Subject: [PATCH] vt: Make SAK run in process context.
Posted by ebiederm on Mon, 11 Dec 2006 13:07:03 GMT
View Forum Message <> Reply to Message

This defers SAK so we can use the normal console semaphore to order
operations.

This removes the xchg operations that I used to attempt to attmically
update struct pid, because of the strange locking used for SAK. With
SAK using the normal console semaphore nothing special is needed.

Cc: Oleg Nesterov <oleg@tv-sign.ru>
Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
drivers/char/keyboard.c | 11 +++--------
drivers/char/sysrq.c | 7 ++++---
drivers/char/vt_ioctl.c | 27 +++++++++++++++++++++++++--
include/linux/vt_kern.h | 1 +
4 files changed, 33 insertions(+), 13 deletions(-)

```
diff --git a/drivers/char/keyboard.c b/drivers/char/keyboard.c
index 7a6c1c0..bd6912d 100644
--- a/drivers/char/keyboard.c
+++ b/drivers/char/keyboard.c
@@ -595,15 +595,10 @@ static void fn_spawn_con(struct vc_data *vc)

 static void fn_SAK(struct vc_data *vc)
 {
- struct tty_struct *tty = vc->vc_tty;
+ static DECLARE_WORK(SAK_work, deferred_SAK, NULL);
+ SAK_work.data = &vc_cons[fg_console];

- /*
-  * SAK should also work in all raw modes and reset
-  * them properly.
-  */
- if (tty)
-  do_SAK(tty);
- reset_vc(vc);
+ schedule_work(&SAK_work);
 }

 static void fn_null(struct vc_data *vc)
diff --git a/drivers/char/sysrq.c b/drivers/char/sysrq.c
index 05810c8..b4f67a0 100644
--- a/drivers/char/sysrq.c
+++ b/drivers/char/sysrq.c
@@ -61,9 +61,10 @@ static struct sysrq_key_op sysrq_loglevel_op = {
```

```
 #ifdef CONFIG_VT
 static void sysrq_handle_SAK(int key, struct tty_struct *tty)
 {
- if (tty)
-  do_SAK(tty);
- reset_vc(vc_cons[fg_console].d);
+ static DECLARE_WORK(SAK_work, deferred_SAK, NULL);
+ SAK_work.data = &vc_cons[fg_console];
+
+ schedule_work(&SAK_work);
 }
 static struct sysrq_key_op sysrq_SAK_op = {
  .handler = sysrq_handle_SAK,
diff --git a/drivers/char/vt_ioctl.c b/drivers/char/vt_ioctl.c
index 311493e..5ecc71e 100644
--- a/drivers/char/vt_ioctl.c
+++ b/drivers/char/vt_ioctl.c
@@ -672,7 +672,8 @@ int vt_ioctl(struct tty_struct *tty, struct file * file,
   vc->vt_mode = tmp;
   /* the frsig is ignored, so we set it to 0 */
   vc->vt_mode.frsig = 0;
-  put_pid(xchg(&vc->vt_pid, get_pid(task_pid(current))));
+  put_pid(vc->vt_pid);
+  vc->vt_pid = get_pid(task_pid(current));
   /* no switch is required -- saw@shade.msu.ru */
   vc->vt_newvt = -1;
   release_console_sem();
@@ -1063,12 +1064,34 @@ void reset_vc(struct vc_data *vc)
  vc->vt_mode.relsig = 0;
  vc->vt_mode.acqsig = 0;
  vc->vt_mode.frsig = 0;
- put_pid(xchg(&vc->vt_pid, NULL));
+ put_pid(vc->vt_pid);
+ vc->vt_pid = NULL;
  vc->vt_newvt = -1;
  if (!in_interrupt())    /* Via keyboard.c:SAK() - akpm */
   reset_palette(vc);
 }

+void deferred_SAK(void *data)
+{
+ struct vc *vc_con = data;
+ struct vc_data *vc;
+ struct tty_struct *tty;
+
+ acquire_console_sem();
+ vc = vc_con->d;
+ if (vc) {
```

```
+  tty = vc->vc_tty;
+  /*
+   * SAK should also work in all raw modes and reset
+   * them properly.
+   */
+  if (tty)
+   do_SAK(tty);
+  reset_vc(vc);
+ }
+ release_console_sem();
+}
+
 /*
  * Performs the back end of a vt switch
  */
diff --git a/include/linux/vt_kern.h b/include/linux/vt_kern.h
index 37a1a41..a727896 100644
--- a/include/linux/vt_kern.h
+++ b/include/linux/vt_kern.h
@@ -74,6 +74,7 @@ int con_copy_unimap(struct vc_data *dst_vc, struct vc_data *src_vc);
 int vt_waitactive(int vt);
 void change_console(struct vc_data *new_vc);
 void reset_vc(struct vc_data *vc);
+void deferred_SAK(void *data);

 /*
  * vc_screen.c shares this temporary buffer with the console write code so that
--
1.4.4.1.g278f
```

