Шилина Дарья Александровна студентка кафедра комплексной безопасности критически важных объектов, Российский государственный университет им. И.М. Губкина, Россия, г. Москва e-mail: schilina.dash@yandex.ru

Подрезов Павел Александрович студент, кафедра комплексной безопасности критически важных объектов, Российский государственный университет им. И.М. Губкина, Россия, г. Москва

НАСТРОЙКА СЕТЕВОГО МОСТА НА БАЗЕ КУМ

Аннотация: Конфигурирование мостов — важная задачей настройки сети виртуальных машин. Благодаря мостам происходит подключение к физической сети и возможность изолироваться нескольким виртуальным машинам от хост-системы. Данная статья создана с целью, дать общее понимание и представление по настройке сетевых мостов на базе KVM с использованием различных инструментов и технологий. Мы рассмотрим три основных инструмента для настройки сетевых мостов в KVM, а именно iproute2, openvswitch, systemd-networkd. Для проверки реализации воспользуемся простой топологией, представленной на рисунке 1. В качестве хоста будет выступать ALT Linux версии 10.4, а на гостевых машинах будет стоять Rosa Plasma 5.12.5.1.

Ключевые слова: сетевой мост, конфигурация, iproute2, openvswitch, systemd-networkd, KVM.

Shilina Daria Alexandrovna student, Department of Integrated Security of Critical Facilities Russian State University named after I.M. Gubkin Russia, Moscow

Podrezov Pavel Alexandrovich student, Department of Integrated Security of Critical Facilities, Russian State University named after I.M. Gubkin Russia, Moscow

SETTING UP A KVM-BASED NETWORK BRIDGE

Abstract: Configuring bridges is an important task of configuring a network of virtual machines. Bridges enable connection to a physical network and the ability for several virtual machines to isolate themselves from the host system. This article is designed to provide a general understanding and understanding of how to set up KVM-based network bridges using various tools and technologies. We will look at three main tools for configuring network bridges in KVM, namely iproute2, openvswitch, and systemd-networkd. To verify the implementation, we will use the simple topology shown in Figure 1. ALT Linux version 10.4 will act as the host, and Rosa Plasma 5.12.5.1 will be installed on guest machines.

Key words: network bridge, configuration, iproute2, openvswitch, systemd-networkd, KVM.



Рисунок 1. Топология (Topology)

Каждая из настроек сетевых мостов имеет свои плюсы и минусы, важно осознавать цель создания моста. Начнем более подробный обзор с **iproute2**.

Iproute2 — это набор утилит для управления параметрами сетевых устройств. Является классической для настройки и в основном используется для простых сетевых окружений. Использует всего три основных утилиты: ip, tc, ss.

Его реализация не сложная, ведь является встроенной в системах Linux. Это служит большим преимуществом при базовой настройке сети. Iproute2 позволяет настраивать физические и виртуальные интерфейсы, задавать IPадреса, поднимать и опускать интерфейсы, а также управлять их настройками и производить настройку маршрутизации. Поддерживает QoS и VLAN. Утилита tс позволяет настраивать политику управления трафиком, а также можно создавать туннели GRE и VXLAN. Iproute2 поддерживает такие протоколы, как IPv6 и MPLS, что делает его актуальным для современных сетевых решений.

Пример настройки маршрутизации:

ip route add 10.0.0.0/8 via 192.168.1.1

Пример настройки политик:

tc qdisc add dev eth0 root handle 1: htb default 30

Кратко продемонстрируем и опишем настройку моста с помощью iproute2:



Рисунок 2. Создание моста

2. Проверяем создание моста, выводя информация о сетевых интерфейсах на ALT:

