Privilege dropping one capability at a time

February 6th, 2021, FOSDEM Virt & IaaS Dev Room

Miguel Duarte Barroso

mdbarroso@redhat.com
https://github.com/maiqueb

KubeVirt

Agenda

- Linux capabilities
- KubeVirt architecture
- Motivation, problem, and Goals
- The plan
- Dropping capabilities
- Next steps

Introduction

Linux capabilities

- Traditional UNIX implementations have 2 categories of processes
  - *Privileged* and *unprivileged* processes
- Since Kernel 2.2 privileges were divided in distinct units
  - Independently enabled / disabled
- Per thread attribute
- Simple checks in the kernel code
  - https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/drivers/net/tun.c#n565
  - https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/net/core/sock.c#n567
- Relevant capabilities for KubeVirt:
  - CAP_NET_ADMIN
  - CAP_NET_RAW

Capabilities demo

https://github.com/maiqueb/fosdem2021

Demoing NET_ADMIN

Demoing NET_RAW

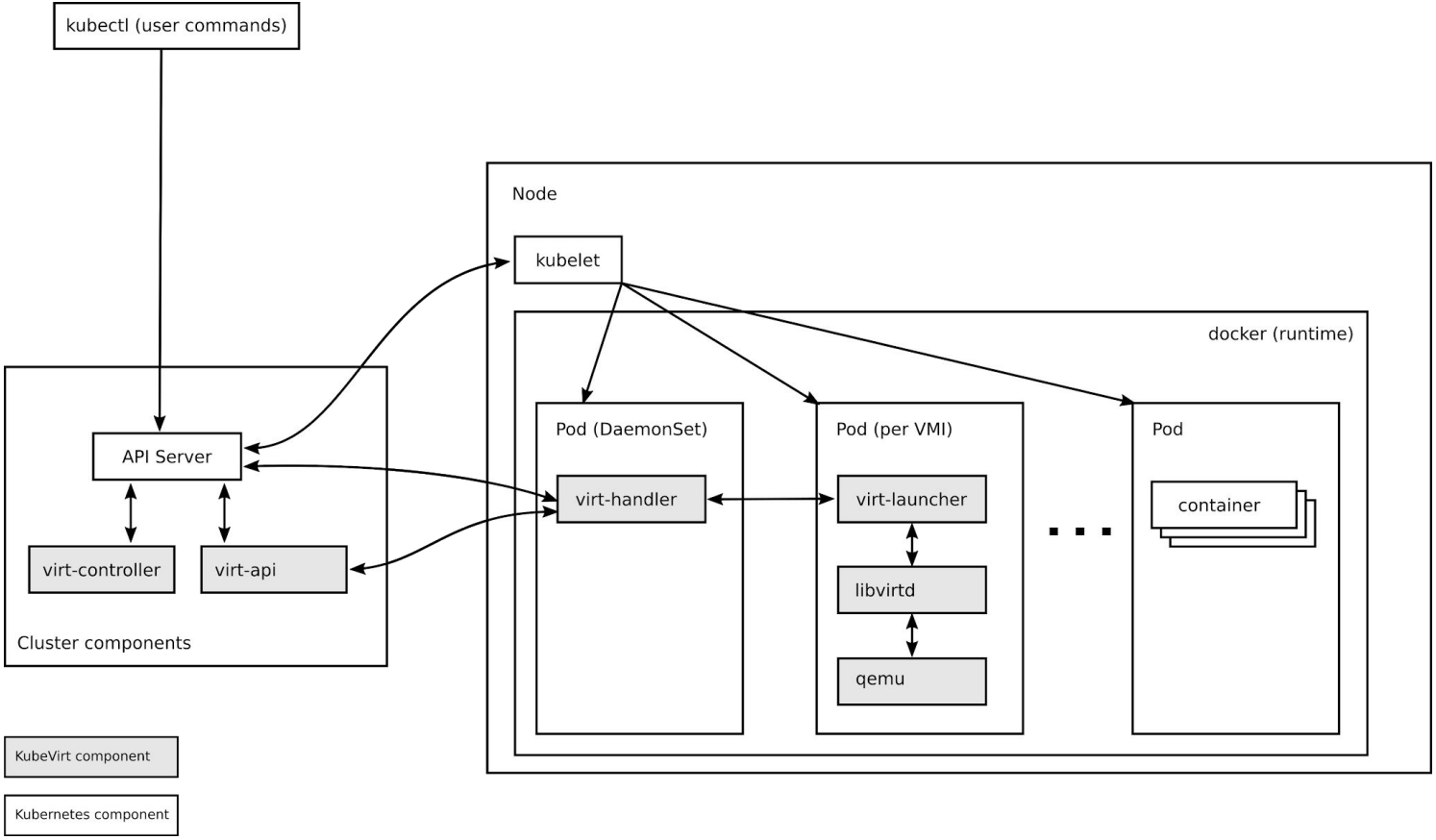
Possible attacks

- NET ADMIN
  - Leak dhcp to the host
- NET RAW
  - ARP / DNS spoofing

DHCP leak

KubeVirt architecture

KubeVirt architecture



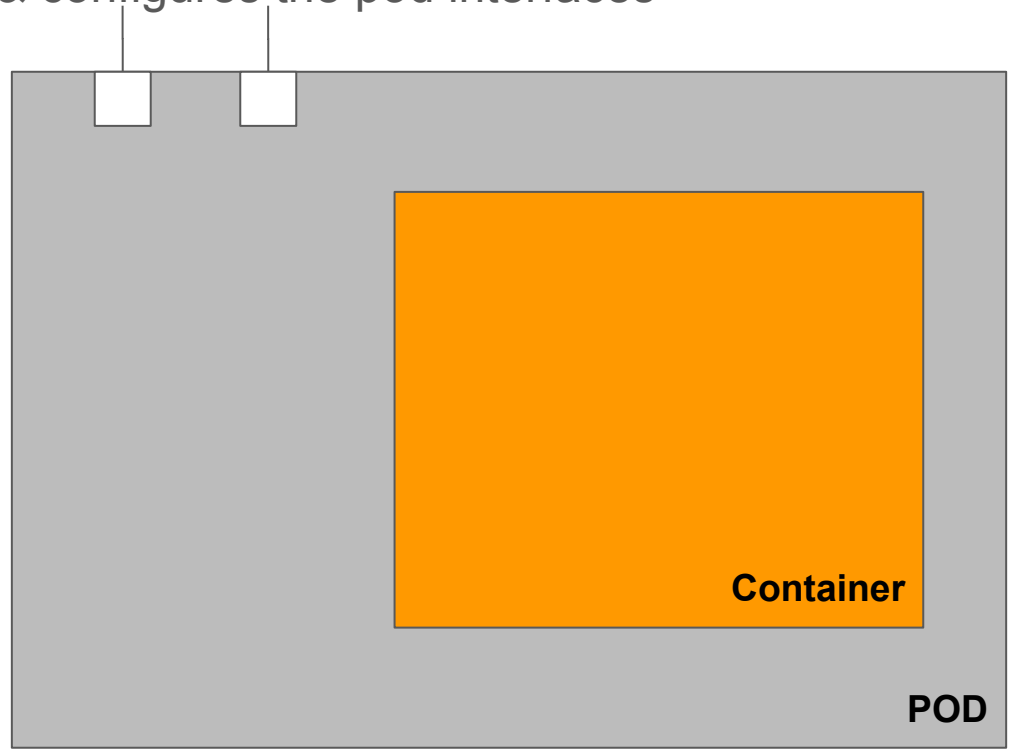
Create VM example

```
 1 ---
 2 apiVersion: kubevirt.io/v1
 3 kind: VirtualMachineInstance
...
 8 spec:
 9   domain:
10     devices:
...
18         interfaces:
19         - masquerade: {}
20           name: default
21         - bridge: {}
22           name: secondary
...
29   networks:
30   - name: default
31     pod: {}
32   - multus:
33         networkName: br10
34     name: secondary
```

