Subject: [PATCH] lost content of /proc/sys/fs/binfmt_misc
Posted by den on Wed, 05 Dec 2007 14:35:18 GMT
View Forum Message <> Reply to Message

/proc/sys/fs/binfmt_misc dentry disappeared during d_revalidate.
d_revalidate only dentries from shadowed one and below.
http://bugzilla.kernel.org/show_bug.cgi?id=9504

CC: Eric W. Biederman <ebiederm@xmission.com>
CC: Marcus Better <marcus@better.se>
Signed-off-by: Denis V. Lunev <den@openvz.org>

```
diff --git a/fs/proc/generic.c b/fs/proc/generic.c
index 5fccfe2..1497ac4 100644
--- a/fs/proc/generic.c
+++ b/fs/proc/generic.c
@@ -380,12 +380,17 @@ static int proc_revalidate_dentry(struct dentry *dentry, struct
nameidata *nd)
  return 0;
 }

-static struct dentry_operations proc_dentry_operations =
+static struct dentry_operations proc_dentry_shadow_operations =
 {
  .d_delete = proc_delete_dentry,
  .d_revalidate = proc_revalidate_dentry,
 };

+static struct dentry_operations proc_dentry_operations =
+{
+ .d_delete = proc_delete_dentry,
+};
+
 /*
  * Don't create negative dentries here, return -ENOENT by hand
  * instead.
@@ -394,6 +399,7 @@ struct dentry *proc_lookup(struct inode * dir, struct dentry *dentry, struct
nam
 {
  struct inode *inode = NULL;
  struct proc_dir_entry * de;
+ int use_shadow = 0;
  int error = -ENOENT;

  lock_kernel();
@@ -406,8 +412,10 @@ struct dentry *proc_lookup(struct inode * dir, struct dentry *dentry, struct
nam
    if (!memcmp(dentry->d_name.name, de->name, de->namelen)) {
```

```
     unsigned int ino;

-    if (de->shadow_proc)
+    if (de->shadow_proc) {
      de = de->shadow_proc(current, de);
+     use_shadow = 1;
+    }
     ino = de->low_ino;
     de_get(de);
     spin_unlock(&proc_subdir_lock);
@@ -423,6 +431,8 @@ struct dentry *proc_lookup(struct inode * dir, struct dentry *dentry, struct
nam

  if (inode) {
   dentry->d_op = &proc_dentry_operations;
+  dentry->d_op = use_shadow ?
+   &proc_dentry_shadow_operations : dentry->d_parent->d_op;
   d_add(dentry, inode);
   return NULL;
  }
```

Subject: Re: [PATCH] lost content of /proc/sys/fs/binfmt_misc
Posted by ebiederm on Thu, 06 Dec 2007 10:17:16 GMT
View Forum Message <> Reply to Message

"Denis V. Lunev" <den@openvz.org> writes:

> /proc/sys/fs/binfmt_misc dentry disappeared during d_revalidate.
> d_revalidate only dentries from shadowed one and below.
> http://bugzilla.kernel.org/show_bug.cgi?id=9504

Denis this is decent except it doesn't completely close the possibility of
leaking mounts. It just refuses to leak mounts on everything except /proc/net.

Eric

Subject: [PATCH] proc: Do not invalidate dentries with submounts
Posted by ebiederm on Thu, 06 Dec 2007 10:22:50 GMT
View Forum Message <> Reply to Message

If the dcache path to a mount point is ever broken it becomes
impossible to unmount it, and we leak a vfsmount. Therefore it is not
valid to invalidate dentries with mount points at or below them.

This patch uses the have_submounts test as the other network

filesystem revalidate routines do.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 fs/proc/base.c | 9 +++++++++
 fs/proc/generic.c | 5 +++++
 2 files changed, 14 insertions(+), 0 deletions(-)

diff --git a/fs/proc/base.c b/fs/proc/base.c
index 0e71707..552d752 100644
--- a/fs/proc/base.c
+++ b/fs/proc/base.c
@@ -1216,6 +1216,9 @@ static int pid_revalidate(struct dentry *dentry, struct nameidata *nd)
   put_task_struct(task);
   return 1;
  }
+ /* Force validity if something is mounted under us */
+ if (inode && S_ISDIR(inode->i_mode) && have_submounts(dentry))
+ return 1;
  d_drop(dentry);
  return 0;
 }
@@ -1393,6 +1396,9 @@ static int tid_fd_revalidate(struct dentry *dentry, struct nameidata *nd)
   }
   put_task_struct(task);
  }
+ /* Force validity if something is mounted under us */
+ if (inode && S_ISDIR(inode->i_mode) && have_submounts(dentry))
+ return 1;
  d_drop(dentry);
  return 0;
 }
@@ -2056,6 +2062,9 @@ static int proc_base_revalidate(struct dentry *dentry, struct nameidata
*nd)
   put_task_struct(task);
   return 1;
  }
+ /* Force validity if something is mounted under us */
+ if (inode && S_ISDIR(inode->i_mode) && have_submounts(dentry))
+ return 1;
  d_drop(dentry);
  return 0;
 }
diff --git a/fs/proc/generic.c b/fs/proc/generic.c
index 4abd568..233dcdc 100644
--- a/fs/proc/generic.c
+++ b/fs/proc/generic.c
@@ -370,6 +370,11 @@ static int proc_delete_dentry(struct dentry * dentry)

```
 static int proc_revalidate_dentry(struct dentry *dentry, struct nameidata *nd)
 {
+ struct inode *inode = dentry->d_inode;
+
+ /* Force validity if something is mounted under us */
+ if (inode && S_ISDIR(inode->i_mode) && have_submounts(dentry))
+  return 1;
  d_drop(dentry);
  return 0;
 }
--
1.5.3.rc6.17.g1911
```

