg;
> int mode;
> struct ipc_namespace *ns;
> +#ifdef CONFIG_CHECKPOINT_RESTORE
> + struct msg_msg *copy = NULL;
> + unsigned long copy_number = 0;
> +#endif
>
> if (msqid < 0 || (long) bufsz < 0)
> return -EINVAL;
> + if (msgflg & MSG_COPY) {
> +#ifdef CONFIG_CHECKPOINT_RESTORE
> +
> + if (msgflg & MSG_COPY) {
> + copy_number = msgtyp;
> + msgtyp = 0;
> + }
> +
> + /*
> + * Create dummy message to copy real message to.
> + */
> + copy = load_msg(buf, bufsz);
> + if (IS_ERR(copy))
> + return PTR_ERR(copy);
> + copy->m_ts = bufsz;
> +#else
> + return -ENOSYS;
> +#endif
> + }
> mode = convert_mode(&msgtyp, msgflg);
> ns = current->nsproxy->ipc_ns;
>

```

```

> msq = msg_lock_check(ns, msqid);
> - if (IS_ERR(msq))
> + if (IS_ERR(msq)) {
> + #ifdef CONFIG_CHECKPOINT_RESTORE
> + if (msgflg & MSG_COPY)
> + free_msg(copy);
> + #endif
> return PTR_ERR(msq);
> + }
>
> for (;;) {
> struct msg_receiver msr_d;
> struct list_head *tmp;
> + long msg_counter = 0;
>
> msg = ERR_PTR(-EACCES);
> if (ipcperms(ns, &msq->q_perm, S_IRUGO))
> @@ -820,8 +849,16 @@ long do_msgrcv(int msqid, void __user *buf, size_t bufsz, long
msgtyp,
> if (mode == SEARCH_LESSEQUAL &&
> walk_msg->m_type != 1) {
> msgtyp = walk_msg->m_type - 1;
> + #ifdef CONFIG_CHECKPOINT_RESTORE
> + } else if (msgflg & MSG_COPY) {
> + if (copy_number == msg_counter) {
> + msg = copy_msg(walk_msg, copy);
> + break;
> + }
> + #endif
> } else
> break;
> + msg_counter++;
> }
> tmp = tmp->next;
> }
> @@ -834,6 +871,10 @@ long do_msgrcv(int msqid, void __user *buf, size_t bufsz, long
msgtyp,
> msg = ERR_PTR(-E2BIG);
> goto out_unlock;
> }
> + #ifdef CONFIG_CHECKPOINT_RESTORE
> + if (msgflg & MSG_COPY)
> + goto out_unlock;
> + #endif
> list_del(&msg->m_list);
> msq->q_qnum--;
> msq->q_rtime = get_seconds();
> @@ -917,8 +958,13 @@ out_unlock:

```

```

> break;
> }
> }
> - if (IS_ERR(msg))
> + if (IS_ERR(msg)) {
> + #ifdef CONFIG_CHECKPOINT_RESTORE
> + if (msgflg & MSG_COPY)
> + free_msg(copy);
> + #endif
> return PTR_ERR(msg);
> + }
>
> bufisz = msg_handler(buf, msg, bufisz);
> free_msg(msg);
> diff --git a/ipc/msgutil.c b/ipc/msgutil.c
> index 26143d3..b281f5c 100644
> --- a/ipc/msgutil.c
> +++ b/ipc/msgutil.c
> @@ -100,7 +100,45 @@ out_err:
> free_msg(msg);
> return ERR_PTR(err);
> }
> + #ifdef CONFIG_CHECKPOINT_RESTORE
> + struct msg_msg *copy_msg(struct msg_msg *src, struct msg_msg *dst)
> + {
> + struct msg_msgseg *dst_pseg, *src_pseg;
> + int len = src->m_ts;
> + int alen;
> +
> + BUG_ON(dst == NULL);
> + if (src->m_ts > dst->m_ts)
> + return ERR_PTR(-EINVAL);
> +
> + alen = len;
> + if (alen > DATALEN_MSG)
> + alen = DATALEN_MSG;
> +
> + dst->next = NULL;
> + dst->security = NULL;
>
> + memcpy(dst + 1, src + 1, alen);
> +
> + len -= alen;
> + dst_pseg = dst->next;
> + src_pseg = src->next;
> + while (len > 0) {
> + alen = len;
> + if (alen > DATALEN_SEG)

```

```

> + alen = DATALEN_SEG;
> + memcpy(dst_pseg + 1, src_pseg + 1, alen);
> + dst_pseg = dst_pseg->next;
> + len -= alen;
> + src_pseg = src_pseg->next;
> + }
> +
> + dst->m_type = src->m_type;
> + dst->m_ts = src->m_ts;
> +
> + return dst;
> +}
> +#endif
> int store_msg(void __user *dest, struct msg_msg *msg, int len)
> {
>     int alen;
> diff --git a/ipc/util.h b/ipc/util.h
> index 271bded..027f507 100644
> --- a/ipc/util.h
> +++ b/ipc/util.h
> @@ -142,6 +142,7 @@ int ipc_parse_version (int *cmd);
>
> extern void free_msg(struct msg_msg *msg);
> extern struct msg_msg *load_msg(const void __user *src, int len);
> +extern struct msg_msg *copy_msg(struct msg_msg *src, struct msg_msg *dst);
> extern int store_msg(void __user *dest, struct msg_msg *msg, int len);
>
> extern void recompute_msgmni(struct ipc_namespace *);
>
> --
> 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

```
