KeyVirt

ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ

Программный комплекс «плагин keyvirt-ovn-firewall-plugin»

Управление сетевой безопасностью OVN (Open Virtual Network) для KeyVirt

Оглавление

1. Вводная

Плагин «keyvirt-ovn-firewall-plugin» встраивается в WebAdmin платформы KeyVirt и реализует полноценный веб-интерфейс управления сетевой безопасностью виртуальных машин, работающих через встроенный сетевой провайдер OVN («ovirt-provider-ovn») – без необходимости работать с REST API провайдера напрямую и без прямого доступа к базе данных OVN Northbound (принцип минимальных привилегий: бэкенд плагина работает от непривилегированного пользователя «ovirt» и не имеет доступа к OVN NBDB).

Основные функции:

- Группы безопасности и правила ACL (аналог Security Groups в OpenStack Neutron) – белый список разрешённого трафика;
- Готовые шаблоны групп портов (типовые сетевые сервисы: DNS, DHCP, веб, почта, LDAP/Kerberos/FreeIPA, файловые шары, удалённое управление, мониторинг, кластеризация) – подставляются прямо в форму создания правила без предварительного создания;
- Готовые шаблоны наборов адресов (приватные сети RFC1918, loopback, link-local, cloud-metadata, публичные DNS-резолверы) – для поля «Источник/назначение трафика»;
- Логические сети и подсети OVN (overlay/vlan/flat);
- Логические маршрутизаторы OVN с внешним шлюзом (External Gateway);
- Автоматизация физической привязки vlan/flat-сети к NIC хоста (несущая сеть, кластер, Non-VM Network, VLAN, привязка к адаптеру) одной кнопкой, с проверками на конфликты и безопасность;
- Интерактивная графическая Топология сети (SVG-диаграмма с перетаскиванием узлов) с диагностикой типовых рассогласований;
- Журнал аудита действий (подписанные HMAC записи, хранение 365 дней) в соответствии с Приказом ФСТЭК России №187 (защита КИИ);
- Контроль соответствия требованиям ФСТЭК №187 и учёт инцидентов;
- Интеграция с ГосСОПКА (отправка событий по Syslog RFC5424 или REST);
- Веб-форма задания сервисной учётной записи для доступа к «ovirt-provider-ovn» (без правки конфигурационных файлов по SSH), пароль хранится на диске в зашифрованном виде.

2. Системные требования

Компонент	Требование
ОС	RHEL 9 / CentOS Stream 9 / Platform V SberLinux OS Server 9.x архитектуры x86_64

Платформа	Engine 4.5.6 или новее (проверено на версиях 4.5.6 и 4.5.8 – полный цикл создание/подключение/удаление сети работает идентично)
Провайдер	«ovirt-provider-ovn» 1.2.37 или новее, с зарегистрированным и рабочим внешним провайдером сети (Compute → Providers)
Права	root для установки. Далее сервис работает от пользователя «ovirt»
Python	3.9+
Apache	httpd 2.4.51+ с модулем mod_wsgi 4.7.1+

Пакеты, ставящиеся автоматически как RPM-зависимости:

- python3-mod_wsgi >= 4.7.1
- python3-requests >= 2.25
- python3-cryptography (или python3.11-cryptography) – для шифрования пароля сервисной учётной записи
- python3-pip – для установки flask (не пакетирован в базовых репозиториях RHEL/CentOS 9 и Sberlinux-fstec-*, ставится через pip3 автоматически в %post)
- ovirt-openvswitch-ovn >= 2.15 – реальный минимум, который уже требует сам ovirt-provider-ovn (ovn2.13 >= 22.03 или ovn22.09 или ovn >= 22.03) – конкретное имя пакета OVN зависит от дистрибутива
- policycoreutils, python3-policycoreutils – для SELinux-политик

Backend плагина скомпилирован через Cython в нативный «.so»-файл – исходный Python-код в состав установленного пакета не входит (обфускация).

3. Установка и первоначальная настройка

3.1. Установка RPM

Выполнить от имени root на сервере oVirt Engine:

- bash dnf install ./keyvirt-ovn-firewall-plugin-<версия>-1.<dist>.x86_64.rpm

При **первой установке** пакет автоматически (сценарий «%post»):

- Размещает файлы бэкенда, UI-плагина, конфигурацию Apache (mod_wsgi), systemd-юнит (резервный вариант), logrotate;
- Выставляет владельца и права на конфигурационный файл «/etc/ovirt-engine/ovn-firewall.conf»: «root:ovirt», «640» – это важно, т.к. бэкенд работает от пользователя «ovirt» и должен иметь возможность прочитать собственный конфиг;

- Убирает из скопированного шаблона конфига строки «OVIRT_ENGINE_URL» / «OVN_PROVIDER_URL» / «OVN_IDENTITY_URL», содержащие «$(hostname -f)» – это не shell-подстановка (конфиг парсится построчно чистым Python, а не через shell), поэтому такие строки нужно убирать, чтобы сработали дефолты бэкенда на основе реального FQDN хоста («socket.getfqdn()»);
- Генерирует случайный «AUDIT_HMAC_SECRET» («openssl rand -hex 32») вместо статичной заглушки из шаблона;
- Пытается автоматически определить логин сервисной учётной записи («OVN_PROVIDER_USER»), читая «/etc/ovirt-provider-ovn/conf.d/10-setup-ovirt-provider-ovn.conf» (значение «ovirt-admin-user-name=») – имя AAA-профиля реально отличается между разными engine («admin@ovirt@internalsso» vs «admin@ovirt@internal»), угадывать одним статичным значением ненадёжно;
- Перезапускает httpd, чтобы подхватить новый mod_wsgi-конфиг.

При **обновлении** версии пакета ничего из вышеперечисленного заново не выполняется – существующий «/etc/ovirt-engine/ovn-firewall.conf» не трогается (RPM-механизм «%config(noreplace)»).

Владелец конфигурационного файла («root:ovirt») переустанавливается на каждой установке – и первой, и при обновлении – это исправление только прав доступа, не содержимого файла.

3.2. Пароль сервисной учётной записи

Единственный обязательный ручной шаг установка пароля сервисной учётной записи.