[pavel@host-15 Рабочий стол]\$ ip -c a
1: lo: <loopback,up,lower_up> mtu 65536 qdisc noqueue state UNKNOWN group defaul</loopback,up,lower_up>
t qlen 1000
link/loopback 00:00:00:00:00 brd 00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
2: enp0s3: <broadcast,multicast,up,lower_up> mtu 1500 qdisc fq_codel master br0</broadcast,multicast,up,lower_up>
state UP group default qlen 1000
link/ether 08:00:27:25:d7:44 brd ff:ff:ff:ff:ff:ff
inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
valid_lft 86058sec preferred_lft 86058sec
<pre>inet6 fe80::a00:27ff:fe25:d744/64 scope link</pre>
valid_lft forever preferred_lft forever
3: br0: <broadcast,multicast,up,lower_up> mtu 1500 qdisc noqueue state UP group</broadcast,multicast,up,lower_up>
default qlen 1000
link/ether 46:41:44:e7:ac:d3 brd ff:ff:ff:ff:ff:ff
inet 192.168.1.1/24 scope global br0
valid_lft forever preferred_lft forever
<pre>inet6 fe80::4441:44ff:fee7:acd3/64 scope link</pre>
valid_lft forever preferred_lft forever

Рисунок 3. Сетевые интерфейсы

3. Добавляем на интерфейсы enp1s0 ip адреса для Rosa1 и Rosa2, а далее

для проверки правильности настройки запускаем ping.

-: bash Konsole		* 💹 🖈	~: bash — Konsole
Файл Правка Вид Закладки Модули Настройка Справка		Файл Правка Вид Закладки Модули Настройка	Справка
📑 Новая вкладка 🔲 Разделить окно 🗸	🖹 Копировать 📄 Вставить	📑 Новая вкладка 👖 Разделить окно \vee	🖄 Копировать 💼 Вставить
uslid_lff forever preferred lff forever Liverses-begen = 5 pin 92/108.1.2 PING 192.108.1.2 (192.108.1.2) 50(84) bytes of data. 64 bytes from 192.108.1.2: long_seq2 tile4 timee.090 ms 64 bytes from 192.108.1.2: long_seq2 tile4 timee.001 ms cipytes from 192.108.1.2: long seq2 tile4 timee.004 ms tile4 bytes from 192.108.1.2: long seq4 tile4 timee.004 ms tile5 from 192.108.1.2: long seq4 tile4 timee.004 ms tile5 from 192.108.1.2: long seq4 tile4 timee.02 ms tile5 trom 192.108.1.2: long seq4 tile4 timee.02 ms tile5 trom 192.108.1.2: long seq4 tile4 timee.03 ms		<pre>inct : 1//28 scope host valid (if forever preferred_iff forever : employ embadocat ymatricaty or income UPP mark inct 120: 100: 100: 100: 100: 100: 100: 100:</pre>	1500 gdisc fq_codel state UP group default glen 1000 ffiff .noprefixroute. .ni,50 ms .woue state UNKNOMM group default glen 1000 00:00:00 1500 gdisc fq_codel state UP group default glen 1000 ffiff .noprefixroute.

Рисунок 4. Проверка ping

Следующей рассмотрим утилиту **openvswitch**. Она уже хорошо работает в

сложных сетевых окружениях, потому что в отличие от стандартных Linuxмостов предоставляет большой функционал, включающий в себя, например, VLAN, tunneling и разные другие.

OVS, по сути, работает, как коммутатор уровня L2, так как перенаправляет трафик между портами. Его работа осуществляется как на физических серверах, так и внутри виртуальных машин. Openvswitch имеет весь основной функционал iproute2, а также обладает совместимостью с SDN, механизмом фильтрации ACL и шифрования, поддерживает мультикастинг.

Пример настройки QoS, создание очереди:

ovs-vsctl set port eth0 qos=@newq -- --id=@newq create QoS type=linux-htb other-config:max-rate=1000000

Применение QoS к порту:

ovs-vsctl set port eth0 qos=@newq

Openvswitch оптимизирован для работы в высоконагруженных окружениях. Может работать как на уровне пользовательского пространства, так и в режиме ядра, что позволяет достичь высокой производительности. Широко используется в облачных платформах, таких как OpenStack, а также в крупных корпоративных сетях. Подходит для виртуализированных сред, где необходимы сложные сценарии управления трафиком и сетевыми ресурсами.

