Subject: Re: [RFC][patch 3/3] activate filtering for the bind
Posted by Daniel Lezcano on Mon, 10 Sep 2007 13:52:17 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn wrote:
> 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?

Paul told me about a ip isolation based on the netfilter and a specific
iptable module:

"
On 9/6/07, Daniel Lezcano <dlezcano@fr.ibm.com> wrote:
 > >
 > > I am really not opposed to iptables, I was thinking that if we want to
 > > have bind filtering, security provides the framework for that and
adding
 > > new hooks for the iptable will just add a hook duplication because they
 > > are the same.
 > >
 > > So the result is:
 > >
 > > 1 - create a container => network.ipv4 (allowed addresses)
 > > 2 - echo add 192.168.20.10 > network.ipv4
 > >
 > > The application running inside the container can not use another
address
 > > than the one assigned to it.
 > >
 > > This features is needed for some IP jailing like linux-vserver or for
 > > security. The association container + IP isolation is really a good
feature.

> >
>> > > For instance, I personally am much more interested in being able to
>> > > control ports rather than IP addresses (although that could be
>> > > interesting too).
> >
> > What do want to do ? Can you describe the features you want ?
> > Is it a bind filtering for port ? If this is the case, then I can add
> > two new files:
> > network.tcp.ports
> > network.udp.ports
> > and extend the hooks to check the port too.
> >

I think that (at least today :-)) my ideal interface would be a list
of tuples of the form:

local port range/remote ip address/remote ip mask/remote port range

because I don't really care about multiple local addresses, but I do
care about binding to local ports and connecting to remote addresses
and ports.

But other people (e.g. Eric) have completely different requirements.
Creating an API and mechanism that satisfies everyone is going to
result in you reimplementing a significant chunk of the iptables
functionality.

> >
>> > > And someone else might have completely different
>> > > needs (e.g. people mentioned IPv6). Rather than you having to
>> > > implement all of these things, just giving a tag that can be tied to
>> > > iptables means that people can define these rules themselves in
>> > > userspace.
> >
> > I understand. But I don't see how we can handle bind filtering (ip
| port).
> >

1) Completely separately from containers, we create a new iptable
called something like "socket", with predefined chains BIND, ACCEPT
and CONNECT. When ever someone tries to do a bind(), accept() or
connect() we create a fake packet with the appropriate local and
remote addresses and ports, and feed it through the appropriate chain.
If it gets though OK we allow the operation to proceed, else we fail
with EPERM.

2) We create a new container (control group) subsystem, e.g. called
"network_id" that does two things:

- creates a simple state object with a single uniquely-generated
integer network_id for each control group

- provides a new iptable match module ("control_group"?) that matches
if the current task's network_id is within a given range.

Then the user can create pretty much arbitrary rules with the existing
iptables tools and primitives. No complex new user APIs needed.

Paul"

> 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?

No, for that netfilters hooks are in the right place.

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