Subject: [PATCH 11/12] L2 network namespace: sockets proc view virtualization
Posted by Mishin Dmitry on Wed, 06 Dec 2006 22:30:00 GMT
View Forum Message <> Reply to Message

Only current net namespace sockets or all sockets in case of init_net_ns should
 be visible through proc interface.

 Signed-off-by: Dmitry Mishin <dim@openvz.org>

```
---
 include/net/af_unix.h |   21 ++++++++++++++++++----
 net/ipv4/tcp_ipv4.c   |    9 +++++++++
 net/ipv4/udp.c        |   13 ++++++++++--
 3 files changed, 37 insertions(+), 6 deletions(-)

--- linux-2.6.19-rc6-mm2.orig/include/net/af_unix.h
+++ linux-2.6.19-rc6-mm2/include/net/af_unix.h
@@ -19,9 +19,13 @@ extern atomic_t unix_tot_inflight;

 static inline struct sock *first_unix_socket(int *i)
 {
+ struct sock *sk;
+
  for (*i = 0; *i <= UNIX_HASH_SIZE; (*i)++) {
-  if (!hlist_empty(&unix_socket_table[*i]))
-   return __sk_head(&unix_socket_table[*i]);
+  for (sk = sk_head(&unix_socket_table[*i]); sk; sk = sk_next(sk))
+   if (net_ns_match(sk->sk_net_ns, current_net_ns) ||
+    net_ns_match(current_net_ns, &init_net_ns))
+    return sk;
  }
  return NULL;
 }
@@ -32,10 +36,19 @@ static inline struct sock *next_unix_soc
  /* More in this chain? */
  if (next)
   return next;
+ for (; next != NULL; next = sk_next(next)) {
+  if (!net_ns_match(next->sk_net_ns, current_net_ns) &&
+   !net_ns_match(current_net_ns, &init_net_ns))
+   continue;
+  return next;
+ }
  /* Look for next non-empty chain. */
  for ((*i)++; *i <= UNIX_HASH_SIZE; (*i)++) {
-  if (!hlist_empty(&unix_socket_table[*i]))
-   return __sk_head(&unix_socket_table[*i]);
+  for (next = sk_head(&unix_socket_table[*i]); next;
```

```
+       next = sk_next(next))
+   if (net_ns_match(next->sk_net_ns, current_net_ns) ||
+    net_ns_match(current_net_ns, &init_net_ns))
+    return next;
  }
  return NULL;
 }
--- linux-2.6.19-rc6-mm2.orig/net/ipv4/tcp_ipv4.c
+++ linux-2.6.19-rc6-mm2/net/ipv4/tcp_ipv4.c
@@ -2032,6 +2032,9 @@ get_req:
  }
 get_sk:
  sk_for_each_from(sk, node) {
+  if (!net_ns_match(sk->sk_net_ns, current_net_ns) &&
+   !net_ns_match(current_net_ns, &init_net_ns))
+   continue;
   if (sk->sk_family == st->family) {
    cur = sk;
    goto out;
@@ -2083,6 +2086,9 @@ static void *established_get_first(struc

   read_lock(&tcp_hashinfo.ehash[st->bucket].lock);
   sk_for_each(sk, node, &tcp_hashinfo.ehash[st->bucket].chain) {
+   if (!net_ns_match(sk->sk_net_ns, current_net_ns) &&
+    !net_ns_match(current_net_ns, &init_net_ns))
+    continue;
    if (sk->sk_family != st->family) {
     continue;
    }
@@ -2142,6 +2148,9 @@ get_tw:
   sk = sk_next(sk);

  sk_for_each_from(sk, node) {
+  if (!net_ns_match(sk->sk_net_ns, current_net_ns) &&
+   !net_ns_match(current_net_ns, &init_net_ns))
+   continue;
   if (sk->sk_family == st->family)
    goto found;
  }
--- linux-2.6.19-rc6-mm2.orig/net/ipv4/udp.c
+++ linux-2.6.19-rc6-mm2/net/ipv4/udp.c
@@ -1546,6 +1546,9 @@ static struct sock *udp_get_first(struct
  for (state->bucket = 0; state->bucket < UDP_HTABLE_SIZE; ++state->bucket) {
   struct hlist_node *node;
   sk_for_each(sk, node, state->hashtable + state->bucket) {
+   if (!net_ns_match(sk->sk_net_ns, current_net_ns) &&
+    !net_ns_match(current_net_ns, &init_net_ns))
+    continue;
```

```
    if (sk->sk_family == state->family)
     goto found;
   }
@@ -1562,8 +1565,14 @@ static struct sock *udp_get_next(struct
  do {
   sk = sk_next(sk);
 try_again:
-  ;
- } while (sk && sk->sk_family != state->family);
+  if (!sk)
+   break;
+  if (sk->sk_family != state->family)
+   continue;
+  if (net_ns_match(sk->sk_net_ns, current_net_ns) ||
+   net_ns_match(current_net_ns, &init_net_ns))
+   break;
+ } while (1);

  if (!sk && ++state->bucket < UDP_HTABLE_SIZE) {
   sk = sk_head(state->hashtable + state->bucket);
```

_____

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