

---

Subject: Re: [RFC][patch 2/3] network security hooks  
Posted by [serue](#) on Wed, 05 Sep 2007 16:04:08 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Quoting dlezcano@fr.ibm.com (dlezcano@fr.ibm.com):

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

>

> After all, the network security hooks are placed exactly at the places we need.

> This patch plugs the network security hooks with network container subsystem.

>

> The hooks always do nothing when they are called from a process running inside  
> the root container.

>

> The security hooks are not activated by default when the root container is created,  
> I let the first child container to do that, that ensure correct kernel initialization.

>

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

>

>

> ---

> kernel/container\_network.c | 148 ++++++  
> 1 file changed, 148 insertions(+)

>

> Index: 2.6-mm/kernel/container\_network.c

> =====

> --- 2.6-mm.orig/kernel/container\_network.c

> +++ 2.6-mm/kernel/container\_network.c

> @@ -11,6 +11,7 @@

> #include <linux/ctype.h>

> #include <linux/list.h>

> #include <linux/spinlock.h>

> +#include <linux/security.h>

>

> struct network {

> struct container\_subsys\_state css;

> @@ -28,12 +29,21 @@

> .ipv4\_list\_lock = \_\_RW\_LOCK\_UNLOCKED(top\_network.ipv4\_list\_lock),

> };

>

> +static int security\_registered;

> +static int secondary;

> +

> struct container\_subsys network\_subsys;

>

> enum container\_filetype {

> FILE\_IPV4,

> };

>

```

> +static inline struct network *task_network(struct task_struct *task)
> +{
> + return container_of(task_subsys_state(task, network_subsys_id),
> +     struct network, css);
> +}
> +
> static inline struct network *container_network(struct container *container)
> {
> return container_of(
> @@ -41,6 +51,125 @@
>     struct network, css);
> }
>
> +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)

```

I'm not sure that's the way to go here is it? Maybe you want to allow a CAP\_NET\_ADMIN process in top\_network to drop a network address so that if it does a bind(INADDR\_ANY) it won't get connections for vserver1?

```

> + return 0;
> +
> + 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;
> +
> + return 0;
> +}
> +
> +static int network_socket_bind(struct socket *sock,
> +     struct sockaddr *address,
> +     int addrlen)
> +{
> + struct network *network;
> +
> + network = task_network(current);

```

```

> + if (!network || network == &top_network)
> + return 0;
> +
> + return 0;
> +}
> +
> +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;
> +
> + 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;
> +
> + 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;
> +
> + return 0;
> +}
> +
> +static void network_socket_post_accept(struct socket *sock,
> +    struct socket *newsock)
> +{
> + struct network *network;
> +
> + network = task_network(current);

```

```

> + if (!network || network == &top_network)
> + return;
> +}
> +
> +static int network_socket_sendmsg(struct socket *sock,
> +    struct msghdr *msg, int size)
> +{
> + struct network *network;
> +
> + network = task_network(current);
> + if (!network || network == &top_network)
> + return 0;
> +
> + return 0;
> +}
> +
> +static int network_socket_recvmsg(struct socket *sock,
> +    struct msghdr *msg, int size,
> +    int flags)
> +{
> + struct network *network;
> +
> + network = task_network(current);
> + if (!network || network == &top_network)
> + return 0;
> +
> + return 0;
> +}
> +
> +static struct security_operations network_security_ops = {
> + .socket_create = network_socket_create,
> + .socket_post_create = network_socket_post_create,
> + .socket_bind = network_socket_bind,
> + .socket_connect = network_socket_connect,
> + .socket_listen = network_socket_listen,
> + .socket_accept = network_socket_accept,
> + .socket_post_accept = network_socket_post_accept,
> + .socket_sendmsg = network_socket_sendmsg,
> + .socket_recvmsg = network_socket_recvmsg,
> +};
> +
> static struct container_subsys_state *network_create(struct container_subsys *ss,
>     struct container *container)
> {
> @@ -61,6 +190,25 @@
> INIT_LIST_HEAD(&network->ipv4_list);
> network->ipv4_list_lock = __RW_LOCK_UNLOCKED(network->ipv4_list_lock);
>

```

```

> + /*
> + * register the network security hooks only one time
> + * after the root container is created, the first non
> + * root container has the assignment to register the
> + * security hooks
> + *
> + */
> + if (!security_registered) {
> +   if (register_security(&network_security_ops)) {
> +     if (mod_reg_security(KBUILD_MODNAME,
> +       &network_security_ops)) {
> +       kfree(network);
> +       return ERR_PTR(-EINVAL);
> +     }
> +     secondary = 1;
> +   }
> +   security_registered = 1;
> + }
> +
>   return &network->css;
> }

```

This is an unusual way to do this... Usually one registers the lsm hooks at module load. Though I can't think of any reason it's particularly bad, other than the register\_security() might have succeeded when you are first loaded and might fail later.

Congratulations, I think you're the first to do this :)

Note, you don't ever unregister yourself, so there is no point in defining 'secondary'.

-serge

---

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

---