Bind Mechanism

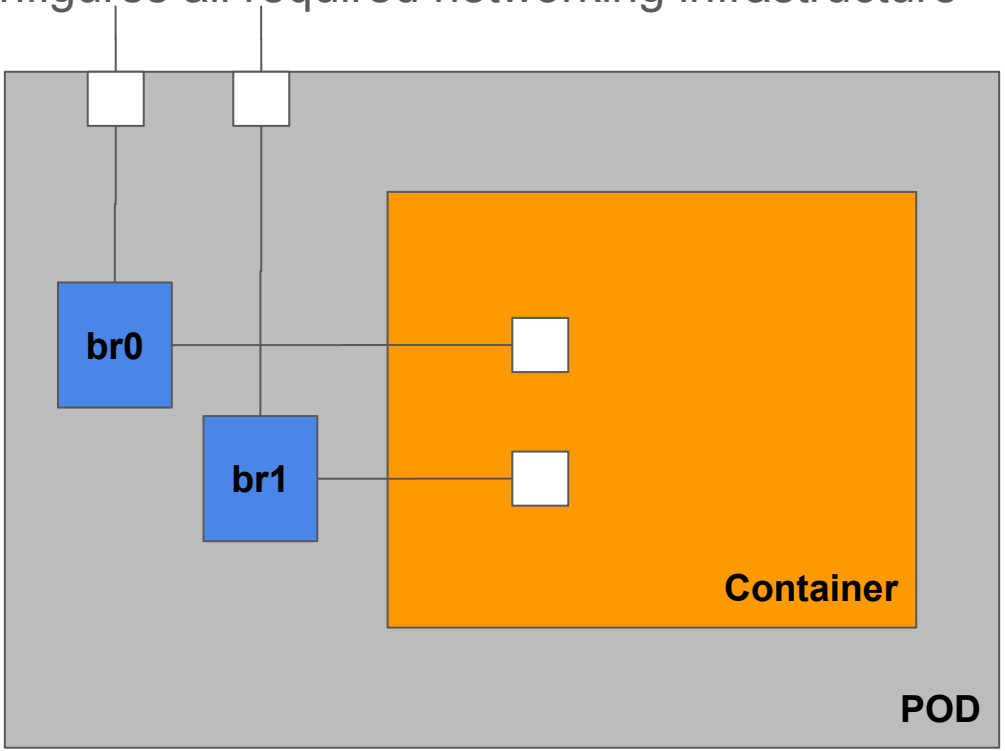
Binding Mechanisms

- CNI creates & configures the pod interfaces



Binding Mechanisms

- Creates & configures all required networking infrastructure



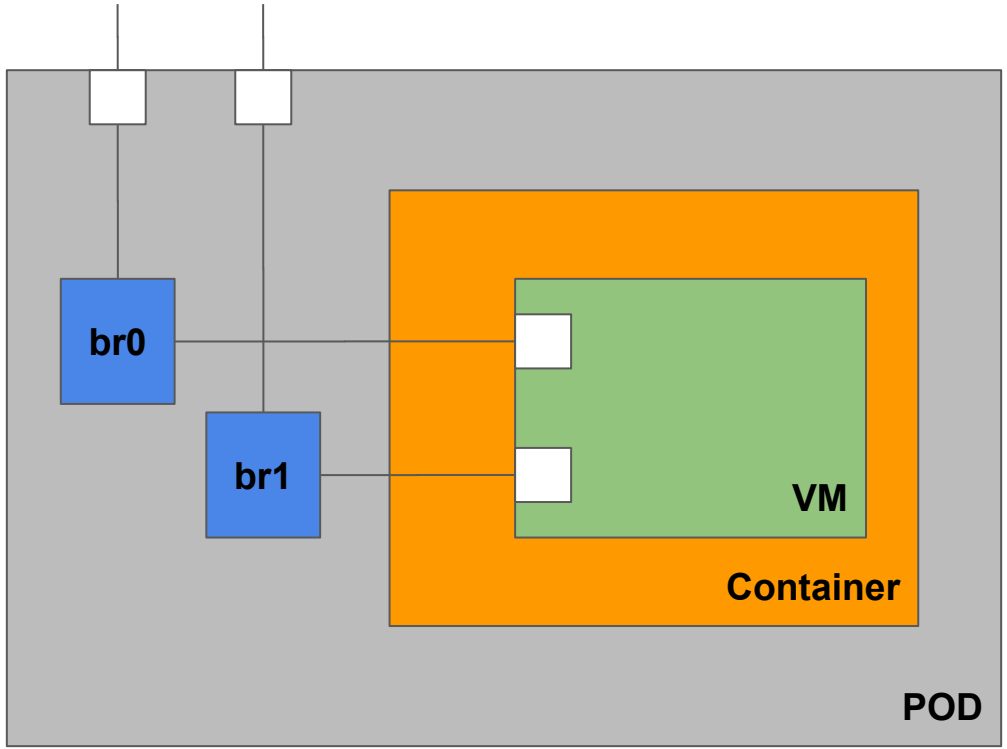
Binding Mechanisms

- Translation between a KubeVirt specification to a Libvirt Dom XML definition

```yaml
---
apiVersion: kubevirt.io/v1
kind: VirtualMachineInstance
...
spec:
  domain:
    devices:
...
      interfaces:
      - masquerade: {}
        name: default
      - bridge: {}
        name: secondary
...
  networks:
  - name: default
    pod: {}
  - multus:
      networkName: br10
    name: secondary
```

```xml
<interface type='ethernet'>
  <mac address='02:00:00:58:9e:f3'/>
  <target dev='tap0' managed='no'/>
  <model type='virtio-non-transitional'/>
  <mtu size='1440'/>
  <alias name='ua-default'/>
  <rom enabled='no'/>
  <address type='pci' domain='0x0000' bus='0x01' slot='0x00' function='0x0'/>
</interface>
<interface type='ethernet'>
  <mac address='c6:af:24:f0:14:fe'/>
  <target dev='tap1' managed='no'/>
  <model type='virtio-non-transitional'/>
  <mtu size='9000'/>
  <alias name='ua-secondary'/>
  <rom enabled='no'/>
  <address type='pci' domain='0x0000' bus='0x02' slot='0x00' function='0x0'/>
</interface>
```

Binding Mechanisms



Architecture take-aways

- Two components
  - Trusted
  - Untrusted
- Two stages of VM networking configuration
  - Features privileged operations
  - Only features unprivileged operations

