Subject: [PATCH 1/4] proc: fix NULL ->i_fop oops
Posted by Alexey Dobriyan on Fri, 16 Nov 2007 15:06:51 GMT
View Forum Message <> Reply to Message

proc_kill_inodes() can clear ->i_fop in the middle of vfs_readdir resulting in NULL dereference during "file->f_op->readdir(file, buf, filler)".

The solution is to remove proc_kill_inodes() completely:
a) we don't have tricky modules implementing their tricky readdir hooks which
   could keeping this revoke from hell.
b) In a situation when module is gone but PDE still alive, standard readdir
   will return only "." and "..", because pde->next was cleared by
   remove_proc_entry().
c) the race proc_kill_inode() destined to prevent is not completely fixed, just
   race window made smaller, because vfs_readdir() is run without sb_lock held and
   without file_list_lock held. Effectively, ->i_fop is cleared at random moment,
   which can't fix properly anything.

```
BUG: unable to handle kernel NULL pointer dereference at virtual address 00000018
printing eip: c1061205 *pdpt = 0000000005b22001 *pde = 0000000000000000
Oops: 0000 [#1] PREEMPT SMP
Modules linked in: foo af_packet ipv6 cpufreq_ondemand loop serio_raw sr_mod k8temp cdrom hwmon amd_rng
Pid: 2033, comm: find Not tainted (2.6.24-rc1-b1d08ac064268d0ae2281e98bf5e82627e0f0c56 #2)
EIP: 0060:[<c1061205>] EFLAGS: 00010246 CPU: 0
EIP is at vfs_readdir+0x47/0x74
EAX: c6b6a780 EBX: 00000000 ECX: c1061040 EDX: c5decf94
ESI: c6b6a780 EDI: fffffffe EBP: c9797c54 ESP: c5decf78
 DS: 007b ES: 007b FS: 00d8 GS: 0033 SS: 0068
Process find (pid: 2033, ti=c5dec000 task=c64bba90 task.ti=c5dec000)
Stack: c5decf94 c1061040 fffffff7 0805ffbc 00000000 c6b6a780 c1061295 0805ffbc
       00000000 00000400 00000000 00000004 0805ffbc 4588eff4 c5dec000 c10026ba
       00000004 0805ffbc 00000400 0805ffbc 4588eff4 bfdc6c70 000000dc 0000007b
Call Trace:
 [<c1061040>] filldir64+0x0/0xc5
 [<c1061295>] sys_getdents64+0x63/0xa5
 [<c10026ba>] sysenter_past_esp+0x5f/0x85
 =======================
Code: 49 83 78 18 00 74 43 8d 6b 74 bf fe ff ff ff 89 e8 e8 b8 c0 12 00 f6 83 2c 01 00 00 10 75 22 8b 5e 10 8b 4c 24 04 89 f0 8b 14 24 <ff> 53 18 f6 46 1a 04 89 c7 75 0b 8b 56 0c 8b 46 08 e8 c8 66 00
EIP: [<c1061205>] vfs_readdir+0x47/0x74 SS:ESP 0068:c5decf78
```

Signed-off-by: Alexey Dobriyan <adobriyan@sw.ru>

```
 fs/proc/generic.c |   37 -------------------------------------
 fs/proc/internal.h |    2 --
 fs/proc/root.c     |    2 +-
 3 files changed, 1 insertion(+), 40 deletions(-)

--- a/fs/proc/generic.c
+++ b/fs/proc/generic.c
@@ -544,41 +544,6 @@ static int proc_register(struct proc_dir_entry * dir, struct proc_dir_entry *
dp
  return 0;
 }

-/*
- * Kill an inode that got unregistered..
- */
-static void proc_kill_inodes(struct proc_dir_entry *de)
-{
- struct list_head *p;
- struct super_block *sb;
-
- /*
-  * Actually it's a partial revoke().
-  */
- spin_lock(&sb_lock);
- list_for_each_entry(sb, &proc_fs_type.fs_supers, s_instances) {
-  file_list_lock();
-  list_for_each(p, &sb->s_files) {
-   struct file *filp = list_entry(p, struct file,
-       f_u.fu_list);
-   struct dentry *dentry = filp->f_path.dentry;
-   struct inode *inode;
-   const struct file_operations *fops;
-
-   if (dentry->d_op != &proc_dentry_operations)
-    continue;
-   inode = dentry->d_inode;
-   if (PDE(inode) != de)
-    continue;
-   fops = filp->f_op;
-   filp->f_op = NULL;
-   fops_put(fops);
-  }
-  file_list_unlock();
- }
- spin_unlock(&sb_lock);
-}
-
 static struct proc_dir_entry *proc_create(struct proc_dir_entry **parent,
       const char *name,
       mode_t mode,
@@ -753,8 +718,6 @@ void remove_proc_entry(const char *name, struct proc_dir_entry *parent)
 continue_removing:
   if (S_ISDIR(de->mode))
    parent->nlink--;
-  if (!S_ISREG(de->mode))
-   proc_kill_inodes(de);
   de->nlink = 0;
   WARN_ON(de->subdir);
   if (!atomic_read(&de->count))
--- a/fs/proc/internal.h
+++ b/fs/proc/internal.h
@@ -78,5 +78,3 @@ static inline int proc_fd(struct inode *inode)
 {
  return PROC_I(inode)->fd;
 }
-
-extern struct file_system_type proc_fs_type;
--- a/fs/proc/root.c
+++ b/fs/proc/root.c
@@ -98,7 +98,7 @@ static void proc_kill_sb(struct super_block *sb)
  put_pid_ns(ns);
 }

-struct file_system_type proc_fs_type = {
+static struct file_system_type proc_fs_type = {
  .name  = "proc",
  .get_sb  = proc_get_sb,
  .kill_sb = proc_kill_sb,
```