Пример создания сетевого моста:

- 1. Установку при помощи команды: apt-get install openvswitch
- Непосредственное создание моста при помощи таких команд как: Sytemctl start openvswitch Ovs-vsctl add-br ovs0
- 3. Добавление интерфейсов:

Ovs-vsctl add-port ovs0 vnet1

Ovs-vsctl add-port ovs0 vnet2

4. Проверка корректности настройки и запуск ping для заранее установленных ip адресов:



Рисунок 5. Проверка настройки

Thet 10.0 2 3724 brd 10.0 2 35 Scope global dyn	-: bash konsole
valid_lft 85066sec preferred_lft 85066sec	
inet6 fold: about 27ff fo25 d74/64 scope link	Модули Настройка Справка
wawn правка вид закладки модули Hactpon valid_lft forever preferred_lft forever	
3: ovs-system: <broadcast,multicast> mtu 1500 qdisc n</broadcast,multicast>	cop state UNM group default glen 1000 🛛 👘 🗸 🖉 Romepowers 🖓 Romepowers 🖓 Romepowers
☐ Hogam Bixnagka	
4: br0 <broadcast, multicast=""> mtu 1500 gdisc noop sta</broadcast,>	te una group default glen 1000 : icmp_seq=3 ttl=64 time=0.563 ms
64 bytes from 192.168.122.186: icmp_seg=1 ttl link/ether 08:00:27:25:d7:44 brd ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:	: icmp_seq=4 ttl=64 time=0.552 ms
64 bytes from 192.168.122.106: icmp_seq=2 ttl 8: wnet1: <broadcast_multicast_up_lower up=""> mtu 1500</broadcast_multicast_up_lower>	gdisc fg codel master ovs-system state UNKNOWN group : icmp_seq=5 ttl=64 time=0.695 ms
64 bytes from 192.168.122.166; icmp_seq=3 ttl default glen 1000	: icmp seq=6 ttl=64 time=1.25 ms
64 bytes from 192.168.122.106: icmp_seq=4 ttl link/ether fe:54:00:f7:98:42 brd ff:ff:ff:ff:ff:ff:ff:	: icmp seq=7 ttl=64 time=0.812 ms
64 bytes from 192.168.122.106; icmp_seq=5 ttl g: uneta: <broadcast.multicast.up.lower up=""> mtu 1500</broadcast.multicast.up.lower>	udisc fg codel master ovs-system state UNKNOWN group : icmp seq=8 ttl=64 time=0.933 ms
64 bytes from 192.168.122.186: icmp_seq=6 ttl default glen 1000	
64 bytes from 192.168.122.106: icmp_seq=7 ttl link/ether fe:54:00:1f:17:7f brd ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:	168.122.152
64 bytes from 192.168.122.106: icmp seq=8 ttl 10: own0: <broadcast multicast=""> mtu 1500 gdisc poop s</broadcast>	tate mail group default glen 1000
64 bytes from 192.168.122.166: icmp_seq=9 ttl link/ether 70:80:65:60:ab:46 brd ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:	.122.152) 56(84) bytes of data.
64 bytes from 192.168.122.106; icmp seq=10 tt	
64 bytes from 192.168.122.106: icmp seq=11 tt	
64 bytes from 192.168.122.106; icmp_seq=12 ttl=64 time=0.661 ms	5 packets transmitted, 0 received, 100% packet loss, time 4113ms
64 bytes from 192.168.122.106: icmp seq=13 ttl=64 time=0.638 ms	
64 bytes from 192.168.122.106: icmp_seq=14 ttl=64 time=0.804 ms	livearosa-mp5kgh - \$ ip -c a
64 bytes from 192.168.122.106: icmp seq=15 ttl=64 time=0.592 ms	1: lo: <loopback.up.lower up=""> mtu 65536 gdisc nogueue state UNKNOWN group default glen 1000</loopback.up.lower>
64 bytes from 192.168.122.106: icmp_seq=16 ttl=64 time=0.631 ms	link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00
64 bytes from 192.168.122.106: icmp_seq=17 ttl=64 time=1.52 ms	inet 127,0.0,1/8 scope host lo
64 bytes from 192.168.122.106: icmp_seq=18 ttl=64 time=0.700 ms	valid lft forever preferred lft forever
64 bytes from 192.168.122.106: icmp_seq=19 ttl=64 time=0.760 ms	inet6 11/128 scope host
64 bytes from 192.168.122.106: icmp_seq=20 ttl=64 time=0.511 ms	valid lft forever preferred lft forever
64 bytes from 192.168.122.186: icmp seq=21 ttl=64 time=0.573 ms	2: enpls0: <broadcast.multicast.up.lower up=""> mtu 1500 gdisc fg codel state UP group default glen 18</broadcast.multicast.up.lower>
64 bytes from 192.168.122.106: icmp_seq=22 ttl=64 time=0.580 ms	link/ether 52:54:00:1f:17:7f brd ff:ff:ff:ff:ff
64 bytes from 192.168.122.186: icmp seq=23 ttl=64 time=1.45 ms	inet 192,165,122,106/24 brd 192,168,122,255 scope global dynamic noprefixroute enpls0
^c	valid lft 2912sec preferred lft 2912sec
192.168.122.106 ping statistics	inet6 fe80::4047:19b4:d019:e405/64 scope link noprefixroute
23 packets transmitted, 23 received, 0% packet loss, time 22469ms	valid lft forever preferred lft forever
rtt min/avg/max/mdev = 0.469/0.785/1.524/0.281 ms	livemrosa-np5kgh - 5
live@rosa-xg7ksi - \$	