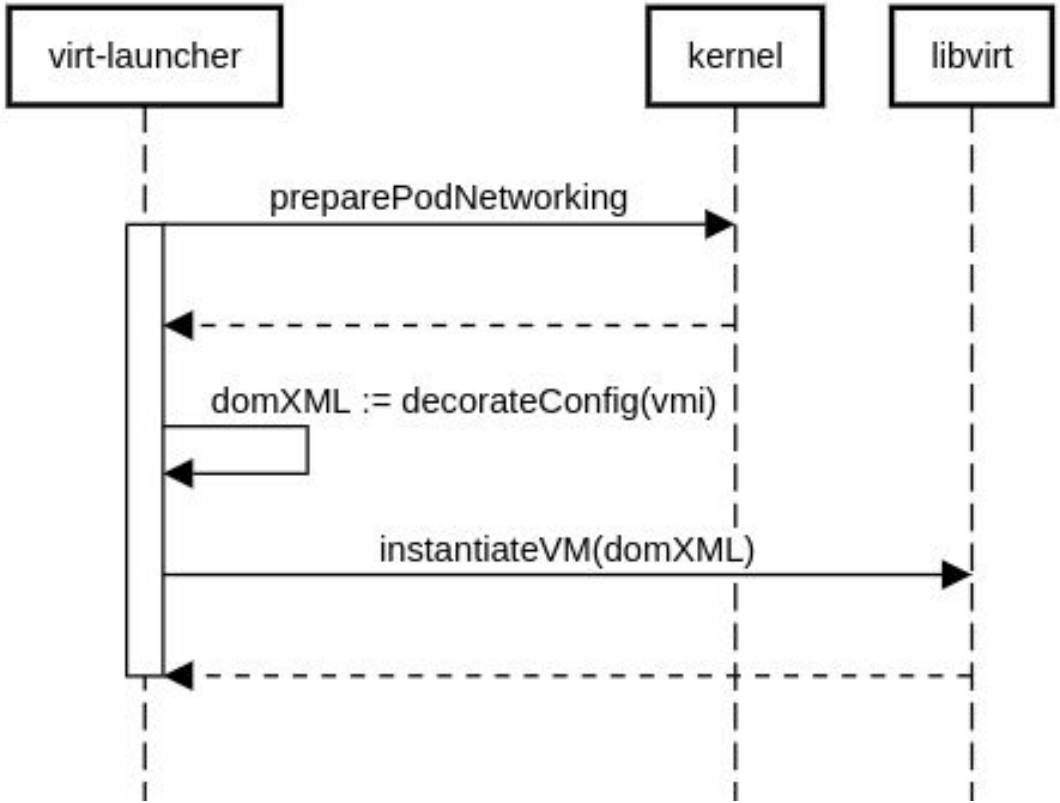
Motivation & Goals

Motivation, Problem, and Goals

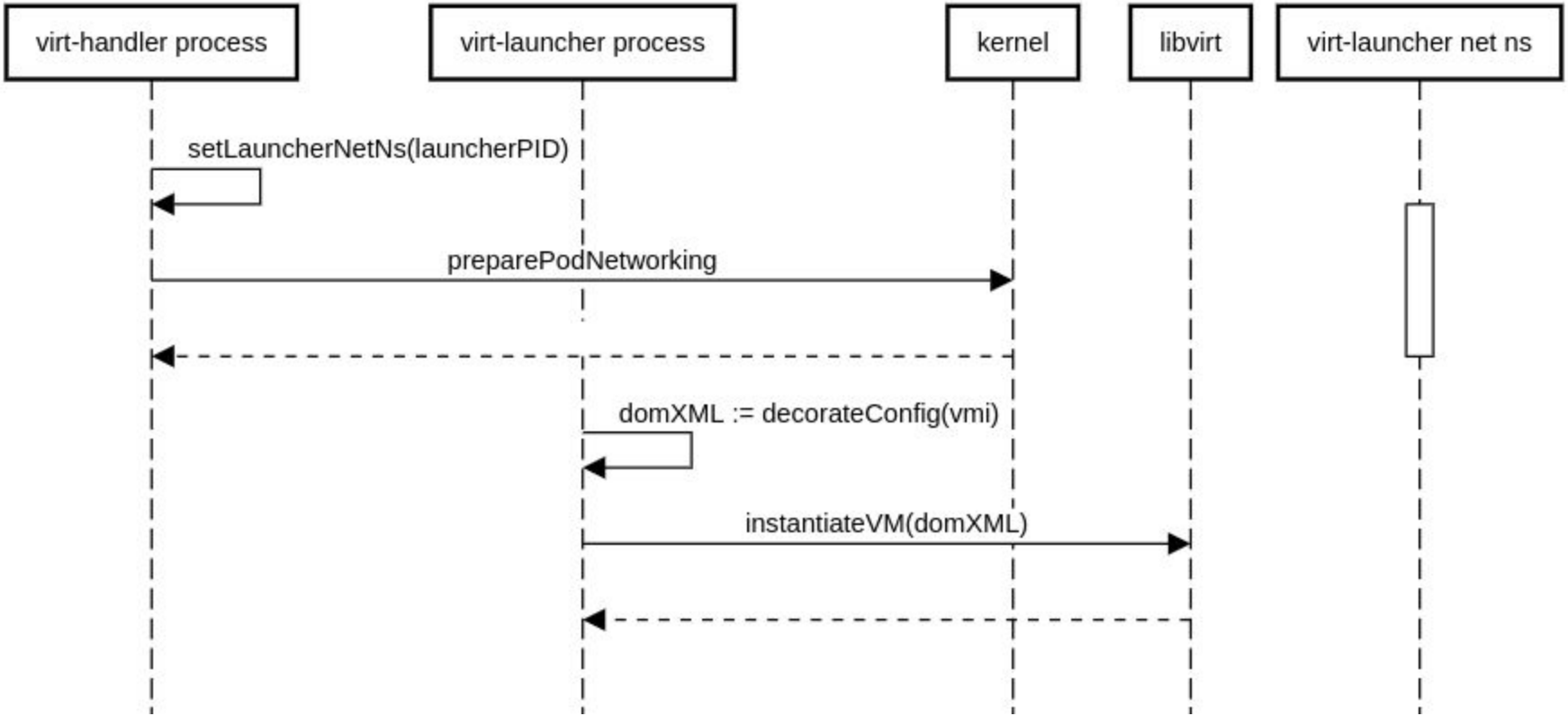
- Motivation
  - Reduce solution attack surface
- Problem
  - An untrusted component executed privileged operations
- Goals
  - Get rid of CAP_NET_ADMIN, CAP_NET_RAW from the untrusted component

