

AF_IPN: Inter Process Networking, try these...

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-12/msg02415.html>

- *From:* renzo@xxxxxxxxxxx (Renzo Davoli)
 - *Date:* Fri, 7 Dec 2007 22:18:05 +0100
-

Andi, David,

I disagree. If you suspect we would be better using IP multicast, I think your suspects are not supported.

Try the following exercises, please.... Can you provide better solutions without IPN?

renzo

Exercise #1.

I am a user (NOT ROOT), I like kvm, qemu etc. I want an efficient network between my VM.

My solution:

I Create a IPN socket, with protocol IPN_VDESWITCH and all the VM can communicate.

Your solution:

- I am condemned by two kernel developers to run the switch in the userland
- I beg the sysadm to give me some pre-allocated taps connected together by a kernel bridge.
- I create a multicast socket limited to this host (TTL=0) and I use it like a hub. It cannot switch the packets.

Exercise #2.

I am a sysadm (maybe a lab administrator). I want my users (not root) of the group "vmenabled" to run their VM connected to a network. I have hundreds of users in vmenabled(say students).

My Solution:

I create a IPN socket, with protocol IPN_VDESWITCH, connected to a virtual interface say ipn0. I give to the socket permission 760 owner root:vmenabled.

Your solution:

- I am condemned by two kernel developers to run the switch in the userland
- I create a multicast socket connected to a tap and then I define iptables filters to avoid unauthorized users to join the net.
- I create hundreds of preallocated tap interfaces, at least one per user.

AF_IPN: Inter Process Networking, try these...

Exercise #3.

I am a user (NOT ROOT) and I have a heavy stream of *very private data* generated by some processes that must be received by several processes.

I am looking for an efficient solution.

Data can be ASCII strings, or a binary stream.

It is not a "networking" issue, it is just IPC.

My solution.

I Create a IPN socket with permission 700, IPN_BROADCAST protocol. All the processes connect to the socket either for writing or for reading (or both).

Your solution:

– I am condemned by two kernel developers to use userland inefficient solutions like named pipes, tee, or a user daemon among AF_UNIX sockets.

– If I use multicast, others can read the stream.

(security by obscurity? the attacker do not know the address?)

– I use a multicast socket with SSL (it sounds funny to use encryption to talk with myself, exposing the stream to crypto attack).

—

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>