Subject: Re: [PATCH] proc: Do not invalidate dentries with submounts
Posted by den on Thu, 06 Dec 2007 10:30:45 GMT
View Forum Message <> Reply to Message

you have changed the behavior of revalidation by shadows. I think it
will be better to restore it and keep new one for shadows (and below)
only, which has been done by my yesterday patch.

Regards,
 Den

Eric W. Biederman wrote:
> If the dcache path to a mount point is ever broken it becomes
> impossible to unmount it, and we leak a vfsmount. Therefore it is not
> valid to invalidate dentries with mount points at or below them.
>
> This patch uses the have_submounts test as the other network
> filesystem revalidate routines do.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
> ---
> fs/proc/base.c | 9 +++++++++
> fs/proc/generic.c | 5 +++++
> 2 files changed, 14 insertions(+), 0 deletions(-)
>
> diff --git a/fs/proc/base.c b/fs/proc/base.c
> index 0e71707..552d752 100644
> --- a/fs/proc/base.c
> +++ b/fs/proc/base.c
> @@ -1216,6 +1216,9 @@ static int pid_revalidate(struct dentry *dentry, struct nameidata *nd)
>   put_task_struct(task);
>   return 1;
>   }

> + /* Force validity if something is mounted under us */
> + if (inode && S_ISDIR(inode->i_mode) && have_submounts(dentry))
> + return 1;
> d_drop(dentry);
> return 0;
> }
> @@ -1393,6 +1396,9 @@ static int tid_fd_revalidate(struct dentry *dentry, struct nameidata
*nd)
> }
> put_task_struct(task);
> }
> + /* Force validity if something is mounted under us */
> + if (inode && S_ISDIR(inode->i_mode) && have_submounts(dentry))
> + return 1;
> d_drop(dentry);
> return 0;
> }
> @@ -2056,6 +2062,9 @@ static int proc_base_revalidate(struct dentry *dentry, struct
nameidata *nd)
> put_task_struct(task);
> return 1;
> }
> + /* Force validity if something is mounted under us */
> + if (inode && S_ISDIR(inode->i_mode) && have_submounts(dentry))
> + return 1;
> d_drop(dentry);
> return 0;
> }
> diff --git a/fs/proc/generic.c b/fs/proc/generic.c
> index 4abd568..233dcdc 100644
> --- a/fs/proc/generic.c
> +++ b/fs/proc/generic.c
> @@ -370,6 +370,11 @@ static int proc_delete_dentry(struct dentry * dentry)
>
> static int proc_revalidate_dentry(struct dentry *dentry, struct nameidata *nd)
> {
> + struct inode *inode = dentry->d_inode;
> +
> + /* Force validity if something is mounted under us */
> + if (inode && S_ISDIR(inode->i_mode) && have_submounts(dentry))
> + return 1;
> d_drop(dentry);
> return 0;
> }

Subject: Re: [PATCH] proc: Do not invalidate dentries with submounts

"Denis V. Lunev" <den@sw.ru> writes:

> you have changed the behavior of revalidation by shadows. I think it
> will be better to restore it and keep new one for shadows (and below)
> only, which has been done by my yesterday patch.

- I think it is better to move forward rather then back.
- The old proc dentry caching behavior is actually too aggressive, and has
  problem corner cases. Keeping the dentries when we have something mounted
  on top is a trade off that is the least of two evils.
- My change fixes the mount leak on all of /proc not just on /proc/generic.

What you did is a hack that restored the old slightly buggy behavior. Which
is fine if we can't find anything better. It is not code that is on the
path towards a /proc that properly caches it's dentries.

With the old behavior a random user space application can open a file or
a directory in /proc pinning it's dcache entry. Then the module supplying
that open file can be removed and reinserted. Until the user space application
removes reference to that /proc file all you will be able to find is the
version of the file from before /proc was removed.

That sounds like a way to trigger nasty behavior to me. I would like
to remove that possibility from the kernel if I can.

Eric