Pod networking configuration split

(Original) Pod networking setup



Pod networking setup



VM networking configuration split

- https://github.com/kubevirt/kubevirt/issues/3085
- https://github.com/kubevirt/kubevirt/pull/2837

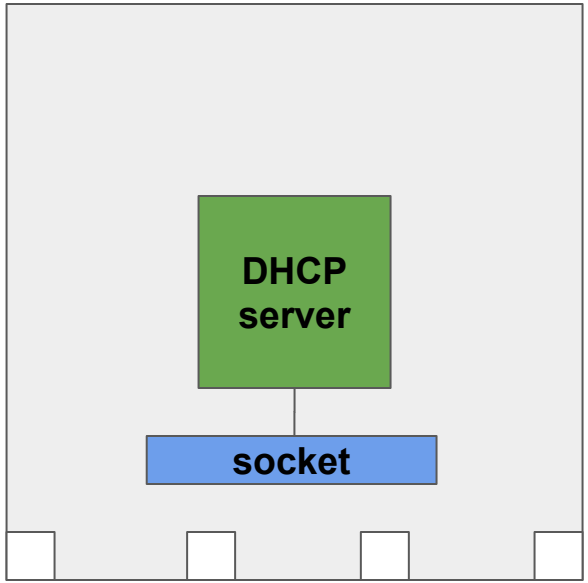
Dropping CAP_NET_RAW

Getting rid of CAP_NET_RAW

- **SO_BINDTODEVICE** socket option is ***set*** on the DHCP server
- https://github.com/kubevirt/kubevirt/pull/4501