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [PATCH] vt: Make SAK run in process context.
Posted by Andrew Morton on Mon, 11 Dec 2006 20:56:40 GMT
View Forum Message <> Reply to Message

On Mon, 11 Dec 2006 06:07:03 -0700
ebiederm@xmission.com (Eric W. Biederman) wrote:

>
> This defers SAK so we can use the normal console semaphore to order
> operations.
>
> This removes the xchg operations that I used to attempt to attmically

> update struct pid, because of the strange locking used for SAK. With
> SAK using the normal console semaphore nothing special is needed.
>

This is all a bit smelly.

>
> diff --git a/drivers/char/keyboard.c b/drivers/char/keyboard.c
> index 7a6c1c0..bd6912d 100644
> --- a/drivers/char/keyboard.c
> +++ b/drivers/char/keyboard.c
> @@ -595,15 +595,10 @@ static void fn_spawn_con(struct vc_data *vc)
>
> static void fn_SAK(struct vc_data *vc)
> {
> - struct tty_struct *tty = vc->vc_tty;
> + static DECLARE_WORK(SAK_work, deferred_SAK, NULL);
> + SAK_work.data = &vc_cons[fg_console];

static storage...

> static void sysrq_handle_SAK(int key, struct tty_struct *tty)
> {
> - if (tty)
> - do_SAK(tty);
> - reset_vc(vc_cons[fg_console].d);
> + static DECLARE_WORK(SAK_work, deferred_SAK, NULL);

More.

>
> +void deferred_SAK(void *data)
> +{
> + struct vc *vc_con = data;
> + struct vc_data *vc;
> + struct tty_struct *tty;
> +
> + acquire_console_sem();
> + vc = vc_con->d;
> + if (vc) {
> + tty = vc->vc_tty;
> + /*
> +   * SAK should also work in all raw modes and reset
> +   * them properly.
> +   */
> + if (tty)
> +   do_SAK(tty);
> + reset_vc(vc);

> + }
> + release_console_sem();
> +}

And a workqueue callback which calls a function which immediately does
another schedule_work().

I suspect you can fix all of this by passing a function pointer into
do_SAK(): to either __do_SAK or to some new function which does the vc
lookup then calls __do_SAK().

It probably means that you'll need to pass some payload into the workqueue
callback, and dhowells just went and broke that on us. That can be fixed
by adding a new `void *tty_struct.SAK_work_data'.

hmm, do_SAK() is being a bit bad, overwriting the ->SAK_work on a
work_struct which might presently be scheduled. To do this safely we need
a new variant on queue_work():

```
int queue_work_with_data(struct workqueue_struct *wq,
   struct work_struct *work, void **datap, void *data
{
 int ret = 0, cpu = get_cpu();

 if (!test_and_set_bit(WORK_STRUCT_PENDING, &work->management)) {
  if (datap)
   *datap = data;
  if (unlikely(is_single_threaded(wq)))
   cpu = singlethread_cpu;
  BUG_ON(!list_empty(&work->entry));
  __queue_work(per_cpu_ptr(wq->cpu_wq, cpu), work);
  ret = 1;
 }
 put_cpu();
 return ret;
}
```

then, of course,

```
int queue_work(struct workqueue_struct *wq, struct work_struct *work)
{
 return queue_work_with_data(wq, work, NULL, NULL);
}
```

iirc, other places in the kernel need queue_work_with_data(), for removal
of the *_WORK_NAR() stuff.

