Subject: openvz + ipv6
Posted by Romain Riviere on Wed, 22 Feb 2006 00:26:41 GMT
View Forum Message <> Reply to Message

Hello,

Follow-up to the conversation I've had on #openvz...

I had been meaning to try OpenVZ (stable 2.6.8 022) until tonight. But I
also use IPv6. As it turned out, the openvz patch makes ipv6 unusable.
The fact that it contains modifications on functions used in both ipv4
and ipv6 networking makes it impossible to even compile ipv6 support.

It would be nice if somehow ipv6 could at least remain usable on the
host system !

Cheerio !
Romain (aka Mr_Smoke on FreeNode)

Subject: Re: openvz + ipv6
Posted by dev on Wed, 22 Feb 2006 13:09:16 GMT
View Forum Message <> Reply to Message

> Follow-up to the conversation I've had on #openvz...
>
> I had been meaning to try OpenVZ (stable 2.6.8 022) until tonight. But I
> also use IPv6. As it turned out, the openvz patch makes ipv6 unusable.
> The fact that it contains modifications on functions used in both ipv4
> and ipv6 networking makes it impossible to even compile ipv6 support.
>
> It would be nice if somehow ipv6 could at least remain usable on the
> host system !
Can you send me your config? and I will send you a patch to fix it then.

Kirill

Subject: Re: openvz + ipv6
Posted by dev on Wed, 22 Feb 2006 18:11:56 GMT
View Forum Message <> Reply to Message

Hello,

> I had been meaning to try OpenVZ (stable 2.6.8 022) until tonight. But I
> also use IPv6. As it turned out, the openvz patch makes ipv6 unusable.
> The fact that it contains modifications on functions used in both ipv4

> and ipv6 networking makes it impossible to even compile ipv6 support.
yup, it was fixed in 2.6.15 already.
I backported that patch set and attached it for you. It compiles, but I
have not tested whether it works, sorry.

> It would be nice if somehow ipv6 could at least remain usable on the
> host system !
Agreed, it should be usable :)
But it requires some isolation efforts which prevent VPS from using it... :(