«ovirt-provider-ovn» не принимает SSO-токен WebAdmin напрямую – только логин/пароль через собственный Identity API (порт 35357). Логин плагин определяет сам (см. 3.1), а вот пароль – секрет, который RPM в принципе не может узнать заранее.

Рекомендуемый способ – через веб-форму (без SSH):

1) Открыть WebAdmin: «https://<engine-fqdn>/ovirt-engine/webadmin» В левом меню открыть пункт «OVN Firewall» (если не появился – обновить страницу браузера, Ctrl+F5; перезапуск «ovirt-engine» не требуется – движок читает дескрипторы плагинов при каждом HTTP-запросе к WebAdmin).
2) Перейти в «Настройки» → «Подключение к OVN».
3) В поле «Имя пользователя» уже должно быть подставлено значение, определённое автоматически при установке (например «admin@ovirt@internal») – это обычный администратор oVirt (не отдельная техническая учётная запись), состоящий в группе NetAdmin в конфигурации «ovirt-provider-ovn».
4) В поле «Пароль» ввести пароль этой учётной записи (как правило – тот же, что используется для входа в WebAdmin под этим логином).
5) Нажать «Проверить соединение» – должно появиться сообщение «Соединение успешно».

6) Нажать «Сохранить». Пароль на диске хранится в зашифрованном виде (AES-256-GCM, ключ дерайвится из «/etc/machine-id» конкретного сервера – скопировать файл настроек на другой сервер и расшифровать пароль там не получится) в отдельном файле «/var/lib/ovirt-engine/ovn-firewall-credentials.json» (не в основном «.conf»).

Изменения применяются немедленно, без перезапуска httpd.

Альтернативный способ – вручную, через SSH (нужен, только если веб-форма ещё не открывается из-за полного отсутствия пароля):

- vi /etc/ovirt-engine/ovn-firewall.conf
  a) # задать строку:
  b) # OVN_PROVIDER_PASSWORD=<пароль учётной записи OVN_PROVIDER_USER>
- systemctl reload-or-try-restart httpd

3.3. Проверка установки

Проверить, что пакет установлен и сервис активен:

- rpm -q keyvirt-ovn-firewall-plugin
- systemctl is-active httpd

Проверить базовую доступность API (не требует пароля из шага 3.2 – эта проверка смотрит только на TCP-доступность «ovirt-provider-ovn», не на корректность логина/пароля):

- curl -sk http://127.0.0.1/ovn-firewall-api/health | python3 -m json.tool

Ожидаемый ответ:

- json

```
{
  "status": "ok",
  "ovn_provider": "reachable",
  "audit_log": "ok",
  "api_model": "4.6",
  "gossopka_enabled": false
}
```

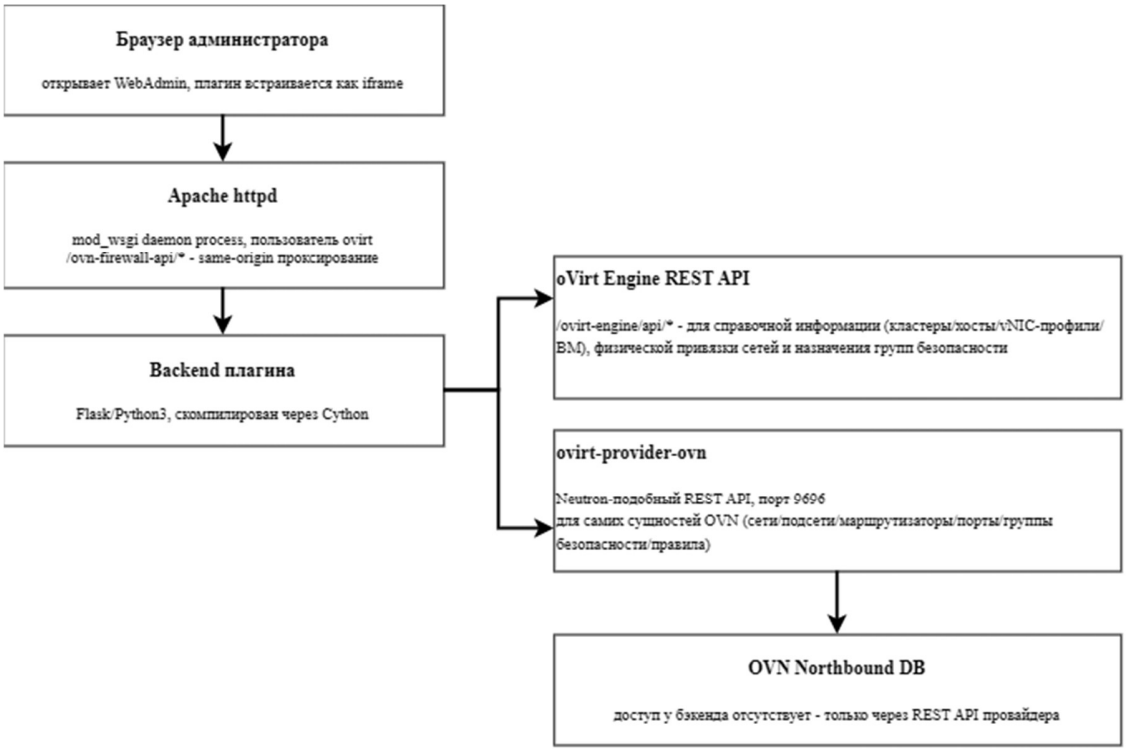
Важно: этот ответ будет «ok» даже если пароль из шага 3.2 ещё не задан или неверен – единственный надёжный способ проверить реальную работоспособность после шага 3.2 – кнопка «Проверить конфигурацию» на странице «Топология сети» внутри WebAdmin (см. раздел 5.2).

3.4. Раздача запросов

Пакет ставит как mod_wsgi-конфигурацию Apache («/etc/httpd/conf.d/keyvirt-ovn-firewall.conf» – это и есть реальный путь обработки запросов), так и systemd-юнит «keyvirt-ovn-firewall.service» – но второй нужен только как запасной вариант на случай отсутствия «python3-mod_wsgi» (при обычной установке через dnf эта RPM-зависимость уже гарантирует его наличие, поэтому запускать этот systemd-сервис не нужно). Проверить, что реально обслуживает трафик:

- httpd -M | grep wsgi

4. Архитектура



Аутентификация к плагину: заголовок «X-Auth-Token» с реальным SSO-токеном oVirt (не «Authorization: Bearer»). Токен пользователя проверяется через «GET /ovirt-engine/api» с этим токеном.

