author	Jann Horn <jannh@google.com>	2020-12-03 02:25:05 +0100
committer	Greg Kroah-Hartman <gregkh@linuxfoundation.org>	2020-12-04 17:39:58 +0100
commit	c8bcd9c5be24fb9e6132e97da5a35e55a83e36b9 (patch)	
tree	1b7e3191d3fd63c02d3029b19e45d88d322397df	
parent	54ffccbf053b5b6ca4f6e45094b942fab92a25fc (diff)	
download	linux-c8bcd9c5be24fb9e6132e97da5a35e55a83e36b9.tar.gz	

diff options

context:	3
space:	include
mode:	unified

tty: Fix ->session locking

```
Currently, locking of ->session is very inconsistent; most places
protect it using the legacy tty mutex, but disassociate_ctty(),
__do_SAK(), tiocspgrp() and tiocgsid() don't.
Two of the writers hold the ctrl_lock (because they already need it for
->pgrp), but __proc_set_tty() doesn't do that yet.

On a PREEMPT=y system, an unprivileged user can theoretically abuse
this broken locking to read 4 bytes of freed memory via TIOCGSID if
tiocgsid() is preempted long enough at the right point. (Other things
might also go wrong, especially if root-only ioctls are involved; I'm
not sure about that.)

Change the locking on ->session such that:

 - tty_lock() is held by all writers: By making disassociate_ctty()
   hold it. This should be fine because the same lock can already be
   taken through the call to tty_vhangup_session().
   The tricky part is that we need to shorten the area covered by
   siglock to be able to take tty_lock() without ugly retry logic; as
   far as I can tell, this should be fine, since nothing in the
   signal_struct is touched in the `if (tty)` branch.
 - ctrl_lock is held by all writers: By changing __proc_set_tty() to
   hold the lock a little longer.
 - All readers that aren't holding tty_lock() hold ctrl_lock: By
   adding locking to tiocgsid() and __do_SAK(), and expanding the area
   covered by ctrl_lock in tiocspgrp().

Cc: stable@kernel.org
Signed-off-by: Jann Horn <jannh@google.com>
Reviewed-by: Jiri Slaby <jirislaby@kernel.org>
Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
```

Diffstat

-rw-r--r--	drivers/tty/tty_io.c	7
-rw-r--r--	drivers/tty/tty_jobctrl.c	44
-rw-r--r--	include/linux/tty.h	4