Рисунок 6. Проверка ping

Наконец, перейдем к **systemd–networkd.** Эта служба является системным демоном, предназначенным для конфигурирования сети. Этот подход является довольно современным, интегрированным с systemd.

Управление сетевыми интерфейсами происходит при помощи конфигурационных файлов, что делает его крайне удобным для администрирования. Достаточно простой формат и читаемость записей гораздо упрощают настройку, нежели работа с множеством команд в терминале. Большим преимуществом, в сравнении с другими методами настройки, можно выделить то, что для изменения какого-то конкретного файла конфигурации достаточно перезагрузить только его, а не терять время на перезагрузку всей сети.

Интеграция с другими компонентами systemd, позволяет упростить автоматизацию процессов, к примеру, можно использовать таймеры для периодической проверки сети, просто создав unit-файл со скриптом:

[Unit]

Description=Check Network Status

[Service]

Type=oneshot ExecStart=/path/to/check_network.sh

WantedBy=timers.target

Systemd-networkd демонстрирует наибольшую эффективность в средах, где systemd выступает в роли системы инициализации. Он широко используется на серверах контейнерного типа, а также в облачных решениях и системах, которые нуждаются в централизованном управлении сетевыми интерфейсами. Это достаточно универсальный инструмент, ведь он использует разнообразные протоколы и буквально каждый параметр сети можно адаптировать в соответствии с конкретными требованиями организации или проекта, обеспечивая полное управление конфигурацией сетевых интерфейсов.

Создаем файл конфигурации для моста по адресу: /etc/systemd/network/10br0.netdev

И записываем туда следующие настройки:



Рисунок 7. Настройки моста

Аналогично создаем файл по адресу /etc/systemd/network/10-br0.network со следующим содержимым:



Рисунок 8. Настройки моста

Сюда же прописываем настройки для наших интерфейсов:

GNU nano	7.2	/etc/systemd/network/10-br0.network	
[Match] Name=nt0		т. — констроител. — Е. — вставите	t (
[Network] DHCP=yes			
[Match]			
Name=vnet0 [Network]			
Bridge=nt0			
[Match] Name=vnet1 [Network] Bridge=nt0	dynami) te		

Рисунок 9. Настройки интерфейсов vnet0 и vnet1

После чего перезагружаем systemd-networkd командой:

Systemctl restart systemd-networkd

Проверяем появился ли наш мост nt0:

	root@host-15: /root	
Файл Правка Вид Поиск Т	Герминал Помощь	
[pavel@host-15 Рабочий с	стол] \$ su -	A
Password:		
[root@host-15 ~]# ip -c		
1: lo: <loopback,up,lowe< td=""><th>R_UP> mtu 65536 qdisc noqueue state UNKNOWN grou</th><td>p defaul</td></loopback,up,lowe<>	R_UP> mtu 65536 qdisc noqueue state UNKNOWN grou	p defaul
t qlen 1000		
link/loopback 00:00:	00:00:00:00 brd 00:00:00:00:00:00	
inet 127.0.0.1/8 sco	ppe host lo	
valid_lft forever	preferred_1ft forever	
inet6 1/128 scope	nost	
Valid_ITT Torever	preferred_ITT forever	
2. Http://www.carkiek.broa	WCAST, MOLTICAST, OF > Mitu 1500 quist noqueue state	DOWN BI
link/ether 7e:96:26:	79.84.4h hrd ff.ff.ff.ff.ff	
3: enp0s3: <broadcast_mu< td=""><th>UTICAST UP LOWER UP> mtu 1500 gdisc fg codel sta</th><td>te UP gr</td></broadcast_mu<>	UTICAST UP LOWER UP> mtu 1500 gdisc fg codel sta	te UP gr
oup default glen 1000	crichshiper (conch_on > med hood darbe hd_coder ord	сс о. _В .
link/ether 08:00:27:	25:d7:44 brd ff:ff:ff:ff:ff	
inet 10.0.2.15/24 br	d 10.0.2.255 scope global dynamic noprefixroute	enp0s3
valid_lft 86377se	ec preferred_lft 86377sec	
inet6 fe80::a00:27ff	:fe25:d744/64 scope link	
valid_lft forever	preferred_lft forever	
[root@host-15 ~]# S		

Рисунок 10. Проверка настройки

Заранее установив адреса на машины, пингуем Rosa1 и Rosa2:

li	/e@rosa	a-nb8l	ıuh ~ \$ pir	ng 192.	168.122.206			
PI	NG 192.	168.1	122.206 (19	92.168.	122.206) 56	(84) byt	tes of data.	
64	bytes	from	192.168.12	22.206:	icmp_seq=1	ttl=64	time=1.97 m	IS
64	bytes	from	192.168.12	22.206:	icmp_seq=2	ttl=64	time=0.658	ms
64	bytes	from	192.168.12	22.206:	icmp_seq=3	ttl=64	time=0.747	ms
64	bytes	from	192.168.12	22.206:	icmp_seq=4	ttl=64	time=1.07 m	IS
64	bytes	from	192.168.12	22.206:	<pre>icmp_seq=5</pre>	ttl=64	time=1.00 m	IS
64	bytes	from	192.168.12	22.206:	icmp_seq=6	ttl=64	time=0.419	ms
64	bytes	from	192.168.12	22.206:	icmp_seq=7	ttl=64	time=0.716	ms

Рисунок 11. Проверка ping

А теперь разберёмся, как наши утилиты работают с KVM, и как происходит привязка адресов и интерфейсов в виртуализированной среде. Сначала виртуальная машина отправляет пакеты через виртуальный интерфейс vnet0 на мост br0, после чего эти пакеты уже пересылаются на физический интерфейс eth0, определяя получателя по MAC-адресу.

Ір-адрес назначается в виртуальной машине внутри гостевой операционной системы и обычно является адресом из той же подсети, что и ірадрес, назначенный br0. Таким образом, виртуальная машина получает сетевой доступ через физический интерфейс с ір-адресом, настроенным в виртуальной машине.