Аутентификация к «ovirt-provider-ovn»: отдельный механизм – служебная учётная запись (см. раздел 3.2), которой бэкенд получает собственный «neutron-токен» через Identity API провайдера (порт 35357, Keystone-совместимый v2.0), кэшируемый на ~1 час.

Стек технологий:

Компонент	Технология
Frontend	Vanilla JavaScript + PatternFly CSS Интерактивная SVG-графика для Топологии без внешних библиотек
Backend	Python 3.9 + Flask (скомпилирован через Cython)
WSGI-сервер	mod_wsgi daemon mode
Веб-сервер	Apache httpd 2.4
Шифрование	AES-256-GCM для пароля сервисной учётной записи, ключ из «/etc/machine-id» (python3-cryptography)

5. Работа с плагином – разделы меню

5.1. Панель управления

Дашборд со сводной статистикой (число сетей/маршрутизаторов/групп безопасности/правил) и лентой последних событий.

5.2. Топология сети

Интерактивная SVG-диаграмма всех сетей, подсетей, маршрутизаторов и внешнего шлюза, построенная в 4 яруса сверху вниз:

1) Внешняя сеть (обобщённый узел, появляется, если хоть у одного маршрутизатора настроен внешний шлюз);
2) Логическая сеть-шлюз (используемая как External Gateway);
3) Логические маршрутизаторы;
4) Логические сети (interior – подключённые как обычные внутренние).

Цвета линий:

- Сплошная зелёная – interior-подключение подсети к маршрутизатору;
- Пунктирная синяя – сеть-шлюз к маршрутизатору;
- Пунктирная красная – Внешняя сеть к сети-шлюзу.

Цвет окантовки узла:

- Маршрутизатор – чёрный;
- Внешняя сеть – красный;

- Сеть-шлюз – синий;
- Interior-сеть – зелёный.

Узлы можно перетаскивать мышью для удобной расстановки диаграммы (в пределах текущей сессии браузера – позиции не сохраняются на диск). Клик по узлу открывает справа панель деталей с доступными действиями.

Для сети: подключить/переключить/отключить маршрутизатор, создать или отредактировать подсеть, настроить физическую привязку, удалить.

Для маршрутизатора: назначить/снять внешний шлюз, редактировать, удалить.

Кнопка «Проверить конфигурацию» запускает набор диагностических проверок (не требует ручного вмешательства, только чтение):
- Доступность списков сетей/подсетей/маршрутизаторов/портов у провайдера;
- Если список маршрутизаторов недоступен целиком – точечный поиск конкретного битого маршрутизатора;
- Подсети без сети-владельца, сети без единой подсети;
- Повторяющиеся имена сетей/маршрутизаторов;
- Синхронизация сетей провайдера в объекты oVirt Engine («engine_provider_sync») – обнаруживает случай, когда «ovirt-provider-ovn» отвечает, но ни одна его сеть не синхронизирована в Engine (см. раздел 12);
- Физическая привязка «provider:physical_network» к хостам («physical_network_binding») – обнаруживает несущие сети без реальной привязки к NIC хоста (см. раздел 12).

5.3. Группы безопасности и правила ACL

«+ Создать группу» – название, описание, необязательная привязка к конкретной сети, флаги Stateful и Port Security.

«+ Добавить правило» в открытой группе – направление (ingress/egress), тип адреса (IPv4/IPv6), протокол (TCP/UDP/ICMP/ICMPv6/SCTP/любой), диапазон портов, источник/назначение трафика.

Направление (важно понимать правильно, путаница встречается часто):
- Egress (Исходящий) – трафик, уходящий ОТ ВМ этой группы наружу (к другой ВМ, к маршрутизатору, через SNAT в интернет).
- Ingress (Входящий) – трафик, приходящий К ВМ этой группы ИЗВНЕ (от другой ВМ, из внешней сети).

Важно (см. также раздел 11):
Egress в этой модели безопасности разрешён **всегда** и безусловно самим «ovirt-provider-ovn», независимо от того, какие правила настроены в группе – реально ограничить можно только ingress. Явное правило для DHCP (шаблон «dhcp-udp», 67/68 – см. ниже) при необходимости следует создавать с направлением egress: это соответствует тому, что ВМ

сама отправляет DHCPDISCOVER/DHCPREQUEST наружу (единственное направление DHCP, которое вообще имеет смысл описывать правилом ACL). Ответ сервера (DHCPOFFER/ACK, формально ingress) под ACL-фильтрацию фактически не подпадает – проверено вживую через «ovn-sbctl lflow-list»: OVN сам генерирует для него отдельный обходной поток в стадии «ls_out_acl» с приоритетом 34000 («match: outport==<port> && udp.src==67 && udp.dst==68», «action: ct_commit; next» – т.е. всегда разрешено), что значительно выше любых ACL групп безопасности (у них приоритет ~1000-2001). Поэтому **ответ DHCP-сервера никогда не блокируется настройками групп безопасности** – отдельное ingress-правило для него не требуется и не окажет эффекта.

Поддерживаемые поля правила ограничены тем, что реально принимает «ovirt-provider-ovn» (Neutron Security Group Rules API): «direction», «security_group_id» (обязательные); «ethertype», «protocol», «port_range_min», «port_range_max», «remote_ip_prefix», «remote_group_id», «description» (опциональные). Поля «action», «priority», «match», «log», «stateful», «meter» не поддерживаются провайдером – правила транслируются в OVN ACL автоматически по модели белого списка.

Кнопка «Правила OVN» на списке ВМ – эта кнопка не в самом плагине, а добавлена в панель инструментов родного списка виртуальных машин WebAdmin (Виртуализация → Виртуальные машины). При выборе одной ВМ и нажатии переходит на страницу «Правила» и автоматически фильтрует список правил ACL: показывает только правила тех групп безопасности, которые реально назначены сетям vNIC-профилей выбранной ВМ (то есть назначены через значок 🔗 на «Логических сетях», раздел 5.4). Наверху таблицы правил появляется баннер с именем ВМ и кнопкой «Сбросить фильтр». Если у ВМ нет ни одной назначенной группы безопасности – выводится предупреждение (список правил останется пустым, это ожидаемо, а не ошибка).

