Subject: [PATCH v3 3/4] SUNRPC: check RPC inode's pipe reference before dereferencing
Posted by Stanislav Kinsbursky on Mon, 27 Feb 2012 18:05:45 GMT
View Forum Message <> Reply to Message

There are 2 tightly bound objects: pipe data (created for kernel needs, has
reference to dentry, which depends on PipeFS mount/umount) and PipeFS
dentry/inode pair (created on mount for user-space needs). They both
independently may have or have not a valid reference to each other.
This means, that we have to make sure, that pipe->dentry reference is valid on
upcalls, and dentry->pipe reference is valid on downcalls. The latter check is
absent - my fault.
IOW, PipeFS dentry can be opened by some process (rpc.idmapd for example), but
it's pipe data can belong to NFS mount, which was unmounted already and thus
pipe data was destroyed.
To fix this, pipe reference have to be set to NULL on rpc_unlink() and checked
on PipeFS file operations instead of pipe->dentry check.

Note: PipeFS "poll" file operation will be updated in next patch, because it's
logic is more complicated.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

---
 net/sunrpc/rpc_pipe.c | 32 ++++++++++++++++++++++----------
 1 files changed, 19 insertions(+), 13 deletions(-)

diff --git a/net/sunrpc/rpc_pipe.c b/net/sunrpc/rpc_pipe.c
index 6873c9b..b67b2ae 100644
--- a/net/sunrpc/rpc_pipe.c
+++ b/net/sunrpc/rpc_pipe.c
@@ -174,6 +174,7 @@ rpc_close_pipes(struct inode *inode)
   pipe->ops->release_pipe(inode);
  cancel_delayed_work_sync(&pipe->queue_timeout);
  rpc_inode_setowner(inode, NULL);
+ RPC_I(inode)->pipe = NULL;
  mutex_unlock(&inode->i_mutex);
 }

@@ -203,12 +204,13 @@ rpc_destroy_inode(struct inode *inode)
 static int
 rpc_pipe_open(struct inode *inode, struct file *filp)
 {
- struct rpc_pipe *pipe = RPC_I(inode)->pipe;
+ struct rpc_pipe *pipe;
  int first_open;
  int res = -ENXIO;

```
  mutex_lock(&inode->i_mutex);
- if (pipe->dentry == NULL)
+ pipe = RPC_I(inode)->pipe;
+ if (pipe == NULL)
    goto out;
  first_open = pipe->nreaders == 0 && pipe->nwriters == 0;
  if (first_open && pipe->ops->open_pipe) {
@@ -229,12 +231,13 @@ out:
 static int
 rpc_pipe_release(struct inode *inode, struct file *filp)
 {
- struct rpc_pipe *pipe = RPC_I(inode)->pipe;
+ struct rpc_pipe *pipe;
  struct rpc_pipe_msg *msg;
  int last_close;

  mutex_lock(&inode->i_mutex);
- if (pipe->dentry == NULL)
+ pipe = RPC_I(inode)->pipe;
+ if (pipe == NULL)
    goto out;
  msg = filp->private_data;
  if (msg != NULL) {
@@ -270,12 +273,13 @@ static ssize_t
 rpc_pipe_read(struct file *filp, char __user *buf, size_t len, loff_t *offset)
 {
  struct inode *inode = filp->f_path.dentry->d_inode;
- struct rpc_pipe *pipe = RPC_I(inode)->pipe;
+ struct rpc_pipe *pipe;
  struct rpc_pipe_msg *msg;
  int res = 0;

  mutex_lock(&inode->i_mutex);
- if (pipe->dentry == NULL) {
+ pipe = RPC_I(inode)->pipe;
+ if (pipe == NULL) {
    res = -EPIPE;
    goto out_unlock;
  }
@@ -313,13 +317,12 @@ static ssize_t
 rpc_pipe_write(struct file *filp, const char __user *buf, size_t len, loff_t *offset)
 {
  struct inode *inode = filp->f_path.dentry->d_inode;
- struct rpc_pipe *pipe = RPC_I(inode)->pipe;
  int res;

  mutex_lock(&inode->i_mutex);
  res = -EPIPE;
```

```
- if (pipe->dentry != NULL)
-  res = pipe->ops->downcall(filp, buf, len);
+ if (RPC_I(inode)->pipe != NULL)
+  res = RPC_I(inode)->pipe->ops->downcall(filp, buf, len);
  mutex_unlock(&inode->i_mutex);
  return res;
 }
@@ -344,16 +347,18 @@ static long
 rpc_pipe_ioctl(struct file *filp, unsigned int cmd, unsigned long arg)
 {
  struct inode *inode = filp->f_path.dentry->d_inode;
- struct rpc_pipe *pipe = RPC_I(inode)->pipe;
+ struct rpc_pipe *pipe;
  int len;

  switch (cmd) {
  case FIONREAD:
-  spin_lock(&pipe->lock);
-  if (pipe->dentry == NULL) {
-   spin_unlock(&pipe->lock);
+  mutex_lock(&inode->i_mutex);
+  pipe = RPC_I(inode)->pipe;
+  if (pipe == NULL) {
+   mutex_unlock(&inode->i_mutex);
    return -EPIPE;
   }
+  spin_lock(&pipe->lock);
   len = pipe->pipelen;
   if (filp->private_data) {
    struct rpc_pipe_msg *msg;
@@ -361,6 +366,7 @@ rpc_pipe_ioctl(struct file *filp, unsigned int cmd, unsigned long arg)
    len += msg->len - msg->copied;
   }
   spin_unlock(&pipe->lock);
+  mutex_unlock(&inode->i_mutex);
   return put_user(len, (int __user *)arg);
  default:
   return -EINVAL;
```