Subject: Re: [PATCH 1/4] proc: fix NULL ->i_fop oops
Posted by Christoph Hellwig on Mon, 19 Nov 2007 12:51:39 GMT

View Forum Message <> Reply to Message

On Fri, Nov 16, 2007 at 06:06:51PM +0300, Alexey Dobriyan wrote:
> proc_kill_inodes() can clear ->i_fop in the middle of vfs_readdir resulting in
> NULL dereference during "file->f_op->readdir(file, buf, filler)".
>
> The solution is to remove proc_kill_inodes() completely:
> a) we don't have tricky modules implementing their tricky readdir hooks which
>   could keeping this revoke from hell.
> b) In a situation when module is gone but PDE still alive, standard readdir
>   will return only "." and "..", because pde->next was cleared by
>   remove_proc_entry().
> c) the race proc_kill_inode() destined to prevent is not completely fixed, just
>   race window made smaller, because vfs_readdir() is run without sb_lock held and
>   without file_list_lock held. Effectively, ->i_fop is cleared at random moment,

>   which can't fix properly anything.

Nice, getting rid of this is a very good step formwards. Unfortunately
we have another copy of this junk in
security/selinux/selinuxfs.c:sel_remove_entries() which would need the
same treatment.

Subject: Re: [PATCH 1/4] proc: fix NULL ->i_fop oops
Posted by Stephen Smalley on Tue, 20 Nov 2007 15:05:05 GMT

View Forum Message <> Reply to Message

On Mon, 2007-11-19 at 12:51 +0000, Christoph Hellwig wrote:
> On Fri, Nov 16, 2007 at 06:06:51PM +0300, Alexey Dobriyan wrote:
> > proc_kill_inodes() can clear ->i_fop in the middle of vfs_readdir resulting in
> > NULL dereference during "file->f_op->readdir(file, buf, filler)".
> >
> > The solution is to remove proc_kill_inodes() completely:
> > a) we don't have tricky modules implementing their tricky readdir hooks which
> >   could keeping this revoke from hell.
> > b) In a situation when module is gone but PDE still alive, standard readdir
> >   will return only "." and "..", because pde->next was cleared by
> >   remove_proc_entry().
> > c) the race proc_kill_inode() destined to prevent is not completely fixed, just
> >   race window made smaller, because vfs_readdir() is run without sb_lock held and
> >   without file_list_lock held. Effectively, ->i_fop is cleared at random moment,
> >   which can't fix properly anything.
>
> Nice, getting rid of this is a very good step formwards. Unfortunately
> we have another copy of this junk in
> security/selinux/selinuxfs.c:sel_remove_entries() which would need the
> same treatment.

Can't just be dropped completely for selinux - we need a way to drop
obsolete entries from the prior policy when we load a new policy.

Is the only real problem here the clearing of f_op? If so, we can
likely remove that from sel_remove_entries() without harm, and fix the
checks for it to use something more reliable.

--
Stephen Smalley
National Security Agency

Subject: Re: [PATCH 1/4] proc: fix NULL ->i_fop oops
Posted by Christoph Hellwig on Tue, 20 Nov 2007 15:17:31 GMT

View Forum Message <> Reply to Message

On Tue, Nov 20, 2007 at 10:05:05AM -0500, Stephen Smalley wrote:
> > Nice, getting rid of this is a very good step formwards. Unfortunately
> > we have another copy of this junk in
> > security/selinux/selinuxfs.c:sel_remove_entries() which would need the
> > same treatment.
>
> Can't just be dropped completely for selinux - we need a way to drop
> obsolete entries from the prior policy when we load a new policy.
>
> Is the only real problem here the clearing of f_op? If so, we can
> likely remove that from sel_remove_entries() without harm, and fix the
> checks for it to use something more reliable.

f_op removal is the biggest issue. It can't really work and this is the
last instance. But in general having some half-backed attempts at revoke
is never a good idea.

