
Subject: [PATCH v2] SUNRPC: set desired file system root before connecting local transports

Posted by [Stanislav Kinsbursky](#) on Wed, 07 Mar 2012 16:25:17 GMT

[View Forum Message](#) <> [Reply to Message](#)

v2: added Alexander Viro and Christoph Hellwig to recipience.

Today, there is a problem in connecting of local SUNRPC thansports. These transports uses UNIX sockets and connection itself is done by rpciod workqueue. But UNIX sockets lookup is done in context of process file system root. I.e. all local thunsports are connecting in rpciod context.

This works nice until we will try to mount NFS from process with other root - for example in container. This container can have it's own (nested) root and rpcbind process, listening on it's own unix sockets. But NFS mount attempt in this container will register new service (Lockd for example) in global rpcbind - not containers's one.

This patch solves the problem by switching rpciod kernel thread's file system root to right one (stored on transport) while connecting of local transports.

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

```
---
fs/fs_struct.c      | 1 +
net/sunrpc/xprtsock.c | 32 ++++++
2 files changed, 31 insertions(+), 2 deletions(-)
```

```
diff --git a/fs/fs_struct.c b/fs/fs_struct.c
```

```
index 78b519c..0f984c3 100644
```

```
--- a/fs/fs_struct.c
```

```
+++ b/fs/fs_struct.c
```

```
@@ -36,6 +36,7 @@ void set_fs_root(struct fs_struct *fs, struct path *path)
    if (old_root.dentry)
        path_put_longterm(&old_root);
}
```

```
+EXPORT_SYMBOL_GPL(set_fs_root);
```

```
/*
```

```
 * Replace the fs->{pwmnt,pwd} with {mnt,dentry}. Put the old values.
```

```
diff --git a/net/sunrpc/xprtsock.c b/net/sunrpc/xprtsock.c
```

```
index 4c8281d..c94c181 100644
```

```
--- a/net/sunrpc/xprtsock.c
```

```
+++ b/net/sunrpc/xprtsock.c
```

```
@@ -37,6 +37,7 @@
```

```
#include <linux/sunrpc/svcsock.h>
```

```
#include <linux/sunrpc/xprtsock.h>
```

```
#include <linux/file.h>
```

```
+#include <linux/fs_struct.h>
```

```
#ifdef CONFIG_SUNRPC_BACKCHANNEL
```

```

#include <linux/sunrpc/bc_xprt.h>
#endif
@@ -255,6 +256,11 @@ struct sock_xprt {
    void (*old_state_change)(struct sock *);
    void (*old_write_space)(struct sock *);
    void (*old_error_report)(struct sock *);
+
+ /*
+  * Saved transport creator root. Required for local transports only.
+  */
+ struct path root;
};

/*
@@ -1891,6 +1897,7 @@ static void xs_local_setup_socket(struct work_struct *work)
    struct rpc_xprt *xprt = &transport->xprt;
    struct socket *sock;
    int status = -EIO;
+ struct path root;

    if (xprt->shutdown)
        goto out;
@@ -1908,7 +1915,14 @@ static void xs_local_setup_socket(struct work_struct *work)
    dprintk("RPC:    worker connecting xprt %p via AF_LOCAL to %s\n",
        xprt, xprt->address_strings[RPC_DISPLAY_ADDR]);

+ get_fs_root(current->fs, &root);
+ set_fs_root(current->fs, &transport->root);
+
    status = xs_local_finish_connecting(xprt, sock);
+
+ set_fs_root(current->fs, &root);
+ path_put(&root);
+
    switch (status) {
    case 0:
        dprintk("RPC:    xprt %p connected to %s\n",
@@ -2213,6 +2227,18 @@ static void xs_connect(struct rpc_task *task)
    }
}

+static void xs_local_destroy(struct rpc_xprt *xprt)
+{
+ struct sock_xprt *transport = container_of(xprt, struct sock_xprt, xprt);
+ struct path root = transport->root;
+
+ dprintk("RPC:    xs_local_destroy xprt %p\n", xprt);
+

```

```

+ xs_destroy(xprt);
+
+ path_put(&root);
+}
+
/**
 * xs_local_print_stats - display AF_LOCAL socket-specific stats
 * @xprt: rpc_xprt struct containing statistics
@@ -2431,7 +2457,7 @@ static struct rpc_xprt_ops xs_local_ops = {
 .send_request = xs_local_send_request,
 .set_retrans_timeout = xprt_set_retrans_timeout_def,
 .close = xs_close,
- .destroy = xs_destroy,
+ .destroy = xs_local_destroy,
 .print_stats = xs_local_print_stats,
};

@@ -2606,8 +2632,10 @@ static struct rpc_xprt *xs_setup_local(struct xprt_create *args)
 dprintk("RPC:    set up xprt to %s via AF_LOCAL\n",
        xprt->address_strings[RPC_DISPLAY_ADDR]);

- if (try_module_get(THIS_MODULE))
+ if (try_module_get(THIS_MODULE)) {
+ get_fs_root(current->fs, &transport->root);
    return xprt;
+ }
    ret = ERR_PTR(-EINVAL);
out_err:
    xprt_free(xprt);

```