3 files changed, 41 insertions, 14 deletions

```
diff --git a/drivers/tty/tty_io.c b/drivers/tty/tty_io.c
index 9f8b9a567b359..56ade99ef99f4 100644
--- a/drivers/tty/tty_io.c
+++ b/drivers/tty/tty_io.c
@@ -2897,10 +2897,14 @@ void __do_SAK(struct tty_struct *tty)
 	struct task_struct *g, *p;
 	struct pid *session;
 	int		i;
+	unsigned long flags;
 
 	if (!tty)
 		return;
-	session = tty->session;
+
+	spin_lock_irqsave(&tty->ctrl_lock, flags);
+	session = get_pid(tty->session);
+	spin_unlock_irqrestore(&tty->ctrl_lock, flags);
 
 	tty_ldisc_flush(tty);
 
@@ -2932,6 +2936,7 @@ void __do_SAK(struct tty_struct *tty)
 		task_unlock(p);
 	} while_each_thread(g, p);
 	read_unlock(&tasklist_lock);
+	put_pid(session);
 #endif
 }
 

diff --git a/drivers/tty/tty_jobctrl.c b/drivers/tty/tty_jobctrl.c
index baadeea4a289b..aa6d0537b379e 100644
--- a/drivers/tty/tty_jobctrl.c
+++ b/drivers/tty/tty_jobctrl.c
@@ -103,8 +103,8 @@ static void __proc_set_tty(struct tty_struct *tty)
 	put_pid(tty->session);
 	put_pid(tty->pgrp);
 	tty->pgrp = get_pid(task_pgrp(current));
-	spin_unlock_irqrestore(&tty->ctrl_lock, flags);
 	tty->session = get_pid(task_session(current));
+	spin_unlock_irqrestore(&tty->ctrl_lock, flags);
 	if (current->signal->tty) {
 		tty_debug(tty, "current tty %s not NULL!!\n",
 			  current->signal->tty->name);
@@ -293,20 +293,23 @@ void disassociate_ctty(int on_exit)
 	spin_lock_irq(&current->sighand->siglock);
 	put_pid(current->signal->tty_old_pgrp);
 	current->signal->tty_old_pgrp = NULL;
-
 	tty = tty_kref_get(current->signal->tty);
+	spin_unlock_irq(&current->sighand->siglock);
+
 	if (tty) {
 		unsigned long flags;
+
+		tty_lock(tty);
 		spin_lock_irqsave(&tty->ctrl_lock, flags);
 		put_pid(tty->session);
 		put_pid(tty->pgrp);
 		tty->session = NULL;
 		tty->pgrp = NULL;
 		spin_unlock_irqrestore(&tty->ctrl_lock, flags);
+		tty_unlock(tty);
 		tty_kref_put(tty);
```

```
        }

-       spin_unlock_irq(&current->sighand->siglock);
        /* Now clear signal->tty under the lock */
        read_lock(&tasklist_lock);
        session_clear_tty(task_session(current));
@@ -477,14 +480,19 @@ static int tiocspgrp(struct tty_struct *tty, struct tty_struct *real_tty, pid_t
                return -ENOTTY;
        if (retval)
                return retval;
-       if (!current->signal->tty ||
-           (current->signal->tty != real_tty) ||
-           (real_tty->session != task_session(current)))
-               return -ENOTTY;
+
        if (get_user(pgrp_nr, p))
                return -EFAULT;
        if (pgrp_nr < 0)
                return -EINVAL;
+
+       spin_lock_irq(&real_tty->ctrl_lock);
+       if (!current->signal->tty ||
+           (current->signal->tty != real_tty) ||
+           (real_tty->session != task_session(current))) {
+               retval = -ENOTTY;
+               goto out_unlock_ctrl;
+       }
        rcu_read_lock();
        pgrp = find_vpid(pgrp_nr);
        retval = -ESRCH;
@@ -494,12 +502,12 @@ static int tiocspgrp(struct tty_struct *tty, struct tty_struct *real_tty, pid_t
        if (session_of_pgrp(pgrp) != task_session(current))
                goto out_unlock;
        retval = 0;
-       spin_lock_irq(&real_tty->ctrl_lock);
        put_pid(real_tty->pgrp);
        real_tty->pgrp = get_pid(pgrp);
-       spin_unlock_irq(&real_tty->ctrl_lock);
 out_unlock:
        rcu_read_unlock();
+out_unlock_ctrl:
+       spin_unlock_irq(&real_tty->ctrl_lock);
        return retval;
 }

@@ -511,20 +519,30 @@ out_unlock:
  *
  *     Obtain the session id of the tty. If there is no session
  *     return an error.
- *
- *     Locking: none. Reference to current->signal->tty is safe.
  */
 static int tiocgsid(struct tty_struct *tty, struct tty_struct *real_tty, pid_t __user *p)
 {
+       unsigned long flags;
+       pid_t sid;
+
        /*
         * (tty == real_tty) is a cheap way of
         * testing if the tty is NOT a master pty.
        */
        if (tty == real_tty && current->signal->tty != real_tty)
                return -ENOTTY;
+
+       spin_lock_irqsave(&real_tty->ctrl_lock, flags);
        if (!real_tty->session)
-               return -ENOTTY;
-       return put_user(pid_vnr(real_tty->session), p);
+               goto err;
+       sid = pid_vnr(real_tty->session);
+       spin_unlock_irqrestore(&real_tty->ctrl_lock, flags);
+
+       return put_user(sid, p);
+
+err:
+       spin_unlock_irqrestore(&real_tty->ctrl_lock, flags);
+       return -ENOTTY;
 }

 /*

diff --git a/include/linux/tty.h b/include/linux/tty.h
index a99e9b8e4e316..eb33d948788cc 100644
--- a/include/linux/tty.h
+++ b/include/linux/tty.h
@@ -306,6 +306,10 @@ struct tty_struct {
        struct termiox *termiox;        /* May be NULL for unsupported */
        char name[64];
        struct pid *pgrp;               /* Protected by ctrl lock */
+       /*
+        * Writes protected by both ctrl lock and legacy mutex, readers must use
+        * at least one of them.
+        */
        struct pid *session;
        unsigned long flags;
        int count;
```