Subject: Re: [PATCH] vt: Make SAK run in process context.
Posted by ebiederm on Mon, 11 Dec 2006 21:27:40 GMT
View Forum Message <> Reply to Message

Andrew Morton <akpm@osdl.org> writes:

> On Mon, 11 Dec 2006 06:07:03 -0700
> ebiederm@xmission.com (Eric W. Biederman) wrote:
>
>>
>> This defers SAK so we can use the normal console semaphore to order
>> operations.
>>
>> This removes the xchg operations that I used to attempt to attmically
>> update struct pid, because of the strange locking used for SAK. With
>> SAK using the normal console semaphore nothing special is needed.
>>
>
> This is all a bit smelly.

Ok. I will take a second look, thanks for catching this.

I think I was half blind when I prepared this patch, I missed
that do_SAK was scheduling work itself.

>>
>> +void deferred_SAK(void *data)
>> +{
>> + struct vc *vc_con = data;
>> + struct vc_data *vc;
>> + struct tty_struct *tty;
>> +
>> + acquire_console_sem();
>> + vc = vc_con->d;
>> + if (vc) {
>> + tty = vc->vc_tty;
>> + /*
>> +   * SAK should also work in all raw modes and reset
>> +   * them properly.
>> +   */
>> + if (tty)
>> +   do_SAK(tty);
>> + reset_vc(vc);

>> + }
>> + release_console_sem();
>> +}
>
> And a workqueue callback which calls a function which immediately does
> another schedule_work().
>
> I suspect you can fix all of this by passing a function pointer into
> do_SAK(): to either __do_SAK or to some new function which does the vc
> lookup then calls __do_SAK().

Yes. It looks like all I need is an appropriate factor of __do_SAK() that
I can call immediately.

> It probably means that you'll need to pass some payload into the workqueue
> callback, and dhowells just went and broke that on us. That can be fixed
> by adding a new `void *tty_struct.SAK_work_data'.
>
>
> hmm, do_SAK() is being a bit bad, overwriting the ->SAK_work on a
> work_struct which might presently be scheduled. To do this safely we need
> a new variant on queue_work():

And of course there is the truly silly issue that X spells uses
Ctrl-Alt-Backspace instead of the kernel provided SAK to implement this.

Regardless that looks right. Unless there is some locking on the tty we
can exploit.

> int queue_work_with_data(struct workqueue_struct *wq,
>   struct work_struct *work, void **datap, void *data
> {
> int ret = 0, cpu = get_cpu();
>
> if (!test_and_set_bit(WORK_STRUCT_PENDING, &work->management)) {
>   if (datap)
>   *datap = data;
>   if (unlikely(is_single_threaded(wq)))
>   cpu = singlethread_cpu;
>   BUG_ON(!list_empty(&work->entry));
>   __queue_work(per_cpu_ptr(wq->cpu_wq, cpu), work);
>   ret = 1;
> }
> put_cpu();
> return ret;
> }
>
> then, of course,

>
> int queue_work(struct workqueue_struct *wq, struct work_struct *work)
> {
> return queue_work_with_data(wq, work, NULL, NULL);
> }
>
> iirc, other places in the kernel need queue_work_with_data(), for removal
> of the *_WORK_NAR() stuff.

