Subject: [RFC PATCH 4/5] net: fbind() for unix sockets protocol operations introduced
Posted by Stanislav Kinsbursky on Wed, 15 Aug 2012 16:22:20 GMT
View Forum Message <> Reply to Message

Path for unix_address is taken from passed file.
File inode have to be socket.
Since no sunaddr is present, addr->name is constructed at the place. It
obviously means, then path name can be truncated is it's longer then
UNIX_MAX_PATH.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 net/unix/af_unix.c | 78 ++-
 1 files changed, 76 insertions(+), 2 deletions(-)

```
diff --git a/net/unix/af_unix.c b/net/unix/af_unix.c
index b26200d..2f34c9d 100644
--- a/net/unix/af_unix.c
+++ b/net/unix/af_unix.c
@@ -286,12 +286,11 @@ static inline struct sock *unix_find_socket_byname(struct net *net,
  return s;
 }

-static struct sock *unix_find_socket_byinode(struct inode *i)
+static struct sock *__unix_find_socket_byinode(struct inode *i)
 {
  struct sock *s;
  struct hlist_node *node;

- spin_lock(&unix_table_lock);
  sk_for_each(s, node,
       &unix_socket_table[i->i_ino & (UNIX_HASH_SIZE - 1)]) {
   struct dentry *dentry = unix_sk(s)->path.dentry;
@@ -303,6 +302,15 @@ static struct sock *unix_find_socket_byinode(struct inode *i)
  }
  s = NULL;
 found:
+ return s;
+}
+
+static struct sock *unix_find_socket_byinode(struct inode *i)
+{
+ struct sock *s;
+
+ spin_lock(&unix_table_lock);
+ s = __unix_find_socket_byinode(i);
  spin_unlock(&unix_table_lock);
```

```
  return s;
 }
@@ -502,6 +510,7 @@ out:

 static int unix_release(struct socket *);
 static int unix_bind(struct socket *, struct sockaddr *, int);
+static int unix_fbind(struct file *file, struct socket *sock);
 static int unix_stream_connect(struct socket *, struct sockaddr *,
            int addr_len, int flags);
 static int unix_socketpair(struct socket *, struct socket *);
@@ -542,6 +551,7 @@ static const struct proto_ops unix_stream_ops = {
  .owner = THIS_MODULE,
  .release = unix_release,
  .bind =  unix_bind,
+ .fbind = unix_fbind,
  .connect = unix_stream_connect,
  .socketpair = unix_socketpair,
  .accept = unix_accept,
@@ -564,6 +574,7 @@ static const struct proto_ops unix_dgram_ops = {
  .owner = THIS_MODULE,
  .release = unix_release,
  .bind =  unix_bind,
+ .fbind = unix_fbind,
  .connect = unix_dgram_connect,
  .socketpair = unix_socketpair,
  .accept = sock_no_accept,
@@ -586,6 +597,7 @@ static const struct proto_ops unix_seqpacket_ops = {
  .owner = THIS_MODULE,
  .release = unix_release,
  .bind =  unix_bind,
+ .fbind = unix_fbind,
  .connect = unix_stream_connect,
  .socketpair = unix_socketpair,
  .accept = unix_accept,
@@ -843,6 +855,68 @@ static int __unix_add_sock(struct path *path, struct sock *sk,

 }

+static int unix_fbind(struct file *f, struct socket *sock)
+{
+ struct sock *sk = sock->sk, *tmp;
+ struct unix_sock *u = unix_sk(sk);
+ struct unix_address *addr;
+ struct path *path = &f->f_path;
+ struct inode *inode = path->dentry->d_inode;
+ char *buf, *name;
+ int err;
+
```

```
+ err = -ENOTSOCK;
+ if (!S_ISSOCK(inode->i_mode))
+  goto out;
+
+ mutex_lock(&u->readlock);
+
+ err = -EINVAL;
+ if (u->addr)
+  goto out_up;
+
+ err = -ENOMEM;
+ buf = (char*)__get_free_page(GFP_KERNEL);
+ if (!buf)
+  goto out_up;
+
+ err = -EFAULT;
+ name = d_path(path, buf, PAGE_SIZE);
+ if (IS_ERR(name))
+  goto out_up_page;
+
+ addr = kmalloc(sizeof(*addr) + sizeof(struct sockaddr_un), GFP_KERNEL);
+ if (!addr)
+  goto out_up_page;
+
+ addr->name->sun_family = AF_UNIX;
+ addr->len = min(strlen(name), (size_t)UNIX_PATH_MAX);
+ memcpy(addr->name->sun_path, name, addr->len);
+ atomic_set(&addr->refcnt, 1);
+
+ spin_lock(&unix_table_lock);
+
+ err = -EADDRINUSE;
+ tmp = __unix_find_socket_byinode(inode);
+ if (tmp) {
+  sock_put(tmp);
+  unix_release_addr(addr);
+  goto out_unlock;
+ }
+
+ err = __unix_add_sock(path, sk, addr, 0);
+ path_get(path);
+
+out_unlock:
+ spin_unlock(&unix_table_lock);
+out_up_page:
+ free_page((unsigned long)buf);
+out_up:
+ mutex_unlock(&u->readlock);
```

```
+out:
+ return err;
+}
+
 static int unix_bind(struct socket *sock, struct sockaddr *uaddr, int addr_len)
 {
  struct sock *sk = sock->sk;
```