

---

Subject: [PATCH RFC 30/31] net: Make AF\_UNIX per network namespace safe.  
Posted by [ebiederm](#) on Thu, 25 Jan 2007 19:00:32 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

From: Eric W. Biederman <[ebiederm@xmission.com](mailto:ebiederm@xmission.com)> - unquoted

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](mailto:ebiederm@xmission.com)>

---

```
include/net/af_unix.h      | 10 +---
net/unix/af_unix.c         | 116 ++++++-----
net/unix/sysctl_net_unix.c | 24 +++++---
3 files changed, 103 insertions(+), 47 deletions(-)
```

```
diff --git a/include/net/af_unix.h b/include/net/af_unix.h
index c0398f5..1f40dd2 100644
```

```
--- a/include/net/af_unix.h
```

```
+++ b/include/net/af_unix.h
```

```
@@ -89,12 +89,12 @@ struct unix_sock {
#define unix_sk(__sk) ((struct unix_sock *)__sk)
```

```
#ifdef CONFIG_SYSCTL
-extern int sysctl_unix_max_dgram_qlen;
-extern void unix_sysctl_register(void);
-extern void unix_sysctl_unregister(void);
+DECLARE_PER_NET(int, sysctl_unix_max_dgram_qlen);
+extern void unix_sysctl_register(net_t net);
+extern void unix_sysctl_unregister(net_t net);
#else
-static inline void unix_sysctl_register(void) {}
-static inline void unix_sysctl_unregister(void) {}
+static inline void unix_sysctl_register(net_t net) {}
+static inline void unix_sysctl_unregister(net_t net) {}
#endif
#endif
#endif
```

```
diff --git a/net/unix/af_unix.c b/net/unix/af_unix.c
index 8015a03..3f57cb2 100644
```

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

```

+++ b/net/unix/af_unix.c
@@ -118,7 +118,7 @@
#include <linux/security.h>
#include <net/net_namespace.h>

-int sysctl_unix_max_dgram_qlen __read_mostly = 10;
+DEFINE_PER_NET(int, sysctl_unix_max_dgram_qlen) = 10;

struct hlist_head unix_socket_table[UNIX_HASH_SIZE + 1];
DEFINE_SPINLOCK(unix_table_lock);
@@ -245,7 +245,8 @@ 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(net_t net,
+      struct sockaddr_un *sunname,
+      int len, int type, unsigned hash)
{
    struct sock *s;
@@ -254,6 +255,9 @@ 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 (!net_eq(s->sk_net, net))
+            continue;
+
        if (u->addr->len == len &&
            !memcmp(u->addr->name, sunname, len))
            goto found;
@@ -263,21 +267,22 @@ found:
        return s;
    }

-static inline struct sock *unix_find_socket_byname(struct sockaddr_un *sunname,
+static inline struct sock *unix_find_socket_byname(net_t 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(net_t net, struct inode *i)
{
    struct sock *s;
    struct hlist_node *node;
@@ -287,6 +292,9 @@ 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 (!net_eq(s->sk_net, net))
+ continue;
+
    if(dentry && dentry->d_inode == i)
    {
        sock_hold(s);
@@ -588,7 +596,7 @@ static struct sock * unix_create1(net_t net, struct socket *sock)
    &af_unix_sk_receive_queue_lock_key);

    sk->sk_write_space = unix_write_space;
- sk->sk_max_ack_backlog = sysctl_unix_max_dgram_qlen;
+ sk->sk_max_ack_backlog = per_net(sysctl_unix_max_dgram_qlen, net);
    sk->sk_destruct = unix_sock_destructor;
    u = unix_sk(sk);
    u->dentry = NULL;
@@ -604,9 +612,6 @@ out:

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

@@ -650,6 +655,7 @@ static int unix_release(struct socket *sock)
static int unix_autobind(struct socket *sock)
{
    struct sock *sk = sock->sk;
+ net_t net = sk->sk_net;
    struct unix_sock *u = unix_sk(sk);
    static u32 ordernum = 1;
    struct unix_address * addr;
@@ -676,7 +682,7 @@ retry:

```

```

spin_lock(&unix_table_lock);
ordernum = (ordernum+1)&0xFFFF;

- 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... */
@@ -696,7 +702,8 @@ 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(net_t net,
+    struct sockaddr_un *sunname, int len,
    int type, unsigned hash, int *error)
{
    struct sock *u;
@@ -714,7 +721,7 @@ 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;

@@ -730,7 +737,7 @@ 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;
@@ -752,6 +759,7 @@ fail:
static int unix_bind(struct socket *sock, struct sockaddr *uaddr, int addr_len)
{
    struct sock *sk = sock->sk;
+ net_t net = sk->sk_net;
    struct unix_sock *u = unix_sk(sk);
    struct sockaddr_un *sunaddr=(struct sockaddr_un *)uaddr;
    struct dentry *dentry = NULL;
@@ -826,7 +834,7 @@ 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;
@@ -867,6 +875,7 @@ static int unix_dgram_connect(struct socket *sock, struct sockaddr *addr,
    int alen, int flags)
{
    struct sock *sk = sock->sk;
+ net_t net = sk->sk_net;
    struct sockaddr_un *sunaddr=(struct sockaddr_un*)addr;
    struct sock *other;
    unsigned hash;
@@ -882,7 +891,7 @@ static int unix_dgram_connect(struct socket *sock, struct sockaddr *addr,
    !unix_sk(sk)->addr && (err = unix_autobind(sock)) != 0)
    goto out;

- 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;

@@ -955,6 +964,7 @@ static int unix_stream_connect(struct socket *sock, struct sockaddr
*uaddr,
{
    struct sockaddr_un *sunaddr=(struct sockaddr_un *)uaddr;
    struct sock *sk = sock->sk;
+ net_t net = sk->sk_net;
    struct unix_sock *u = unix_sk(sk), *newu, *otheru;
    struct sock *newsk = NULL;
    struct sock *other = NULL;
@@ -994,7 +1004,7 @@ 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;

@@ -1273,6 +1283,7 @@ static int unix_dgram_sendmsg(struct kiocb *kiocb, struct socket
*sock,
{
    struct sock_iocb *siocb = kiocb_to_siocb(kiocb);
    struct sock *sk = sock->sk;
+ net_t net = sk->sk_net;