Wow. The intersection of the clean ups.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: [PATCH] vt: Refactor console SAK processing.
Posted by ebiederm on Tue, 12 Dec 2006 09:56:51 GMT
View Forum Message <> Reply to Message

This does several things.
- It moves looking up of the current foreground console into process
  context where we can safely take the semaphore that protects this
  operation.
- It uses the new flavor of work queue processing.
- This generates a factor of do_SAK, __do_SAK that runs immediately.
- This calls __do_SAK with the console semaphore held ensuring nothing
  else happens to the console while we process the SAK operation.
- With the console SAK processing moved into process context this
  patch removes the xchg operations that I used to attempt to attomically
  update struct pid, because of the strange locking used in the SAK processing.
  With SAK using the normal console semaphore nothing special is needed.

Cc: Oleg Nesterov <oleg@tv-sign.ru>
Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 drivers/char/keyboard.c | 12 +++---------
 drivers/char/sysrq.c | 6 +++---
 drivers/char/tty_io.c | 13 +++++++++----
 drivers/char/vt_ioctl.c | 28 ++++++++++++++++++++++++++--
 include/linux/console_struct.h | 3 +++
 include/linux/tty.h | 1 +
 6 files changed, 45 insertions(+), 18 deletions(-)

```
diff --git a/drivers/char/keyboard.c b/drivers/char/keyboard.c
index 7a6c1c0..c654a3e 100644
--- a/drivers/char/keyboard.c
+++ b/drivers/char/keyboard.c
@@ -595,15 +595,9 @@ static void fn_spawn_con(struct vc_data *vc)

 static void fn_SAK(struct vc_data *vc)
 {
- struct tty_struct *tty = vc->vc_tty;
-
- /*
-  * SAK should also work in all raw modes and reset
-  * them properly.
-  */
- if (tty)
-  do_SAK(tty);
- reset_vc(vc);
+ struct work_struct *SAK_work = &vc_cons[fg_console].SAK_work;
+ PREPARE_WORK(SAK_work, vc_SAK);
+ schedule_work(SAK_work);
 }

 static void fn_null(struct vc_data *vc)
diff --git a/drivers/char/sysrq.c b/drivers/char/sysrq.c
index 05810c8..6cf877f 100644
--- a/drivers/char/sysrq.c
+++ b/drivers/char/sysrq.c
@@ -61,9 +61,9 @@ static struct sysrq_key_op sysrq_loglevel_op = {
 #ifdef CONFIG_VT
 static void sysrq_handle_SAK(int key, struct tty_struct *tty)
 {
- if (tty)
-  do_SAK(tty);
- reset_vc(vc_cons[fg_console].d);
+ struct work_struct *SAK_work = &vc_cons[fg_console].SAK_work;
+ PREPARE_WORK(SAK_work, vc_SAK);
+ schedule_work(SAK_work);
 }
 static struct sysrq_key_op sysrq_SAK_op = {
   .handler = sysrq_handle_SAK,
diff --git a/drivers/char/tty_io.c b/drivers/char/tty_io.c
index 4044c86..a333881 100644
--- a/drivers/char/tty_io.c
+++ b/drivers/char/tty_io.c
@@ -3324,10 +3324,8 @@ int tty_ioctl(struct inode * inode, struct file * file,
   * Nasty bug: do_SAK is being called in interrupt context.  This can
   * deadlock.  We punt it up to process context.  AKPM - 16Mar2001
   */
```

```
-static void __do_SAK(struct work_struct *work)
+void __do_SAK(struct tty_struct *tty)
 {
- struct tty_struct *tty =
-  container_of(work, struct tty_struct, SAK_work);
 #ifdef TTY_SOFT_SAK
  tty_hangup(tty);
 #else
@@ -3399,6 +3397,13 @@ static void __do_SAK(struct work_struct *work)
 #endif
 }

