
Subject: [PATCH 5/5] net: Make AF_UNIX per network namespace safe.

Posted by [ebiederm](#) on Sat, 29 Sep 2007 01:08:05 GMT

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

Because of the global nature of garbage collection, and because of the cost of per namespace hash tables `unix_socket_table` has been kept global. With a filter added on lookups so we don't see sockets from the wrong namespace.

Currently I don't fold the namespace into the hash so multiple namespaces using the same socket name will be guaranteed a hash collision.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

net/unix/af_unix.c | 113 ++++++++++++++++++++++++++++++++++++++-----
1 files changed, 86 insertions(+), 27 deletions(-)

diff --git a/net/unix/af_unix.c b/net/unix/af_unix.c

index 10e7312..2792ef2 100644

--- a/net/unix/af_unix.c

+++ b/net/unix/af_unix.c

@@ -270,7 +270,8 @@ static inline void unix_insert_socket(struct hlist_head *list, struct sock *sk)

spin_unlock(&unix_table_lock);
}

-static struct sock *__unix_find_socket_byname(struct sockaddr_un *sunname,

+static struct sock *__unix_find_socket_byname(struct net *net,

+ struct sockaddr_un *sunname,
+ int len, int type, unsigned hash)

{
struct sock *s;

@@ -279,6 +280,9 @@ static struct sock *__unix_find_socket_byname(struct sockaddr_un *sunname,

sk_for_each(s, node, &unix_socket_table[hash ^ type]) {
struct unix_sock *u = unix_sk(s);

+ if (s->sk_net != net)

+ continue;

+

if (u->addr->len == len &&
!memcmp(u->addr->name, sunname, len))
goto found;

@@ -288,21 +292,22 @@ found:

return s;
}

```

-static inline struct sock *unix_find_socket_byname(struct sockaddr_un *sunname,
+static inline struct sock *unix_find_socket_byname(struct net *net,
+      struct sockaddr_un *sunname,
+      int len, int type,
+      unsigned hash)
{
    struct sock *s;

    spin_lock(&unix_table_lock);
- s = __unix_find_socket_byname(sunname, len, type, hash);
+ s = __unix_find_socket_byname(net, sunname, len, type, hash);
    if (s)
        sock_hold(s);
    spin_unlock(&unix_table_lock);
    return s;
}

-static struct sock *unix_find_socket_byinode(struct inode *i)
+static struct sock *unix_find_socket_byinode(struct net *net, struct inode *i)
{
    struct sock *s;
    struct hlist_node *node;
@@ -312,6 +317,9 @@ static struct sock *unix_find_socket_byinode(struct inode *i)
    &unix_socket_table[i->i_ino & (UNIX_HASH_SIZE - 1)]) {
        struct dentry *dentry = unix_sk(s)->dentry;

+ if (s->sk_net != net)
+     continue;
+
        if(dentry && dentry->d_inode == i)
        {
            sock_hold(s);
@@ -630,9 +638,6 @@ out:

static int unix_create(struct net *net, struct socket *sock, int protocol)
{
- if (net != &init_net)
-     return -EAFNOSUPPORT;
-
    if (protocol && protocol != PF_UNIX)
        return -EPROTONOSUPPORT;

@@ -676,6 +681,7 @@ static int unix_release(struct socket *sock)
static int unix_autobind(struct socket *sock)
{
    struct sock *sk = sock->sk;
+ struct net *net = sk->sk_net;
    struct unix_sock *u = unix_sk(sk);