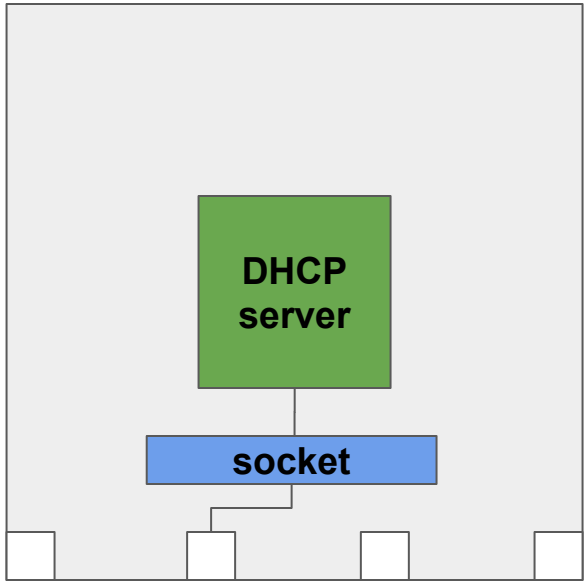
Bound to device

1. Create **DGRAM** socket



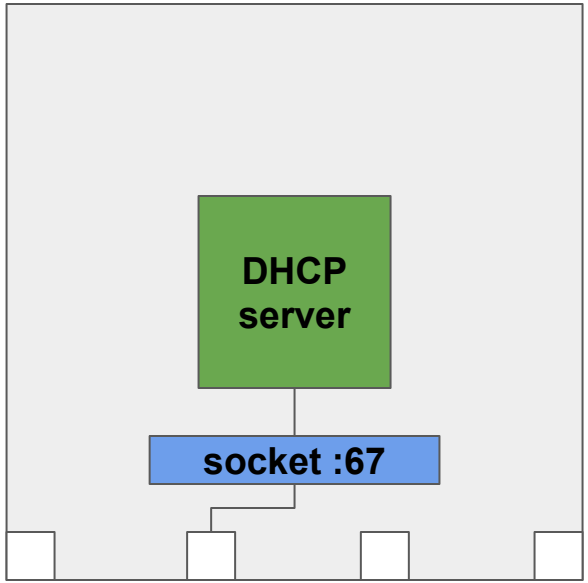
Bound to device

1. Create **DGRAM** socket
2. setsockopt(**SO_BINDTODEVICE**, iface2)



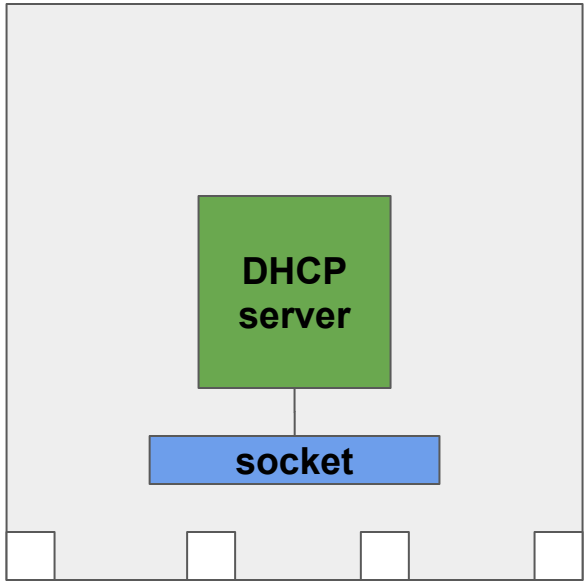
Bound to device

1. Create **DGRAM** socket
2. setsockopt(**SO_BINDTODEVICE**, iface2)
3. bind(socket, **port**)



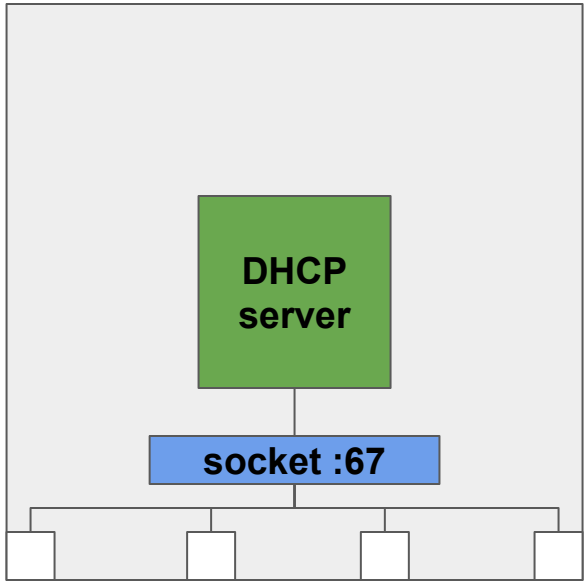
Filter by interface index

1. Create **DGRAM** socket



Filter by interface index

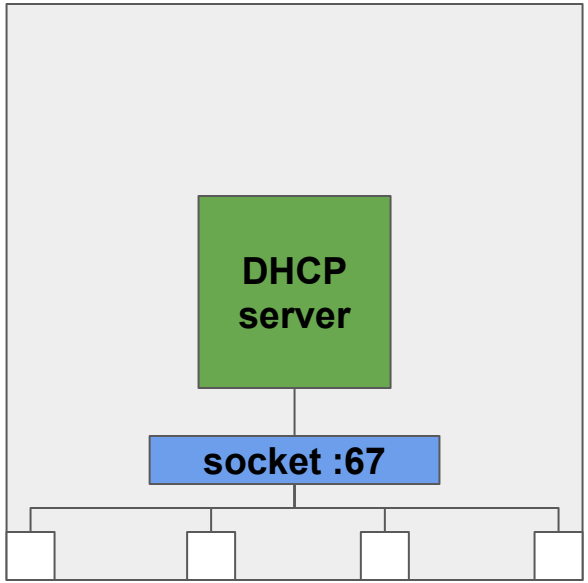
1. Create **DGRAM** socket
2. bind(socket, **port**)



Filter by interface index

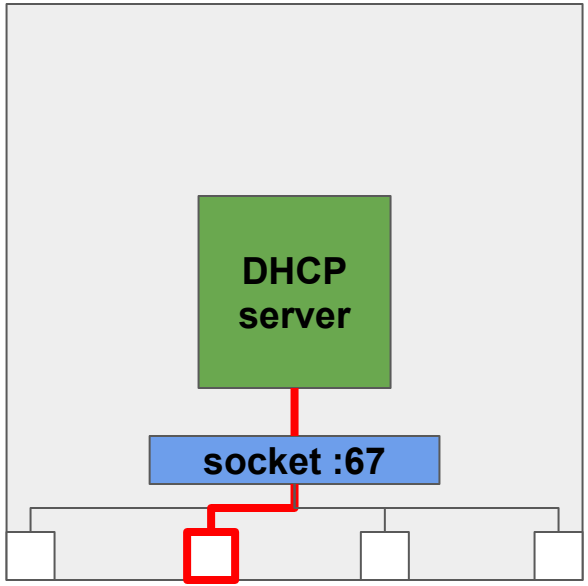
1. Create **DGRAM** socket
2. bind(socket, **port**)
3. Request iface index on read msgs

```
if err := p.SetControlMessage(ipv4.FlagInterface, true); err != nil {
  return nil, err
}
```