+static void do_SAK_work(struct work_struct *work)
+{
+ struct tty_struct *tty =
+  container_of(work, struct tty_struct, SAK_work);
+ __do_SAK(tty);
+}
+
 /*
  * The tq handling here is a little racy - tty->SAK_work may already be queued.
  * Fortunately we don't need to worry, because if ->SAK_work is already queued,
@@ -3409,7 +3414,7 @@ void do_SAK(struct tty_struct *tty)
 {
  if (!tty)
   return;
- PREPARE_WORK(&tty->SAK_work, __do_SAK);
+ PREPARE_WORK(&tty->SAK_work, do_SAK_work);
  schedule_work(&tty->SAK_work);
 }

diff --git a/drivers/char/vt_ioctl.c b/drivers/char/vt_ioctl.c
index 311493e..8384846 100644
--- a/drivers/char/vt_ioctl.c
+++ b/drivers/char/vt_ioctl.c
@@ -672,7 +672,8 @@ int vt_ioctl(struct tty_struct *tty, struct file * file,
   vc->vt_mode = tmp;
   /* the frsig is ignored, so we set it to 0 */
   vc->vt_mode.frsig = 0;
-  put_pid(xchg(&vc->vt_pid, get_pid(task_pid(current))));
+  put_pid(vc->vt_pid);
+  vc->vt_pid = get_pid(task_pid(current));
   /* no switch is required -- saw@shade.msu.ru */
   vc->vt_newvt = -1;
   release_console_sem();
@@ -1063,12 +1064,35 @@ void reset_vc(struct vc_data *vc)
  vc->vt_mode.relsig = 0;
 vc->vt_mode.acqsig = 0;
```

```
  vc->vt_mode.frsig = 0;
- put_pid(xchg(&vc->vt_pid, NULL));
+ put_pid(vc->vt_pid);
+ vc->vt_pid = NULL;
  vc->vt_newvt = -1;
  if (!in_interrupt())    /* Via keyboard.c:SAK() - akpm */
   reset_palette(vc);
 }

+void vc_SAK(struct work_struct *work)
+{
+ struct vc *vc_con =
+  container_of(work, struct vc, SAK_work);
+ struct vc_data *vc;
+ struct tty_struct *tty;
+
+ acquire_console_sem();
+ vc = vc_con->d;
+ if (vc) {
+  tty = vc->vc_tty;
+  /*
+   * SAK should also work in all raw modes and reset
+   * them properly.
+   */
+  if (tty)
+   __do_SAK(tty);
+  reset_vc(vc);
+ }
+ release_console_sem();
+}
+
 /*
  * Performs the back end of a vt switch
  */
diff --git a/include/linux/console_struct.h b/include/linux/console_struct.h
index ed6c0fe..a86162b 100644
--- a/include/linux/console_struct.h
+++ b/include/linux/console_struct.h
@@ -11,6 +11,7 @@

 #include <linux/wait.h>
 #include <linux/vt.h>
+#include <linux/workqueue.h>

 struct vt_struct;

@@ -103,6 +104,7 @@ struct vc_data {
```

```
 struct vc {
  struct vc_data *d;
+ struct work_struct SAK_work;

  /* might add  scrmem, vt_struct, kbd  at some time,
     to have everything in one place - the disadvantage
@@ -110,6 +112,7 @@ struct vc {
 };

 extern struct vc vc_cons [MAX_NR_CONSOLES];
+extern void vc_SAK(struct work_struct *work);

 #define CUR_DEF  0
 #define CUR_NONE 1
diff --git a/include/linux/tty.h b/include/linux/tty.h
index 65cbcf2..0161a8c 100644
--- a/include/linux/tty.h
+++ b/include/linux/tty.h
@@ -291,6 +291,7 @@ extern void tty_vhangup(struct tty_struct * tty);
 extern void tty_unhangup(struct file *filp);
 extern int tty_hung_up_p(struct file * filp);
 extern void do_SAK(struct tty_struct *tty);
+extern void __do_SAK(struct tty_struct *tty);
 extern void disassociate_ctty(int priv);
 extern void tty_flip_buffer_push(struct tty_struct *tty);
 extern speed_t tty_get_baud_rate(struct tty_struct *tty);
--
1.4.4.1.g278f
```

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers