The KVM scripts can not undo the firewall rules as the tap interface is not available once the VM is deleted. The tap can be stored when the VM is created, or it may be obtained by the current rules given that we have the MAC address of the interface.

Associated revisions

Revision 2ae6ac33 - 10/27/2009 11:46 AM - Jaime Melis

Modified ebtables scripts to use FORWARD rules (#160)

git-svn-id: http://svn.opennebula.org/one/trunk@895 3034c82b-c49b-4eb3-8279-a7a1fdf6c0c0

Revision 033a20eb - 10/27/2009 05:03 PM - Jaime Melis

Modified ebtables-xen script to use FORWARD rules (#160)

git-svn-id: http://svn.opennebula.org/one/trunk@898 3034c82b-c49b-4eb3-8279-a7a1fdf6c0c0

Revision 8d4caf1 - 05/12/2017 12:23 PM - Jaime Melis

Merge pull request #160 from atodorov-storpool/tmCephOnSharedFs

tm/ceph [pre,post]migrate do nothing if not SYSTEM DS

History

#1 - 10/07/2009 09:25 PM - Ruben S. Montero

Also check that the isolation is really taking place.

A message from Shi Jin:

```
  After some debugging, I think in order to restrict a VM to a particular MAC address, we need to work on the FORWARD chain. Therefore I added the following to the ebtables-kvm script:
  forward_rule1="FORWARD -s ! #{iface_mac}/FF:FF:FF:FF -j DROP"
  forward_rule2="FORWARD -d ! #{iface_mac}/FF:FF:FF:FF -j DROP"
```
And call them in start (similar in stop
activate(forward_rule1)
activate(forward_rule2)

This has been working for me. If I tried to change the MAC address

within the VM I will lose connection.

I guess similarly I can work out rules on FORWARD to prevent users

from changing the VM IP address as well.

#2 - 10/09/2009 12:21 PM - Ruben S. Montero
- Assignee changed from Jaime Melis to Javi Fontan

#3 - 10/09/2009 02:17 PM - Javi Fontan
- Status changed from New to Assigned

#4 - 10/23/2009 04:52 PM - Javi Fontan

Added a new script 'ebtables-flush' that deletes all ebtables rules not used. Commit r889

#5 - 10/24/2009 05:28 AM - Shi Jin

Javi Fontan wrote:

Added a new script 'ebtables-flush' that deletes all ebtables rules not used. Commit r889

Hi Javi,

I tried your script. It removes all ebtables, including those are still in use.
I am running KVM as the hypervisor. I think the issue is that the rules operate on vnet1, vnet2,... instead of br0, br1, br2. Therefore,
[[if !interfaces.include?(rule[:interface])]]
is always true. Maybe this works for Xen but it does not for my KVM case.
How do you think?
Thanks.

#6 - 10/24/2009 07:18 AM - Shi Jin

I've modify the script to work with my KVM based OpenNebula system. The code is

#!/usr/bin/env ruby

require 'rexml/document'

def deactivate(rule)
  system "sudo ebtables -D #{rule}"
def interfaces_for_VM(vm, interfaces)
    nets=`virsh dumpxml #{vm}`
    doc=REXML::Document.new(nets).root
    doc.elements.each('/domain/devices/interface') {|net|
        tap=net.elements['target'].attributes['dev']
        interfaces.push(tap)
    }
end

def get_vm_list
    vmList=`virsh list`
    vms=vmList.split("\n")[2..-1].collect{|l| l.split.first}
    vms
end

def get_interfaces
    interfaces=[]
    vmList=get_vm_list
    vmList.each do |vm|
        interfaces_for_VM(vm, interfaces)
    end
    interfaces
end

RULE_TYPES={
    'INPUT' => /-i ([\w\.-]+) /,
    'OUTPUT' => /-o ([\w\.-]+) /
}

def get_rules
    rules=Hash.new
    RULE_TYPES.each do |name, reg|
        r=Array.new
        ebtables_exit=`sudo ebtables -L #{name}`
        rules[name]=ebtables_exit.split("\n")[3..-1].collect{|l|
            line=l.strip
            m=line.match(reg)
            if m
                interface=m[1]
                { :interface => interface,
                :rule => line
                }
            else
                nil
            end
        end.compact
    end
    rules
end

05/13/2021

3/4
interfaces=get_interfaces

all_rules=get_rules

all_rules.each do |chain, rules|
  rules.each do |rule|
    if !interfaces.include?(rule[:interface])
      deactivate("#{chain} #{rule[:rule]}")
    end
  end
end
end

#7 - 10/26/2009 10:53 PM - Ruben S. Montero

There was a couple of bugs in the rules for the ebtables. As Shi Jin suggested previously, the target rule should be FORWARD. Let say that a VM has virtual NIC (vnet3) attached to the bridge for the private networks. The rules should be:

# Accept all packets traversing the forward rule
ebtables -P FORWARD ACCEPT

# Filter those packets with a MAC not included in the virtual net
# This rules applies to packets being forwarded from the bridge to vnet3
ebtables -A FORWARD -s ! 00:03:c0:a8:c8:00/ff:ff:ff:ff:ff:00 -o vnet3 -j DROP

# Filter those packets with a MAC different from that assigned by OpenNebula
# This rules applies to packets being forwarded from vnet3 to the bridge
ebtables -A FORWARD -s ! 00:03:c0:a8:c8:0d -i vnet3 -j DROP

Also we do not really see the need to filter packets based on the IP. The user is responsible of setting the IPs in her own private network. However this could be very useful if someone wants to provide basic firewalling capabilities for the public IPs, like Amazon EC2 for the VMs...

#8 - 10/27/2009 12:48 PM - Jaime Melis
- Status changed from Assigned to Closed
- Resolution set to fixed

Modified scripts according to Ruben Montero’s comment. Tested and it works. Closing ticket.