Subject: Re: [PATCH 1/4] proc: fix NULL ->i_fop oops
Posted by Stephen Smalley on Tue, 20 Nov 2007 15:22:37 GMT

View Forum Message <> Reply to Message

On Tue, 2007-11-20 at 15:17 +0000, Christoph Hellwig wrote:
> On Tue, Nov 20, 2007 at 10:05:05AM -0500, Stephen Smalley wrote:
> > > Nice, getting rid of this is a very good step formwards. Unfortunately
> > > we have another copy of this junk in
> > > security/selinux/selinuxfs.c:sel_remove_entries() which would need the
> > > same treatment.
> >
> > Can't just be dropped completely for selinux - we need a way to drop
> > obsolete entries from the prior policy when we load a new policy.
> >
> > Is the only real problem here the clearing of f_op? If so, we can
> > likely remove that from sel_remove_entries() without harm, and fix the
> > checks for it to use something more reliable.
>
> f_op removal is the biggest issue. It can't really work and this is the
> last instance. But in general having some half-backed attempts at revoke
> is never a good idea.

Yes, we're not trying to revoke per se, but just re-populate a set of
directories that represent elements of policy state on a policy
reload. /selinux/booleans is one example - a directory with one entry
per policy boolean defined by the policy. Old directory tree gets torn
down on each policy reload and replaced.

--
Stephen Smalley
National Security Agency

Subject: [patch 1/1] selinux: do not clear f_op when removing entries
Posted by Stephen Smalley on Wed, 21 Nov 2007 14:01:36 GMT

View Forum Message <> Reply to Message

On Tue, 2007-11-20 at 15:17 +0000, Christoph Hellwig wrote:
> On Tue, Nov 20, 2007 at 10:05:05AM -0500, Stephen Smalley wrote:
> > > Nice, getting rid of this is a very good step formwards. Unfortunately
> > > we have another copy of this junk in
> > > security/selinux/selinuxfs.c:sel_remove_entries() which would need the
> > > same treatment.
> >
> > Can't just be dropped completely for selinux - we need a way to drop
> > obsolete entries from the prior policy when we load a new policy.
> >
> > Is the only real problem here the clearing of f_op? If so, we can
> > likely remove that from sel_remove_entries() without harm, and fix the
> > checks for it to use something more reliable.
>
> f_op removal is the biggest issue. It can't really work and this is the
> last instance. But in general having some half-backed attempts at revoke
> is never a good idea.

Do not clear f_op when removing entries since it isn't safe to do.

Signed-off-by: Stephen Smalley <sds@tycho.nsa.gov>

```
 security/selinux/selinuxfs.c |   28 +---------------------------
 1 file changed, 1 insertion(+), 27 deletions(-)

diff --git a/security/selinux/selinuxfs.c b/security/selinux/selinuxfs.c
index f5f3e6d..ac6fe99 100644
--- a/security/selinux/selinuxfs.c
+++ b/security/selinux/selinuxfs.c
@@ -838,10 +838,6 @@ static ssize_t sel_read_bool(struct file *filep, char __user *buf,

  ret = -EFAULT;

- /* check to see if this file has been deleted */
- if (!filep->f_op)
-  goto out;
-
```

```
  if (count > PAGE_SIZE) {
   ret = -EINVAL;
   goto out;
@@ -882,10 +878,6 @@ static ssize_t sel_write_bool(struct file *filep, const char __user *buf,
  if (length)
   goto out;

- /* check to see if this file has been deleted */
- if (!filep->f_op)
-  goto out;
-
  if (count >= PAGE_SIZE) {
   length = -ENOMEM;
   goto out;
@@ -940,10 +932,6 @@ static ssize_t sel_commit_bools_write(struct file *filep,
  if (length)
   goto out;

- /* check to see if this file has been deleted */
- if (!filep->f_op)
-  goto out;
-
  if (count >= PAGE_SIZE) {
   length = -ENOMEM;
   goto out;
@@ -982,11 +970,9 @@ static const struct file_operations sel_commit_bools_ops = {
  .write          = sel_commit_bools_write,
 };

-/* partial revoke() from fs/proc/generic.c proc_kill_inodes */
 static void sel_remove_entries(struct dentry *de)
 {
- struct list_head *p, *node;
- struct super_block *sb = de->d_sb;
+ struct list_head *node;

  spin_lock(&dcache_lock);
  node = de->d_subdirs.next;
@@ -1006,18 +992,6 @@ static void sel_remove_entries(struct dentry *de)
  }

  spin_unlock(&dcache_lock);
-
- file_list_lock();
- list_for_each(p, &sb->s_files) {
-  struct file * filp = list_entry(p, struct file, f_u.fu_list);
-  struct dentry * dentry = filp->f_path.dentry;
-
-  if (dentry->d_parent != de) {
-   continue;
-  }
-  filp->f_op = NULL;
- }
- file_list_unlock();
 }

 #define BOOL_DIR_NAME "booleans"
```

--
Stephen Smalley
National Security Agency