Thanks,
Kirill

```
--- ./include/linux/socket.h.xxx 2006-02-22 16:28:38.000000000 +0300
+++ ./include/linux/socket.h 2006-02-22 17:37:57.000000000 +0300
@@ -291,6 +291,7 @@ extern void memcpy_tokerneliovec(struct
 extern int move_addr_to_user(void *kaddr, int klen, void __user *uaddr, int __user *ulen);
 extern int move_addr_to_kernel(void __user *uaddr, int ulen, void *kaddr);
 extern int put_cmsg(struct msghdr*, int level, int type, int len, void *data);
+extern int vz_security_proto_check(int family, int type, int protocol);

 #endif
 #endif /* not kernel and not glibc */
--- ./kernel/vecalls.c.xxx 2006-02-22 16:28:51.000000000 +0300
+++ ./kernel/vecalls.c 2006-02-22 17:43:11.000000000 +0300
@@ -2520,14 +2520,6 @@ static int ve_netdev_cleanup(struct net_
    err = rc;
  }

-#if (defined(CONFIG_ATALK) || defined(CONFIG_ATALK_MODULE) ||  \
-     defined(CONFIG_DECNET) || defined(CONFIG_DECNET_MODULE) ||  \
-     defined(CONFIG_IPV6) || defined(CONFIG_IPV6_MODULE) ||  \
-     defined(CONFIG_ECONET) || defined(CONFIG_ECONET_MODULE) ||  \
-     defined(CONFIG_NET_FASTROUTE))
-#error "AppleTalk, DECnet, IPv6, Econet or FASTROUTE is (are) unsupported"
-#endif
-
  return err;
 }

--- ./net/core/rtnetlink.c.xxx 2006-02-22 16:28:45.000000000 +0300
+++ ./net/core/rtnetlink.c 2006-02-22 17:37:57.000000000 +0300
@@ -294,6 +294,8 @@ static int rtnetlink_dump_all(struct sk_
   if (rtnetlink_links[idx] == NULL ||
       rtnetlink_links[idx][type].dumpit == NULL)
    continue;
+  if (vz_security_proto_check(idx, 0, 0))
+   continue;
```

```
   if (idx > s_idx)
    memset(&cb->args[0], 0, sizeof(cb->args));
   if (rtnetlink_links[idx][type].dumpit(skb, cb))
@@ -362,7 +364,7 @@ rtnetlink_rcv_msg(struct sk_buff *skb, s
   return 0;

  family = ((struct rtgenmsg*)NLMSG_DATA(nlh))->rtgen_family;
- if (family >= NPROTO) {
+ if (family >= NPROTO || vz_security_proto_check(family, 0, 0)) {
   *errp = -EAFNOSUPPORT;
   return -1;
  }
--- ./net/ipv6/addrconf.c.xxx 2006-02-22 17:40:13.000000000 +0300
+++ ./net/ipv6/addrconf.c 2006-02-22 17:40:27.000000000 +0300
@@ -1875,6 +1875,10 @@ static int addrconf_notify(struct notifi
  struct net_device *dev = (struct net_device *) data;
  struct inet6_dev *idev = __in6_dev_get(dev);

+ /* not virtualized yet */
+ if (!ve_is_super(get_exec_env()))
+  return NOTIFY_OK;
+
  switch(event) {
  case NETDEV_UP:
   switch(dev->type) {
--- ./net/ipv6/tcp_ipv6.c.xxx 2006-02-22 16:28:42.000000000 +0300
+++ ./net/ipv6/tcp_ipv6.c 2006-02-22 21:07:00.000000000 +0300
@@ -142,7 +142,7 @@ static int tcp_v6_get_port(struct sock *
   do { rover++;
    if ((rover < low) || (rover > high))
     rover = low;
-   head = &tcp_bhash[tcp_bhashfn(rover)];
+   head = &tcp_bhash[tcp_bhashfn(rover, 0)];
    spin_lock(&head->lock);
    tb_for_each(tb, node, &head->chain)
     if (tb->port == rover)
@@ -162,7 +162,7 @@ static int tcp_v6_get_port(struct sock *
   /* OK, here is the one we will use. */
   snum = rover;
  } else {
-  head = &tcp_bhash[tcp_bhashfn(snum)];
+  head = &tcp_bhash[tcp_bhashfn(snum, 0)];
   spin_lock(&head->lock);
   tb_for_each(tb, node, &head->chain)
    if (tb->port == snum)
@@ -183,7 +183,7 @@ tb_found:
  }
 tb_not_found:
```

```
  ret = 1;
- if (!tb && (tb = tcp_bucket_create(head, snum)) == NULL)
+ if (!tb && (tb = tcp_bucket_create(head, snum, NULL)) == NULL)
    goto fail_unlock;
  if (hlist_empty(&tb->owners)) {
    if (sk->sk_reuse && sk->sk_state != TCP_LISTEN)
@@ -255,7 +255,7 @@ static struct sock *tcp_v6_lookup_listen

  hiscore=0;
  read_lock(&tcp_lhash_lock);