```

```

static u32 ordernum = 1;
struct unix_address * addr;
@@ -702,7 +708,7 @@ retry:
    spin_lock(&unix_table_lock);
    ordernum = (ordernum+1)&0xFFFFF;

- if (__unix_find_socket_byname(addr->name, addr->len, sock->type,
+ if (__unix_find_socket_byname(net, addr->name, addr->len, sock->type,
    addr->hash)) {
    spin_unlock(&unix_table_lock);
    /* Sanity yield. It is unusual case, but yet... */
@@ -722,7 +728,8 @@ out: mutex_unlock(&u->readlock);
    return err;
}

-static struct sock *unix_find_other(struct sockaddr_un *sunname, int len,
+static struct sock *unix_find_other(struct net *net,
+    struct sockaddr_un *sunname, int len,
+    int type, unsigned hash, int *error)
{
    struct sock *u;
@@ -740,7 +747,7 @@ static struct sock *unix_find_other(struct sockaddr_un *sunname, int len,
    err = -ECONNREFUSED;
    if (!S_ISSOCK(nd.dentry->d_inode->i_mode))
        goto put_fail;
- u=unix_find_socket_byinode(nd.dentry->d_inode);
+ u=unix_find_socket_byinode(net, nd.dentry->d_inode);
    if (!u)
        goto put_fail;

@@ -756,7 +763,7 @@ static struct sock *unix_find_other(struct sockaddr_un *sunname, int len,
    }
} else {
    err = -ECONNREFUSED;
- u=unix_find_socket_byname(sunname, len, type, hash);
+ u=unix_find_socket_byname(net, sunname, len, type, hash);
    if (u) {
        struct dentry *dentry;
        dentry = unix_sk(u)->dentry;
@@ -778,6 +785,7 @@ fail:
static int unix_bind(struct socket *sock, struct sockaddr *uaddr, int addr_len)
{
    struct sock *sk = sock->sk;
+ struct net *net = sk->sk_net;
    struct unix_sock *u = unix_sk(sk);
    struct sockaddr_un *sunaddr=(struct sockaddr_un *)uaddr;
    struct dentry * dentry = NULL;
@@ -852,7 +860,7 @@ static int unix_bind(struct socket *sock, struct sockaddr *uaddr, int

```

addr_len)

```
if (!sunaddr->sun_path[0]) {
    err = -EADDRINUSE;
- if (__unix_find_socket_byname(sunaddr, addr_len,
+ if (__unix_find_socket_byname(net, sunaddr, addr_len,
    sk->sk_type, hash)) {
    unix_release_addr(addr);
    goto out_unlock;
@@ -918,6 +926,7 @@ static int unix_dgram_connect(struct socket *sock, struct sockaddr *addr,
    int alen, int flags)
{
    struct sock *sk = sock->sk;
+ struct net *net = sk->sk_net;
    struct sockaddr_un *sunaddr=(struct sockaddr_un*)addr;
    struct sock *other;
    unsigned hash;
@@ -934,7 +943,7 @@ static int unix_dgram_connect(struct socket *sock, struct sockaddr *addr,
    goto out;
```

restart:

```
- other=unix_find_other(sunaddr, alen, sock->type, hash, &err);
+ other=unix_find_other(net, sunaddr, alen, sock->type, hash, &err);
    if (!other)
        goto out;
```

```
@@ -1014,6 +1023,7 @@ static int unix_stream_connect(struct socket *sock, struct sockaddr
*uaddr,
{
    struct sockaddr_un *sunaddr=(struct sockaddr_un *)uaddr;
    struct sock *sk = sock->sk;
+ struct net *net = sk->sk_net;
    struct unix_sock *u = unix_sk(sk), *newu, *otheru;
    struct sock *newsk = NULL;
    struct sock *other = NULL;
@@ -1053,7 +1063,7 @@ static int unix_stream_connect(struct socket *sock, struct sockaddr
*uaddr,
```

restart:

```
/* Find listening sock. */
- other = unix_find_other(sunaddr, addr_len, sk->sk_type, hash, &err);
+ other = unix_find_other(net, sunaddr, addr_len, sk->sk_type, hash, &err);
    if (!other)
        goto out;
```

```
@@ -1329,6 +1339,7 @@ static int unix_dgram_sendmsg(struct kiocb *kiocb, struct socket
*sock,
{
```

```

    struct sock_iocb *siocb = kiocb_to_siocb(kiocb);
    struct sock *sk = sock->sk;
+ struct net *net = sk->sk_net;
    struct unix_sock *u = unix_sk(sk);
    struct sockaddr_un *sunaddr=msg->msg_name;
    struct sock *other = NULL;
@@ -1392,7 +1403,7 @@ restart:
    if (sunaddr == NULL)
        goto out_free;

- other = unix_find_other(sunaddr, namelen, sk->sk_type,
+ other = unix_find_other(net, sunaddr, namelen, sk->sk_type,
    hash, &err);
    if (other==NULL)
        goto out_free;
@@ -1998,12 +2009,18 @@ static unsigned int unix_poll(struct file * file, struct socket *sock,
poll_tabl