Селектор «Шаблон / группа портов» в форме правила всегда виден (не спрятан за выбором протокола) и подмешивает готовые шаблоны вместе с уже созданными пользователем группами портов (раздельные группы в выпадающем списке, без дублирования одинаковых имён). Выбор готового шаблона сам подставляет протокол – он зашит в суффикс имени шаблона («-tcp»/«-udp»). Список готовых шаблонов групп портов по категориям:

Категория	Имя шаблона	Порты	Описание
Базовые сетевые сервисы	«dns-tcp»	53,853	DNS + DNS-over-TLS
Базовые сетевые сервисы	«dns-udp»	53	DNS
Базовые сетевые сервисы	«dhcp-udp»	67,68	DHCP (IPv4)
Базовые сетевые сервисы	«dhcpv6-udp»	547	DHCPv6
Базовые сетевые сервисы	«web-tcp»	80,443	HTTP/HTTPS

Базовые сетевые сервисы	«web-udp»	443	HTTP/3 (QUIC)
Базовые сетевые сервисы	«pxe-udp»	69,4011	Сетевая загрузка: tftp, proxy-dhcp
Почтовые сервисы	«smtp-tcp»	25,465,587	SMTP
Почтовые сервисы	«imap-pop3-tcp»	110,143,993,995	IMAP/IMAPS, POP3/POP3S
Каталоги, аутентификация	«ldap-tcp»	389,636	LDAP/LDAPS
Каталоги, аутентификация	«kerberos-tcp»	88,464	Kerberos
Каталоги, аутентификация	«kerberos-udp»	88,464	Kerberos
Каталоги, аутентификация	«freeipa-tcp»	80,88,389,443,464,636	FreeIPA
Каталоги, аутентификация	«freeipa-udp»	88,123,464	FreeIPA
Файловые шары и доступ	«nfs-tcp»	2049	NFS
Файловые шары и доступ	«nfs-udp»	2049	NFS
Файловые шары и доступ	«smb-tcp»	139,445	SMB/Samba
Файловые шары и доступ	«smb-udp»	137,138	SMB/Samba (NetBIOS)
Файловые шары и доступ	«ftp-tcp»	21	FTP
Файловые шары и доступ	«rsync-tcp»	873	Синхронизация файлов
Файловые шары и доступ	«rsync-udp»	873	Синхронизация файлов
Удалённое управление	«ssh-tcp»	22	SSH
Удалённое управление	«rdp-tcp»	3389	RDP
Удалённое управление	«openvpn-udp»	1194	OpenVPN
Удалённое управление	«wireguard-udp»	51820	WireGuard
Удалённое управление	«squid-tcp»	3128	Прокси (squid)
Синхронизация времени и логи	«ntp-udp»	123	NTP
Синхронизация времени и логи	«syslog-udp»	514,6514	Syslog
Синхронизация времени и логи	«syslog-tcp»	6514	Syslog (TLS)
Базы данных	«postgresql-tcp»	5432	PostgreSQL
Базы данных	«mysql-tcp»	3306	MySQL/MariaDB

Мониторинг и управление	«snmp-udp»	161,162	SNMP + snmptrap
Мониторинг и управление	«zabbix-tcp»	10050,10051	Zabbix агент/сервер
Мониторинг и управление	«prometheus-tcp»	9090	Prometheus
Кластеризация и HA	«corosync-udp»	5404,5405-5412	Corosync
Кластеризация и HA	«pacemaker-tcp»	2224,3121,5403,9929,21064	Pacemaker

Поле «Источник/назначение трафика» аналогично подмешивает готовые шаблоны наборов адресов:

Имя шаблона	Адреса
«rfc1918-private»	10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
«loopback»	127.0.0.0/8
«link-local»	169.254.0.0/16
«cloud-metadata»	169.254.169.254/32
«multicast»	224.0.0.0/4
«any-ipv4»	0.0.0.0/0
«public-dns-resolvers»	8.8.8.8, 8.8.4.4, 1.1.1.1, 1.0.0.1, 77.88.8.8, 77.88.8.1

Выбор готового шаблона (порта или адреса) не создаёт постоянный объект группы/набора – используется его состав только для этого правила. Материализовать шаблон как переиспользуемый объект можно отдельно – кнопкой «Готовые шаблоны» на странице «Группы портов» или «Наборы адресов» (создаёт сразу все отмеченные, пропуская уже существующие по имени).

Группы портов и наборы адресов, созданные как самостоятельные объекты, персистентны на диске («/var/lib/ovirt-engine/ovn-firewall-local-store.json») – переживают рестарт сервиса. Это локальная абстракция плагина, не сущности OVN – сделано намеренно, чтобы не давать бэкенду прямой доступ к OVN NBDB.

5.4. Логические сети

«+ Создать сеть» – название, тип (Overlay/VLAN/Flat), для VLAN/Flat – Physical Network и (для VLAN) VLAN ID, MTU (по умолчанию 1442, рекомендуется для Geneve-туннелей OVN), Port Security.

Сети, используемые как External Gateway какого-либо маршрутизатора, помечены в таблице бейджем «Внешний шлюз» (имя маршрутизатора – во всплывающей подсказке).

Значок 🔗 – назначить группу безопасности по умолчанию для всех ВМ этой сети (через её vNIC-профили в oVirt Engine). Изменение (или снятие) группы применяется немедленно – в том числе к уже включённым, работающим ВМ, а не только к тем, что подключатся к сети позже: плагин обновляет не только advisory-свойство «SecurityGroups» vNIC-профиля (это применилось бы только при следующем подключении/переподключении сетевого интерфейса), но и напрямую проталкивает итоговый список групп на уже существующий OVN-порт каждой реально подключённой ВМ («PUT .../ports/<id>») – подтверждено вживую: смена членства в группе безопасности отражается на трафике сразу, без перезагрузки ВМ или переподключения сетевого адаптера.

Значок 🔌 (только для vlan/flat-сетей с заданным Physical Network) – «Настроить физическую привязку», см. раздел 7.