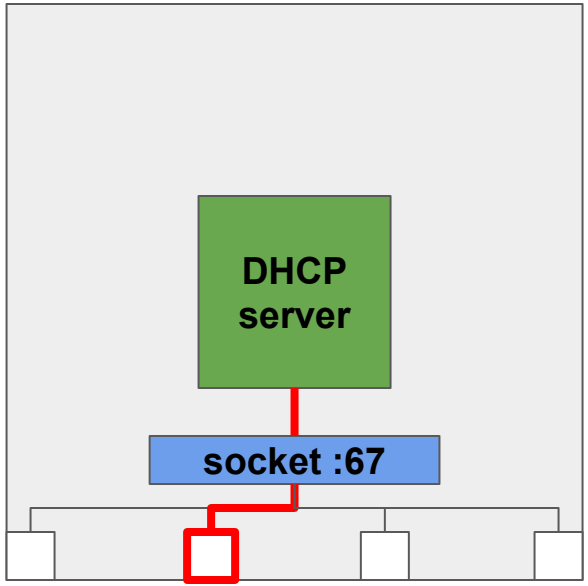
Filter by interface index

1. Create socket
2. bind(socket, port)
3. Request iface index on read msgs
4. Silently drop msgs if msg.iface != desired iface



Filter by interface index

1. Create socket
2. bind(socket, port)
3. Request iface index on read msgs
4. Silently drop msgs if msg.iface != desired iface
5. We *only* reply to messages; the DHCP reply is sent *to* the same interface



Dropping CAP_NET_ADMIN

Things requiring CAP_NET_ADMIN

- Create (and configure) tap device
- Create bridge
- Connect tap to bridge
- Configure bridge
  - Configure MAC / MTU in the bridge
  - Disable TX Checksum Offload

Things requiring CAP_NET_ADMIN

- Create (and configure) tap device
- ~~Create bridge~~
- Connect tap to bridge
- Configure bridge
  - ~~Configure MAC / MTU in the bridge~~
  - Disable TX Checksum Offload

Things requiring CAP_NET_ADMIN

- Create (and configure) tap device
  - Libvirt ***must*** be ‘taught’ to consume a pre-configured tap device
  - Trusted component ***must*** create the tap device on behalf of the untrusted component
    - No “extra” privileges / capabilities on the untrusted component
  - The untrusted component ***must*** adapt its dom xml generation to use Libvirt’s new API
- ~~Create bridge~~
- Connect tap to bridge
- ~~Configure bridge~~
  - ~~Configure MAC / MTU in the bridge~~
  - ~~Disable TX Checksum Offload~~

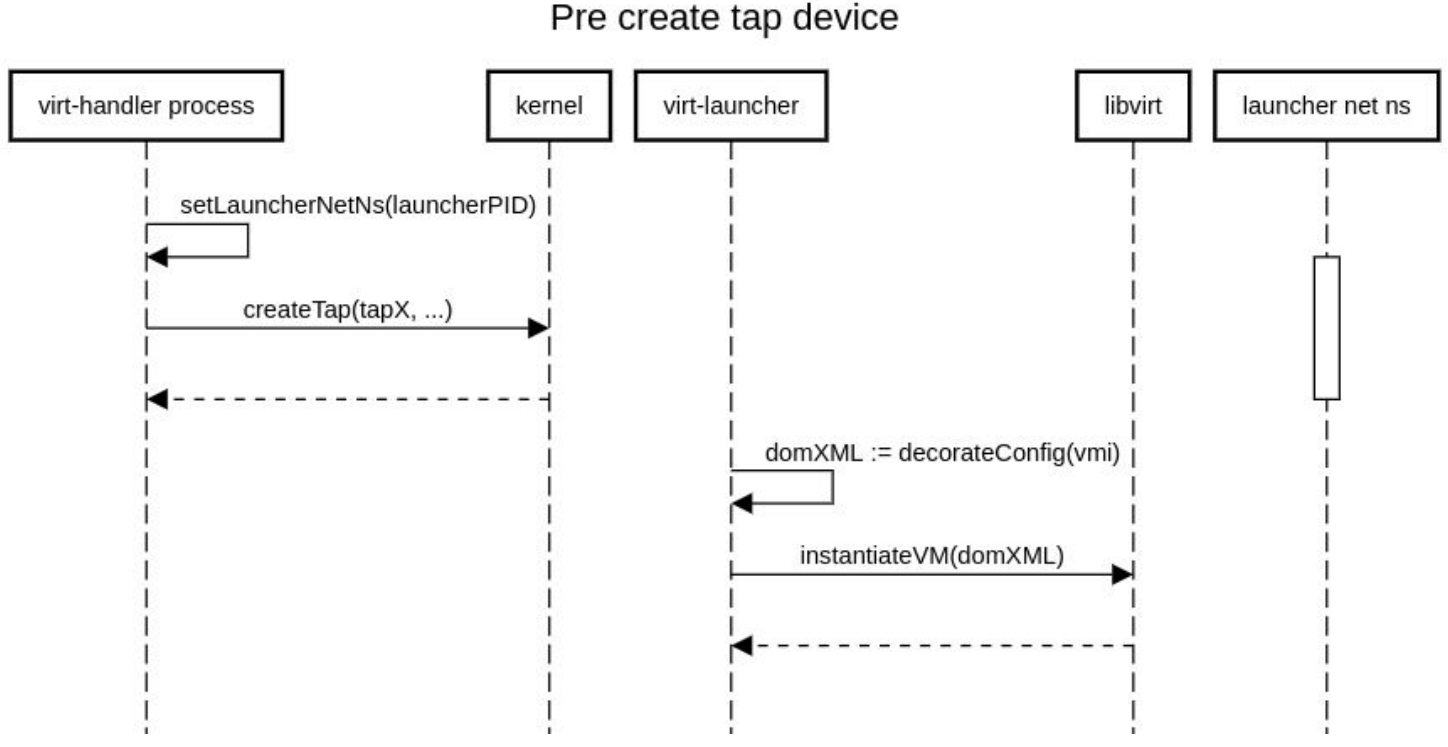
Libvirt: Import a pre-configured tap device