```

```

struct unix_sock *u = unix_sk(sk);
struct sockaddr_un *sunaddr=msg->msg_name;
struct sock *other = NULL;
@@ -1336,7 +1347,7 @@ 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;
@@ -1935,12 +1946,18 @@ 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 {
+ net_t 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 (!net_eq(s->sk_net, iter->net))
+ continue;
    if (off == pos)
        return s;
    ++off;
@@ -1951,17 +1968,24 @@ 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 && !net_eq(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)
@@ -2025,7 +2049,7 @@ 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;
@@ -2036,7 +2060,8 @@ static int unix_seq_open(struct inode *inode, struct file *file)

    seq = file->private_data;
    seq->private = iter;
- *iter = 0;
+ iter->net = get_net(PROC_NET(inode));
+ iter->i = 0;
    out:
    return rc;
    out_kfree:
@@ -2044,12 +2069,20 @@ 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 struct file_operations unix_seq_fops = {
    .owner = THIS_MODULE,
    .open = unix_seq_open,

```

```

.read = seq_read,
.llseek = seq_llseek,
- .release = seq_release_private,
+ .release = unix_seq_release,
};

#endif

@@ -2060,6 +2093,31 @@ static struct net_proto_family unix_family_ops = {
    .owner = THIS_MODULE,
};

+
+static int unix_net_init(net_t net)
+{
+ int error = -ENOMEM;
+ #ifdef CONFIG_PROC_FS
+ if (!proc_net_fops_create(net, "unix", 0, &unix_seq_fops))
+ goto out;
+ #endif
+ unix_sysctl_register(net);
+ error = 0;
+ out:
+ return 0;
+}
+
+static void unix_net_exit(net_t net)
+{
+ unix_sysctl_unregister(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;
@@ -2075,10 +2133,7 @@ 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
- unix_sysctl_register();
+ register_pernet_subsys(&unix_net_ops);

```

```

out:
    return rc;
}
@@ -2086,9 +2141,8 @@ out:
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);
diff --git a/net/unix/sysctl_net_unix.c b/net/unix/sysctl_net_unix.c
index eb0bd57..4b59da8 100644
--- a/net/unix/sysctl_net_unix.c
+++ b/net/unix/sysctl_net_unix.c
@@ -14,11 +14,11 @@
#include <net/af_unix.h>

-static ctl_table unix_table[] = {
+static DEFINE_PER_NET(ctl_table, unix_table[]) = {
{
    .ctl_name = NET_UNIX_MAX_DGRAM_QLEN,
    .procname = "max_dgram_qlen",
- .data = &sysctl_unix_max_dgram_qlen,
+ .data = &__per_net_base(sysctl_unix_max_dgram_qlen),
    .maxlen = sizeof(int),
    .mode = 0644,
    .proc_handler = &proc_dointvec
@@ -26,35 +26,37 @@ static ctl_table unix_table[] = {
{ .ctl_name = 0 }
};

-static ctl_table unix_net_table[] = {
+static DEFINE_PER_NET(ctl_table, unix_net_table[]) = {
{
    .ctl_name = NET_UNIX,
    .procname = "unix",
    .mode = 0555,
- .child = unix_table
+ .child = __per_net_base(unix_table)
},
{ .ctl_name = 0 }
};

```

```

-static ctl_table unix_root_table[] = {
+static DEFINE_PER_NET(ctl_table, unix_root_table[]) = {
    {
        .ctl_name = CTL_NET,
        .procname = "net",
        .mode = 0555,
-    .child = unix_net_table
+    .child = __per_net_base(unix_net_table)
    },
    { .ctl_name = 0 }
};

-static struct ctl_table_header * unix_sysctl_header;
+static DEFINE_PER_NET(struct ctl_table_header *, unix_sysctl_header);

-void unix_sysctl_register(void)
+void unix_sysctl_register(net_t net)
{
-    unix_sysctl_header = register_sysctl_table(unix_root_table);
+    ctl_table *table = per_net(unix_root_table, net);
+    per_net(unix_sysctl_header, net) =
+    register_net_sysctl_table(net, table);
}

-void unix_sysctl_unregister(void)
+void unix_sysctl_unregister(net_t net)
{
-    unregister_sysctl_table(unix_sysctl_header);
+    unregister_net_sysctl_table(per_net(unix_sysctl_header, net));
}

--
1.4.4.1.g278f

```

---

Containers mailing list  
Containers@lists.osdl.org  
<https://lists.osdl.org/mailman/listinfo/containers>

---