5.5. Маршрутизаторы

«+ Создать маршрутизатор» – только имя (и опционально описание). В панели маршрутизатора: «⊕ Шлюз» – назначить/снять внешний шлюз (выбор сети-шлюза, подсети, IP-адреса); список подключённых interior-сетей с возможностью отключить (✕) или сменить маршрутизатор (⇄) для каждой подсети отдельно.

5.6. Подсети

Отдельная страница со списком всех подсетей (по всем сетям сразу).

«+ Создать подсеть» – сеть-владелец, CIDR, шлюз (см. предупреждение ниже), DNS-сервер (необязательно), IP-версия. DHCP всегда включён – «ovirt-provider-ovn» не поддерживает «enable_dhcp=false» ни при каких условиях, поэтому этой настройки в форме нет вообще.

DNS-сервер – поле одно (не список через запятую), хотя формально «dns_nameservers» у провайдера это список: проверено вживую и в исходниках провайдера («neutron_api_mappers.py»: «dns = dnses[0] if dnses else None») – «ovirt-provider-ovn» реально использует только первый элемент списка, любые дополнительные значения молча отбрасываются. Указанный адрес раздаётся гостям через DHCP автоматически.

Внимание: если шлюз («gateway_ip») не указан при создании подсети, провайдер не назначает его автоматически – подсеть без шлюза создастся успешно, но её нельзя будет подключить к маршрутизатору (ни как interior-интерфейс, ни как external gateway) – провайдер вернёт ошибку «Attaching by subnet requires the subnet to have a default gateway specified». В таблице подсети без шлюза помечены предупреждением. Провайдер разрешает только одну подсеть на сеть – при попытке создать вторую кнопка на сети автоматически превращается в «✎ Подсеть» (редактирование уже существующей).

5.7. Журнал событий (аудит)

Журнал всех действий пользователей с плагином – создание/изменение/удаление любых объектов. Каждая запись подписана HMAC-SHA256 для защиты от модификации

(требование Приказа ФСТЭК №187). Хранение – 365 дней (ротация через logrotate). Есть поиск, фильтр по уровню (критично/предупреждение/информация) и диапазону дат, экспорт в CSV.

5.8. Контроль соответствия

Кнопка «Запустить проверку» – автоматическая проверка конфигурации плагина и его окружения на соответствие ряду требований Приказа ФСТЭК России №187 (ведение журнала, неотказуемость записей, принцип минимальных привилегий, white-list модель и т.д.), с наглядным отчётом «выполнено/нарушено».

5.9. Инциденты

Ручной учёт инцидентов информационной безопасности (создание записи, изменение статуса: открыт/расследуется/решён/закрыт). Персистентны на диске, без ограничения на объём хранимой истории (в отличие от журнала аудита, где действует лимит на число записей в памяти).

5.10. Настройки → Подключение к OVN

См. раздел 3.2 – веб-форма для задания логина/пароля сервисной учётной записи «ovirt-provider-ovn», с кнопкой проверки соединения.

5.11. Настройки → ГосСОПКА

Настройка автоматической отправки событий уровня warning/critical (удаления объектов, инциденты, экспорт журнала – обычные создания правил не отправляются, чтобы не засорять канал) в ГосСОПКА: два режима – Syslog (RFC5424, UDP) или REST API. Кнопка «Отправить тестовое событие» проверяет канал независимо от переключателя включения автоматической отправки.

6. Первые шаги с нуля – пошаговый сценарий

Ниже – короткий путь от только что установленного плагина (шаги 3.1–3.3 выполнены) до работающей сети с правилом ACL.

Шаг 1:
Создать логическую сеть.
Логические сети → «+ Создать сеть». Для первого знакомства проще всего создать сеть типа Overlay (по умолчанию) – она не требует физической привязки к хосту (в отличие от VLAN/Flat, см. раздел 7, если сразу нужен выход наружу). Название: например «int-web». MTU: оставить 1442.

Шаг 2:
Создать подсеть.
На той же сети (или через панель деталей на Топологии) – «+ Подсеть». Сеть: «int-web».

CIDR: например, «10.10.10.0/24». Шлюз: обязательно указать (например «10.10.10.1») – см. предупреждение в разделе 5.6.

Шаг 3.
Создать маршрутизатор и подключить подсеть.
Маршрутизаторы → «+ Создать маршрутизатор» – только имя, например «router-web». Подключить подсеть из шага 2: Логические сети → кнопка «⇄ К роутеру» на сети «int-web» (либо через панель деталей на Топологии). Кнопка неактивна, если сеть уже подключена к другому маршрутизатору или уже используется как внешний шлюз – это ожидаемое поведение. Проверка: на Топологии должна появиться зелёная сплошная линия маршрутизатор → сеть.

Шаг 4.
Создать группу безопасности и правило.
Группы и правила ACL → «+ Создать группу» – например «web-servers». Открыть группу → «+ Добавить правило»: направление – Входящий (ingress); шаблон/группа портов – выбрать готовый шаблон «web-tcp» (80,443), протокол подставится автоматически; источник/назначение трафика – для теста Любой («0.0.0.0/0»), либо готовый шаблон набора адресов (например «rfc1918-private»).

Шаг 5.
Назначить группу сети по умолчанию.
Логические сети → значок 🔗 на сети «int-web» → выбрать «web-servers». Группа станет дефолтной для всех ВМ, подключённых к этой сети через её vNIC-профиль в Engine.

Шаг 6.
Итоговая проверка.
Топология сети → диаграмма должна показывать маршрутизатор → сеть зелёной линией; кнопка «Проверить конфигурацию» → «overall: pass». Журнал событий → все действия шагов 1–5 должны быть видны.

7. Публикация ВМ во внешнюю (физическую) сеть

Если требуется реальный доступ снаружи (не просто overlay-сеть внутри OVN) – сеть должна быть типа «vlan» или «flat», с привязкой к NIC физического хоста. «ovirt-provider-ovn» не делает NAT (ни SNAT, ни floating IP/DNAT) – поэтому есть два принципиально разных варианта архитектуры:

Вариант А - ВМ прямо на физической сети (без маршрутизатора):
ВМ получает адрес напрямую из физического сегмента. Подходит, когда нет причины изолировать ВМ от физической сети отдельным сегментом. Нужен только шаг 7.1 ниже (создать vlan/flat-сеть), маршрутизатор не требуется.