- https://bugzilla.redhat.com/show_bug.cgi?id=1723367
- Use ‘ethernet’ interface type
- Specify an existent tap device in the ‘target’ element
  - Specify the managed=’no’ attribute

```
<interface type='ethernet'>
  <mac address='02:00:00:2f:54:4f'/>
  <target dev='tap0' managed='no'/>
  <model type='virtio-non-transitional'/>
  <mtu size='1440'/>
  <alias name='ua-testmasquerade'/>
  <rom enabled='no'/>
  <address type='pci' domain='0x0000' bus='0x01' slot='0x00' function='0x0'/>
</interface>
```

Virt-handler: Create tap device on launcher’s net ns

- https://github.com/kubevirt/kubevirt/pull/3290



SELinux considerations

- Tap device created by virt-handler
  - Tap device structure - `tun_socket` - was using the virt-handler selinux context - `spc_t`
  - Virt-launcher (based on `container_t`) could not open the `tun_socket` resource
- Must create the tap w/ the correct virt-launcher label

```
func getProcessCurrentSELinuxLabel(pid int) (string, error) {
  launcherSELinuxLabel, err := selinux.FileLabel(
     fmt.Sprintf("/proc/%d/attr/current", pid))
  if err != nil {
     return "", fmt.Errorf(
        "could not retrieve pid %d selinux label: %v", pid, err)
  }
  return launcherSELinuxLabel, nil
}
```

Virt-launcher: update the binding mechanisms

- Instruct libvirt that the tap device is already created & configured; ready to consume
  - Update the `**BindMechanism**` `**decorateConfig**` method

```
<interface type='ethernet'>
  <mac address='02:00:00:2f:54:4f'/>
  <target dev='tap0' managed='no'/>
  <model type='virtio-non-transitional'/>
  <mtu size='1440'/>
  <alias name='ua-testmasquerade'/>
  <rom enabled='no'/>
  <address type='pci' domain='0x0000' bus='0x01' slot='0x00' function='0x0'/>
</interface>
```

Leftover ...

- Libvirt is being instructed to not touch the tap device (managed = “no”)
- Libvirt is being instructed to set the guest MTU
  - It should just ‘touch’ the guest VM interface MTU, and leave the tap device alone
  - Remember updating the tap device requires CAP_NET_ADMIN
- https://bugzilla.redhat.com/show_bug.cgi?id=1905929
  - Merged upstream a couple of days ago, waiting for a release

Next steps

- Wait for the fix to https://bugzilla.redhat.com/show_bug.cgi?id=1905929 to be released
- Update libvirt version w/ the patched version
- Remove CAP_NET_ADMIN from the virt-launcher pod template
  - Revert this commit

Thank you !!!

Resources

- Capabilities demo => https://github.com/maiqueb/fosdem2021
- Relevant bugs
  - Libvirt MTU bug => https://bugzilla.redhat.com/show_bug.cgi?id=1905929
  - Libvirt tap RFE => https://bugzilla.redhat.com/show_bug.cgi?id=1723367
- PRs
  - Split pod networking setup => https://github.com/kubevirt/kubevirt/pull/2837
  - Pre-create TAP device PR => https://github.com/kubevirt/kubevirt/pull/3290
  - Drop CAP_NET_RAW PR => https://github.com/kubevirt/kubevirt/pull/4501
  - Drop CAP_NET_ADMIN PR => https://github.com/kubevirt/kubevirt/pull/4506

Code Contributors

- Alona Kaplan - remove cap_net_raw
- Ihar Hrachyshka - pod networking configuration phase split
- Miguel Duarte Barroso - remove cap_net_admin