Subject: [RFC][patch 3/3] activate filtering for the bind
Posted by Daniel Lezcano on Tue, 04 Sep 2007 17:00:25 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>

For the moment, I only made this patch for the RFC. It shows how simple it is
to hook different socket syscalls. This patch denies bind to any addresses
which are not in the container IPV4 address list, except for the INADDR_ANY.

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>

---
 kernel/container_network.c | 66 ++++++++++++++++++++++----------------------
 1 file changed, 35 insertions(+), 31 deletions(-)

Index: 2.6-mm/kernel/container_network.c
===
--- 2.6-mm.orig/kernel/container_network.c
+++ 2.6-mm/kernel/container_network.c
@@ -12,6 +12,9 @@
 #include <linux/list.h>
 #include <linux/spinlock.h>
 #include <linux/security.h>
+#include <linux/in.h>
+#include <linux/net.h>
+#include <linux/socket.h>

 struct network {
  struct container_subsys_state css;
@@ -53,24 +56,14 @@

 static int network_socket_create(int family, int type, int protocol, int kern)
 {
- struct network *network;
-
- network = task_network(current);
- if (!network || network == &top_network)
- return 0;
-
+ /* nothing to do right now */
  return 0;
 }

 static int network_socket_post_create(struct socket *sock, int family,
       int type, int protocol, int kern)
 {
- struct network *network;

```
-
- network = task_network(current);
- if (!network || network == &top_network)
-  return 0;
-
+ /* nothing to do right now */
  return 0;
 }

@@ -79,47 +72,58 @@
          int addrlen)
 {
  struct network *network;
+ struct list_head *l;
+ rwlock_t *lock;
+ struct ipv4_list *entry;
+ __be32 addr;
+ int ret = -EPERM;

+ /* Do nothing for the root container */
  network = task_network(current);
  if (!network || network == &top_network)
   return 0;

- return 0;
+ /* Check we have to do some filtering */
+ if (sock->ops->family != AF_INET)
+  return 0;
+
+ l = &network->ipv4_list;
+ lock = &network->ipv4_list_lock;
+ addr = ((struct sockaddr_in *)address)->sin_addr.s_addr;
+
+ if (addr == INADDR_ANY)
+  return 0;
+
+ read_lock(lock);
+ list_for_each_entry(entry, l, list) {
+  if (entry->address != addr)
+   continue;
+  ret = 0;
+  break;
+ }
+ read_unlock(lock);
+
+ return ret;
 }
```

```
 static int network_socket_connect(struct socket * sock,
       struct sockaddr * address,
       int addrlen)
 {
- struct network *network;
-
- network = task_network(current);
- if (!network || network == &top_network)
-  return 0;
-
+ /* nothing to do right now */
  return 0;
 }

 static int network_socket_listen(struct socket * sock, int backlog)
 {
- struct network *network;
-
- network = task_network(current);
- if (!network || network == &top_network)
-  return 0;
-
+ /* nothing to do right now */
  return 0;
 }

 static int network_socket_accept(struct socket *sock,
      struct socket *newsock)
 {
- struct network *network;
-
- network = task_network(current);
- if (!network || network == &top_network)
-  return 0;
-
+ /* nothing to do right now */
  return 0;
 }


--
```