Вариант Б – ВМ за маршрутизатором (внешний шлюз):
Подходит когда нужно объединить/изолировать несколько внутренних (overlay) сетей за одной точкой выхода. Внутренняя подсеть должна быть реальным маршрутизируемым блоком – приватный диапазон «для NAT» здесь не будет доступен снаружи, транслировать его нечем. Сетевые администраторы должны отдельно прописать маршрут на подсеть ВМ через IP шлюза.

Далее – процедура физической привязки vlan/flat-сети к NIC хоста (нужна в обоих вариантах для самой vlan/flat-сети).

7.1. Создать OVN-сеть

Логические сети → «+ Создать сеть»: тип VLAN (с тегом) или Flat (без тега); Physical Network – произвольное имя-метка (например «phus-ovn-1»), которое должно совпадать с именем несущей сети, которая будет создана на следующем шаге.

7.1.1. VLAN или Flat – как выбрать правильно

Это решение зависит от того, где физически на хосте уже происходит разделение по VLAN – на уровне ядра (шаг 7.3, интерфейс вида «bond0.2165») или нет:

- Если порт, который вы подключаете к несущему мосту в шаге 7.3 – это сырой физический адаптер/бонд без собственного VLAN-тега (например «bond0» напрямую, весь порт коммутатора в access-режиме под эту сеть) – используйте тип VLAN с нужным «segmentation_id». OVN сам добавит/снимет тег на выходе – ровно один раз, корректно.
- Если порт, который вы подключаете в шаге 7.3 – это уже готовый VLAN-псевдоинтерфейс ядра («bond0.2165», «ens192.100» и т.п., созданный отдельно от OVN – например, потому что тот же бонд используется ещё для management/других сетей и физически нельзя отдать OVS целиком) – используйте тип Flat, без «segmentation_id». Разделение по VLAN в этом случае уже полностью выполняет ядро (интерфейс «bond0.2165» сам снимает/добавляет тег 2165 при переходе трафика между «bond0» и «bond0.2165») – OVN здесь тегировать не должен.

Реальный инцидент (сеть «net-in», Physical Network «vL2165», порт «bond0.2165» на нескольких хостах): сеть изначально была создана как VLAN с «segmentation_id=2165». Внешне всё выглядело настроенным правильно – привязка физической сети прошла без ошибок, мост и порт были на месте, patch-порт к «br-int» появился, «ovn-bridge-mappings» пропагировался в Southbound DB. Но ВМ за маршрутизатором с таким External Gateway не могла достучаться наружу вообще – маршрутизатор не отвечал даже на ARP по своему собственному внешнему IP.

Причина: у логического порта «localnet» этой сети («ovn-nbctl find Logical_Switch_Port type=localnet») стоял «tag=2165» – OVN добавлял VLAN-тег 2165 на кадр при выходе из «br-int» в несущий мост. А порт «bond0.2165», через который кадр уходил дальше на

физику – это уже готовый VLAN-псевдоинтерфейс ядра (802.1Q, id 2165) поверх разделяемого бонда «bond0» (на этом же бонде висели и другие VLAN – management-сеть и другие тенантские сети, поэтому отдать «bond0» целиком в OVS без тегов было нельзя). В результате кадр уходил на провод с двойным тегом VLAN 2165 (тег от OVN + тег от ядра) – коммутатор либо отбрасывал такой кадр, либо не мог разобрать его как IP-трафик после снятия внешнего тега (EtherType оставался «0x8100» – «ещё один VLAN» – вместо «0x0800»/IPv4).

Диагностика – проверить, что реально стоит на localnet-порту сети:

- ovn-nbctl find Logical_Switch_Port type=localnet # смотреть поле "tag" у записи с нужным "options: {network_name=...}"

Если «tag» непустой (совпадает с VLAN самой сети) – а порт в шаге 7.3 это уже VLAN-псевдоинтерфейс ядра – вы получите ровно эту проблему.

Исправление:

Сменить тип сети с VLAN на Flat (сама привязка к «bond0.2165»/несущему мосту не меняется, физический уровень трогать не нужно). Провайдер не позволяет менять «provider:network_type» у существующей сети – придётся пересоздать:

1) Отключить External Gateway у всех маршрутизаторов, использующих эту сеть (панель роутера → «✕ Убрать шлюз»), запомнив их внешние IP («external_fixed_ips») – они понадобятся на шаге 5.
2) Удалить подсеть этой сети, затем саму сеть.
3) Создать сеть заново с тем же именем и тем же Physical Network, но тип Flat (без VLAN ID)
4) Создать подсеть с теми же параметрами (CIDR/шлюз);
5) Вернуть External Gateway каждому маршрутизатору на новую сеть с теми же «external_fixed_ips», что были до пересоздания – тогда адресация для остальных потребителей этой сети не изменится.

После пересоздания проверить: у localnet-порта новой сети поле «tag» должно быть пустым (см. диагностику выше).

7.2. Настроить физическую привязку (шаги 1–5, кнопка в интерфейсе)

На созданной сети (значок 🔌 в таблице «Логические сети» или кнопка «🔌 Физ. привязка» в панели деталей сети на Топологии) открыть «Настроить физическую привязку».

Сразу при открытии окна плагин показывает блок «Уже настроено» – текущее состояние привязки для введённого имени физической сети (только чтение, ничего не меняет): существует ли несущая сеть, в каких кластерах она числится, переведена ли в Non-VM Network, и таблицу – на каких хостах она реально привязана, к какому адаптеру и с каким VLAN. Если сеть с таким именем ещё не существует – блок сообщает, что привязки нет и форма ниже настроит её с нуля. Это позволяет перед повторным открытием модалки на

уже настроенной сети сразу увидеть, что и где сделано, не пролистывая Setup Host Networks по каждому хосту вручную.

Дальше заполняются поля для проверки/настройки (новой или дополнительной привязки к ещё одному хосту той же сети):

- Физическая сеть: то же имя, что в шаге 7.1 (например «phus-ovn-1»);
- VLAN: тот же тег, что задан у OVN-сети (пусто – для Flat);
- Кластер: выбрать целевой кластер;
- Хосты и адаптеры: отметить каждый хост кластера, где должна быть доступна эта сеть, и выбрать сетевой адаптер (базовый физический NIC или bond – не уже существующий тегированный VLAN-псевдоинтерфейс вида «bond0.2164», если такой уже есть от других сетей).

