Subject: Re: [RFC][patch 3/3] activate filtering for the bind
Posted by serue on Mon, 10 Sep 2007 13:23:30 GMT
View Forum Message <> Reply to Message

Quoting Daniel Lezcano (dlezcano@meiosys.com):
> Serge E. Hallyn wrote:
>> Quoting dlezcano@fr.ibm.com (dlezcano@fr.ibm.com):
>>> 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 @@
>> Please so send -p diffs. I'll assume this is network_socket_bind()
>> given your patch description :)
>>> 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)
>> In bsdjail, if addr == INADDR_ANY, I set addr = jailaddr. Do you think
>> you want to do that?
>
> Good question. This is one think I would like to define. If we do that we

> can not connect via 127.0.0.1. and|or a container can have more than one IP
> address, no ?

Yes.

> IMHO, we should have the loopback address available for all containers and
> that means 127.0.0.1 is an IP address which is not isolated.

For real network namespaces yes. For this version, I would have thought
the goal would be to provide a minimal, useful, but very fast
container-paddress binding.

I guess I'll have to see the rest of your implementation, but I have the
feeling that to not have this limitation you'll affect performance a
bit. And since we are also working on full network namespaces,
providing maximal functionality with worse performance would be a poor
tradeoff here.

But let's see the rest of your implementation.

Did you mention somewhere that Eric still prefers using netfilter rather
than LSM? So what... if a packet comes in with a certain destination
address you can tag it with a container, and once a connection starts
you can use connection tracking to continue tagging it with that
container. You tag an outgoing packet with the container as soon as
it's dumped in the socket, and rules enforce that the source address be
valid for that container. Are you saying the netfilter hooks are in
the wrong places for that?

-serge

> If we choose to deny access to 127.0.0.1, then there will be some issues
> with the routing. If we connect to 127.0.0.1 (this address belongs to the
> root container) from a child container, the source address should be filled
> with an IP address belonging to a container (eg 10.0.0.10), so we have
> (src)10.0.0.1 -> (dst)127.0.0.1, that means the root container will answer
> to 10.0.0.1 and use this address. This is no sense because routing should
> be for the loopback: 127.0.0.1<->127.0.0.1, and we break isolation. Tricky.
>
>>> + 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;
>>>  }
>>>
>>>
>>> --
>>> _____________________________________________________
>>> Containers mailing list
```

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

_____

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