- sk_for_each(sk, node, &tcp_listening_hash[tcp_lhashfn(hnum)]) {
+ sk_for_each(sk, node, &tcp_listening_hash[tcp_lhashfn(hnum, 0)]) {
    if (inet_sk(sk)->num == hnum && sk->sk_family == PF_INET6) {
     struct ipv6_pinfo *np = inet6_sk(sk);

@@ -522,7 +522,7 @@ static int tcp_v6_hash_connect(struct so
    inet_sk(sk)->sport = htons(inet_sk(sk)->num);
  }

- head = &tcp_bhash[tcp_bhashfn(inet_sk(sk)->num)];
+ head = &tcp_bhash[tcp_bhashfn(inet_sk(sk)->num, 0)];
  tb = tb_head(head);

  spin_lock_bh(&head->lock);
--- ./net/ipv6/udp.c.xxx 2006-02-22 17:37:02.000000000 +0300
+++ ./net/ipv6/udp.c 2006-02-22 17:37:11.000000000 +0300
@@ -67,7 +67,9 @@ static int udp_v6_get_port(struct sock *
 {
  struct sock *sk2;
  struct hlist_node *node;
+ struct ve_struct *env;

+ env = VE_OWNER_SK(sk);
  write_lock_bh(&udp_hash_lock);
  if (snum == 0) {
    int best_size_so_far, best, result, i;
@@ -81,7 +83,7 @@ static int udp_v6_get_port(struct sock *
    int size;
    struct hlist_head *list;

-   list = &udp_hash[result & (UDP_HTABLE_SIZE - 1)];
+   list = &udp_hash[udp_hashfn(result, VEID(env))];
    if (hlist_empty(list)) {
     if (result > sysctl_local_port_range[1])
      result = sysctl_local_port_range[0] +
@@ -103,16 +105,17 @@ static int udp_v6_get_port(struct sock *
     result = sysctl_local_port_range[0]
      + ((result - sysctl_local_port_range[0]) &
```

```
         (UDP_HTABLE_SIZE - 1));
-   if (!udp_lport_inuse(result))
+    if (!udp_lport_inuse(result, env))
      break;
    }
  gotit:
    udp_port_rover = snum = result;
   } else {
    sk_for_each(sk2, node,
-       &udp_hash[snum & (UDP_HTABLE_SIZE - 1)]) {
+        &udp_hash[udp_hashfn(snum, VEID(env))]) {
     if (inet_sk(sk2)->num == snum &&
         sk2 != sk &&
+        ve_accessible_strict(VE_OWNER_SK(sk2), env) &&
         (!sk2->sk_bound_dev_if ||
          !sk->sk_bound_dev_if ||
         sk2->sk_bound_dev_if == sk->sk_bound_dev_if) &&
@@ -124,7 +127,7 @@ gotit:

   inet_sk(sk)->num = snum;
   if (sk_unhashed(sk)) {
-  sk_add_node(sk, &udp_hash[snum & (UDP_HTABLE_SIZE - 1)]);
+  sk_add_node(sk, &udp_hash[udp_hashfn(snum, VEID(env))]);
    sock_prot_inc_use(sk->sk_prot);
   }
   write_unlock_bh(&udp_hash_lock);
--- ./net/socket.c.xxx 2006-02-22 16:28:40.000000000 +0300
+++ ./net/socket.c 2006-02-22 17:39:38.000000000 +0300
@@ -81,6 +81,7 @@
  #include <linux/syscalls.h>
  #include <linux/compat.h>
  #include <linux/kmod.h>
+#include <linux/in.h>

  #ifdef CONFIG_NET_RADIO
  #include <linux/wireless.h>  /* Note : will define WIRELESS_EXT */
@@ -1071,6 +1072,37 @@ int sock_wake_async(struct socket *sock,
   return 0;
  }

+int vz_security_proto_check(int family, int type, int protocol)
+{
+#ifdef CONFIG_VE
+ if (ve_is_super(get_exec_env()))
+  return 0;
+
+ switch (family) {
+ case PF_UNSPEC:
```

```
+ case PF_PACKET:
+ case PF_NETLINK:
+ case PF_UNIX:
+  break;
+ case PF_INET:
+  switch (protocol) {
+  case  IPPROTO_IP:
+  case  IPPROTO_ICMP:
+  case  IPPROTO_TCP:
+  case  IPPROTO_UDP:
+  case  IPPROTO_RAW:
+   break;
+  default:
+   return -EAFNOSUPPORT;
+  }
+  break;
+ default:
+  return -EAFNOSUPPORT;
+ }
+#endif
+ return 0;
+}
+
 static int __sock_create(int family, int type, int protocol, struct socket **res, int kern)
 {
  int i;
@@ -1099,6 +1131,11 @@ static int __sock_create(int family, int
   family = PF_PACKET;
  }

+ /* VZ compatibility layer */
+ err = vz_security_proto_check(family, type, protocol);
+ if (err < 0)
+  return err;
+
  err = security_socket_create(family, type, protocol, kern);
  if (err)
   return err;
```