Кнопка «Проверить» выполняет (без единого изменения):

- Занято ли указанное имя несущей сети OVN-синхронизированной сетью (в этом случае – явная ошибка, нужно выбрать другое имя);
- Если несущая сеть с таким именем уже существует – используется ли она реально виртуальными машинами (если да – автоматическая конвертация в Non-VM Network заблокирована как небезопасная, предлагается создать новую несущую сеть на другом VLAN);
- Свободен ли выбранный VLAN на выбранном адаптере каждого хоста (при конфликте – точное имя конфликтующей сети и число ВМ на ней, если есть).

Кнопка «Настроить» выполняет по очереди, при условии что проверка не нашла ни одной ошибки:

1) Создание несущей (обычной, не external-provider) сети oVirt с указанным именем в датацентре выбранного кластера.
2) Добавление этой сети в выбранный кластер.
3) Перевод сети в режим Non-VM Network – обязательный шаг: если оставить сеть как VM Network (значение по умолчанию в oVirt), VDSM создаёт для неё обычный Linux-мост и включает VLAN-интерфейс НИКа внутрь этого моста – интерфейс становится недоступен для последующей привязки к OVS-мосту в шаге 7.3. Симптом обманчиво похож на «всё настроено правильно»: сеть в кластере становится «operational», а реального пути наружу всё равно нет – отличить можно только через «ip a» на самом хосте (в выводе не должно быть «master <имя-моста>»).
4) Установка VLAN-тега несущей сети (если задан).
5) Привязка несущей сети к выбранному NIC на каждом отмеченном хосте через Setup Host Networks, с проверкой связности (изменение автоматически откатывается, если рвёт связь Engine с хостом).

После успешного выполнения шагов 1–5 плагин выводит готовые команды для последнего, шестого шага – с реальными именами хоста, интерфейса и предлагаемого имени моста (не общий шаблон).

7.3. Завершающий шаг – вручную по SSH на каждом хосте

Этот шаг плагин намеренно не выполняет сам – потребовался бы SSH/root-доступа бэкенда к каждому гипервизору, что противоречит принципу минимальных привилегий (см. раздел 1). Выполнить команды, которые вывел мастер после шага 7.2 (общий вид, «ens192.2164»/»br-phus-ovn-1» – примеры реальных имён из тестового стенда):

- ssh -i /etc/pki/ovirt-engine/keys/engine_id_rsa root@<host-fqdn>
- ovs-vsctl add-br br-phus-ovn-1
- ovs-vsctl add-port br-phus-ovn-1 ens192.2164
- ovs-vsctl set open . external-ids:ovn-bridge-mappings=phus-ovn-1:br-phus-ovn-1

Название моста произвольное, важно только совпадение имени слева от «:» в «ovn-bridge-mappings» с Physical Network создаваемой OVN-сети. Это изменение аддитивное – новый мост и новый порт, не затрагивает «br-int» (интеграционный мост OVN) или уже работающие ВМ на этом хосте. Повторить на каждом хосте кластера, где может запускаться ВМ этой сети или работать маршрутизатор.

Обычный root-пароль сервера Engine не подходит для SSH на гипервизорные хосты, несмотря на схожесть окружения – нужен именно ключ движка («/etc/pki/ovirt-engine/keys/engine_id_rsa»), команда запускается с самого сервера Engine.

7.4. Проверка, что шаг 7.3 реально дошёл до OVN

- ovn-sbctl --columns=name,hostname,other_config list Chassis

Смотреть поле «other_config», ключ «ovn-bridge-mappings» – не «external_ids» (в этом поле маппинг никогда не появляется; при проверке легко перепутать колонку и ошибочно решить, что распространение не сработало, хотя на самом деле всё в порядке).

Если нужного хоста вообще нет в списке Chassis – значит на нём не запущен или не подключён «ovn-controller».

Проверить:

- systemctl status ovn-controller
- systemctl enable --now ovn-controller # если служба не запущена

Без работающего «ovn-controller» хост не регистрируется как chassis вообще – настройки OVS (шаг 7.3) в этом случае никакого эффекта не дают, даже если выполнены правильно.

7.5. Переживёт ли это перезагрузку и последующую реконфигурацию хоста

Обычную перезагрузку хоста – да. OVS хранит мосты/порты/external-ids в собственной локальной базе («conf.db»), которую служба «openvswitch» восстанавливает сама при каждом старте (так же переживает перезагрузку и «br-int»). VLAN-интерфейс тоже

переживёт – он создаётся VDSM как часть официальной привязки сети к хосту (шаг 7.2, пункт 5), персистентен по определению.

Реконфигурацию сети хоста через Setup Host Networks – только частично, и это зависит от режима работы кластера («switch_type»). В кластерах с «switch_type=legacy» мостами OVS для OVN управляет только ручной шаг 7.3 – VDSM про них вообще ничего не знает (в legacy-режиме VDSM управляет только Linux-мостами обычных VM-сетей, не OVS). **Из этого следует:**

- -Если позже отвязать несущую сеть от хоста или сменить VLAN-тег через Administration Portal – VDSM снесёт VLAN-интерфейс, а OVS-мост останется с портом, ссылающимся на уже не существующий интерфейс (осиротевший, до ручной чистки «ovs-vsctl del-port»);
- Изменение VLAN-тега несущей сети в будущем нужно также вручную отразить в OVS («ovs-vsctl del-port»/»add-port» с новым именем интерфейса) – Engine это не сделает и не подскажет;
- Реконфигурация других сетей/NIC того же хоста эту привязку не затрагивает;
- В кластерах с «switch_type=OVS» поведение может отличаться (VDSM сам управляет OVS-мостами) – привязку в этом режиме стоит проверять заново перед использованием.

7.6. Подключение сети к маршрутизатору как внешний шлюз

Если выбран вариант Б (ВМ за маршрутизатором) – созданную и физически привязанную сеть нужно подключить не как обычную interior-сеть, а именно как External Gateway: панель маршрутизатора на Топологии → «⊕ Шлюз» → выбрать эту сеть, подсеть и IP-адрес шлюза. Source NAT («enable_snat») провайдером не поддерживается – всегда отправляется «false»; реальный внешний доступ обеспечивается только прямой физической связностью подсети (см. вариант Б выше).