```

```

#ifdef CONFIG_PROC_FS
-static struct sock *unix_seq_idx(int *iter, loff_t pos)
+struct unix_iter_state {
+ struct net *net;
+ int i;
+};
+static struct sock *unix_seq_idx(struct unix_iter_state *iter, loff_t pos)
{
    loff_t off = 0;
    struct sock *s;

- for (s = first_unix_socket(iter); s; s = next_unix_socket(iter, s)) {
+ for (s = first_unix_socket(&iter->i); s; s = next_unix_socket(&iter->i, s)) {
+ if (s->sk_net != iter->net)
+ continue;
    if (off == pos)
        return s;
    ++off;
@@ -2014,17 +2031,24 @@ static struct sock *unix_seq_idx(int *iter, loff_t pos)

static void *unix_seq_start(struct seq_file *seq, loff_t *pos)
{
+ struct unix_iter_state *iter = seq->private;
    spin_lock(&unix_table_lock);
- return *pos ? unix_seq_idx(seq->private, *pos - 1) : ((void *) 1);
+ return *pos ? unix_seq_idx(iter, *pos - 1) : ((void *) 1);
}

static void *unix_seq_next(struct seq_file *seq, void *v, loff_t *pos)

```

```

{
+ struct unix_iter_state *iter = seq->private;
+ struct sock *sk = v;
  ++*pos;

  if (v == (void *)1)
- return first_unix_socket(seq->private);
- return next_unix_socket(seq->private, v);
+ sk = first_unix_socket(&iter->i);
+ else
+ sk = next_unix_socket(&iter->i, sk);
+ while (sk && (sk->sk_net != iter->net))
+ sk = next_unix_socket(&iter->i, sk);
+ return sk;
}

static void unix_seq_stop(struct seq_file *seq, void *v)
@@ -2088,7 +2112,7 @@ static int unix_seq_open(struct inode *inode, struct file *file)
{
  struct seq_file *seq;
  int rc = -ENOMEM;
- int *iter = kmalloc(sizeof(int), GFP_KERNEL);
+ struct unix_iter_state *iter = kmalloc(sizeof(*iter), GFP_KERNEL);

  if (!iter)
    goto out;
@@ -2099,7 +2123,12 @@ static int unix_seq_open(struct inode *inode, struct file *file)

  seq = file->private_data;
  seq->private = iter;
- *iter = 0;
+ iter->i = 0;
+ iter->net = get_proc_net(inode);
+ if (!iter->net) {
+   seq_release_private(inode, file);
+   rc = -ENXIO;
+ }
  out:
  return rc;
  out_kfree:
@@ -2107,12 +2136,20 @@ out_kfree:
  goto out;
}

+static int unix_seq_release(struct inode *inode, struct file *file)
+{
+ struct seq_file *seq = file->private_data;
+ struct unix_iter_state *iter = seq->private;

```

```

+ put_net(iter->net);
+ return seq_release_private(inode, file);
+}
+
static const struct file_operations unix_seq_fops = {
    .owner = THIS_MODULE,
    .open = unix_seq_open,
    .read = seq_read,
    .llseek = seq_lseek,
- .release = seq_release_private,
+ .release = unix_seq_release,
};

#ifdef
@@ -2123,6 +2160,30 @@ static struct net_proto_family unix_family_ops = {
    .owner = THIS_MODULE,
};

+
+static int unix_net_init(struct net *net)
+{
+ int error = -ENOMEM;
+
+ #ifdef CONFIG_PROC_FS
+ if (!proc_net_fops_create(net, "unix", 0, &unix_seq_fops))
+ goto out;
+ #endif
+ error = 0;
+out:
+ return 0;
+}
+
+static void unix_net_exit(struct net *net)
+{
+ proc_net_remove(net, "unix");
+}
+
+static struct pernet_operations unix_net_ops = {
+ .init = unix_net_init,
+ .exit = unix_net_exit,
+};
+
static int __init af_unix_init(void)
{
    int rc = -1;
@@ -2138,9 +2199,7 @@ static int __init af_unix_init(void)
}

```

```

    sock_register(&unix_family_ops);
-#ifdef CONFIG_PROC_FS
- proc_net_fops_create(&init_net, "unix", 0, &unix_seq_fops);
-#endif
+ register_pernet_subsys(&unix_net_ops);
    unix_sysctl_register();
out:
    return rc;
@@ -2150,8 +2209,8 @@ static void __exit af_unix_exit(void)
{
    sock_unregister(PF_UNIX);
    unix_sysctl_unregister();
- proc_net_remove(&init_net, "unix");
    proto_unregister(&unix_proto);
+ unregister_pernet_subsys(&unix_net_ops);
}

module_init(af_unix_init);
--
1.5.3.rc6.17.g1911

```

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