Физический интерфейс хоста (eth0), при использовании моста, фактически перестает иметь собственный ір-адрес, становясь частью моста. Его айпишник переходит к мостовому интерфейсу. Этот адрес используется для связи с другими машинами (физическими или виртуальными), подключенных к этому виртуальная машина получает, свой собственный же мосту. Каждая виртуальный сетевой интерфейс (vnet0) и он же подключается к созданному мосту, что позволяет виртуалкам взаимодействовать друг с другом. И, наконец, каждый виртуальный интерфейс, созданный для ВМ, получает свой уникальный MAC-адрес, который помогает мосту различать сетевые устройства, подключенные к нему.

Во всех трёх случаях, наш мост, настроенный через iproute2, openvswitch или systemd-networkd, является центральным элементом, который связывает виртуальные машины с физической сетью. Независимо от того, какой инструмент мы используем для создания моста, принцип один. Разница заключается лишь в том, как они реализуются в функциональности и инструментах управления.

Теперь мы можем привести сравнительную таблицу по трем видам настройки сетевого моста в KVM.

Таблица 1.

Сложнос	Функцион	Преимуществ	Недостатки	Использова
ть	ал	а		ние
настройк				
И				

Три вида настройки сетевого моста в KVM

iproute2	Простая	Ограничен	Простота в	Не удобен	Простые
		ный	использовани	для	сетевые
			И	сложных	окружения
				настроек	
			Встроенность		
openvswi	Сложная	Расширенн	Гибкость	Требует	Сложные
tch		ый		дополнител	сетевые
			Масштабируе	ьных	окружения
			мость	установок	
			Наличие	Сложный в	
			продвинутых	настройке	
			функций		
systemd-	Средняя	Умеренны	Современный	Зависимост	Окружения,
networkd		й		ь от	использую
			Использует	systemd	щие
			файлы		systemd
			конфигурации		

Итак, после оценки каждой конфигурации становится понятно, для каких целей подходит каждых из выбранных нами подходов. Iproute2 лучше всего использовать, когда ваша задача состоит в создание базового моста, не имеющего никаких сложных требований и продвинутых функций. Openvswitch надо использовать в сложных задачах, где важную роль играют гибкость, масштабируемость и продвинутые функции. Systemd-networkd является современным решением и использует декларативную настройку, описывая параметры сети в файлах конфигурации. Благодаря этому, в данных файлах можно описывать все желаемое состояние сети, вместо ввода большого количества последовательных команд. В связи с этим, данная конфигурация является универсальной для разного рода настроек.

Список литературы:

1. BaseALT. [Электронный ресурс] // Режим доступа: URL:https://www.basealt.ru/en/download (дата обращения: 31.12.2024 г.).

Libvirt (Qemu+KVM+Virt-manager). [Электронный ресурс] // Режим доступа: URL: https://www.altlinux.org/Libvirt_(Qemu%2BKVM%2BVirt-manager) (дата обращения: 31.12.2024 г.).

9

3. Установить rosa-linux. [Электронный ресурс] // Режим доступа: URL: https://rosa.ru/rosa-linux-download-links/ (дата обращения: 21.12.2024).

4. Руководство по использованию iproute2. [Электронный ресурс] // Режим доступа: URL: https://www.altlinux.org/Etcnet (дата обращения: 31.12.2024 г.).

5. Руководство по использованию openvswitch. [Электронный ресурс] // Режим доступа: URL: https://www.altlinux.org/Etcnet/openvswitch (дата обращения: 31.12.2024 г.).

6. Руководство по использованию systemd-networkd. [Электронный pecypc] // Режим доступа: URL: https://www.altlinux.org/Systemd-networkd (дата обращения: 31.12.2024 г.).

7. Уймин А.Г. Компьютерные сети. L2-технологии: практикум. М.: Ай Пи Ар Медиа, 2024. 191 с.

8. Уймин А.Г. Периферийные устройства ЭВМ: практикум. М.: Ай Пи Ар Медиа, 2023. 429 с.