7.7. NAT – вручную поверх уже существующего маршрутизатора

Плагин и «ovirt-provider-ovn» NAT (ни SNAT, ни floating IP/DNAT) не делают вообще – это ограничение самого API провайдера (Neutron-подобный REST, где такой функциональности нет), а не недоработка плагина. Кнопки, которая реально записывает правило в NBDB, в интерфейсе нет и не появится (бэкенд не имеет прав на запись, см. раздел 1) – есть просмотр уже настроенных правил и мастер, который проверяет параметры и готовит команды для ручного выполнения (см. ниже).

Важно: логический маршрутизатор, который создаёт плагин – это обычный OVN «Logical_Router» в Northbound DB, причём его «id» совпадает с UUID соответствующего «Logical_Router» напрямую (проверено вживую: «id» роутера из списка «Логические маршрутизаторы» и «_uuid» объекта в выводе «ovn-nbctl list Logical_Router» – одно и то же значение; это отличается от логических сетей, чей «id» с объектами Engine/OVN не совпадает никогда, см. раздел 5.4). Такие роутеры умеют NAT нативно, средствами самого OVN (таблица «nat», команда «ovn-nbctl lr-nat-add») – это не отдельный «тип» роутера,

который нужно выбирать при создании, а дополнительная конфигурация поверх уже существующего маршрутизатора (в том числе с уже настроенным через раздел 7.6 External Gateway). Отдельная ВМ-шлюз с iptables MASQUERADE (простой вариант для изолированных приватных подсетей за роутером) не нужна, если подходит вариант ниже.

Просмотр уже настроенных правил – кнопка «🔒 NAT» (значок в таблице «Логические маршрутизаторы» или кнопка в панели роутера на Топологии) показывает текущие NAT-правила этого роутера: только чтение, читается напрямую из OVN Northbound DB через единственную узко ограниченную read-only команду («GET /routers/<id>/nat-rules» → «sudo ovn-nbctl lr-nat-list <id>»). Права выдаются самим RPM-пакетом при установке («/etc/sudoers.d/keyvirt-ovn-firewall-nat-readonly») – пользователю «ovirt» разрешена ровно эта одна команда, никаких «lr-nat-add»/»lr-nat-del»/»set». Ничего не добавляет и не меняет – только показывает. Таблица правил содержит и готовую, скопированную команду для ручного удаления каждого конкретного правила («ovn-nbctl lr-nat-del <router> <type> <logical_ip>«) – раньше такая подсказка была только для добавления через мастер, для удаления её приходилось составлять самостоятельно.

Мастер настройки – кнопка «⚙️ Мастер NAT» рядом с «🔒 NAT» (там же – таблица «Логические маршрутизаторы» и панель роутера на Топологии). Плагин по-прежнему не пишет в NBDB – мастер только проверяет введённые параметры против уже существующей конфигурации и формирует готовые команды, которые нужно выполнить вручную **(см. процедуру ниже):**

- Тип (SNAT / DNAT+SNAT), внешний IP (только из уже реально настроенных External Gateway этого роутера – свободный ввод не допускается намеренно, см. объяснение в шаге 3.1 процедуры ниже), внутренняя сеть (из реальных interior-подключений роутера) и внутренний адрес/подсеть;
  Хосты (gateway chassis) – список с чекбоксами (как в «Настроить физическую привязку»), а не одиночный выпадающий список: можно сразу отметить несколько хостов для отказоустойчивости – приоритет назначается по порядку в списке (первый отмеченный – приоритет 10, второй – 20 и т.д.). При открытии окна список сразу отмечает хосты, уже назначенные этому роутеру (новый read-only эндпоинт «GET /routers/<id>/nat-wizard/gateway-chassis», та же команда «ovn-nbctl lrp-get-gateway-chassis», тот же sudoers-файл) – открывая мастер на уже настроенном роутере, сразу видно текущее состояние, а не пустой список;
  Проверка: настроен ли External Gateway, совпадает ли внешний IP с реальным адресом шлюза, принадлежит ли внутренний адрес/подсеть реальной interior-сети этого роутера, что именно изменится в назначении gateway chassis при выбранном наборе хостов, не существует ли уже такое же правило NAT;
  Результат – набор готовых команд «ovn-nbctl» (с уже подставленными реальными id/именами), которые нужно скопировать и выполнить вручную от root там, где доступна OVN Northbound DB. Команды формируются в обе стороны: «lrp-set-gateway-chassis» – для хостов, которых не хватает или у которых изменился приоритет; «lrp-del-gateway-chassis» – для хостов, которые были назначены раньше, но сняты с чекбокса в этом открытии мастера. Если ни один хост ещё не назначен – необходимость выбрать

хотя бы один явно отмечается в результатах проверки (порядок команд важен – см. процедуру ниже).

На хостах с SELinux в режиме enforcing (Platform V SberLinux, RHEL 8+) одного sudoers-файла недостаточно – mod_wsgi-процесс плагина работает в ограниченном домене «httpd_t», которому по умолчанию не разрешено то, что нужно «sudo» для успешного выполнения. **Пакет дополнительно ставит и загружает (полностью автоматически, в «%post»):**

- - SELinux-модуль «keyvirt_ovn_nat_sudo» (собран из исходника «.te», входящего в RPM) – разрешает домену «httpd_t» «capability sys_resource» и «audit_write» (нужны «sudo» самому), плюс «unix_stream_socket connectto» к домену «unconfined_service_t» и «sock_file write» на «var_run_t» (нужны, чтобы «ovn-nbctl» мог подключиться к сокету «/var/run/ovn/ovnnb_db.sock» – в этом окружении его обслуживает процесс именно в этом домене);
- Штатный SELinux-булев «httpd_mod_auth_pam» – даёт вспомогательному процессу «unix_chkpwd» (PAM-проверка учётной записи внутри «sudo», выполняется при каждом вызове «sudo», даже с NOPASSWD) корректно перейти в уже существующий, специально для этого предназначенный домен «chkpwd_t» вместо того, чтобы давать домену «httpd_t» прямой доступ на чтение «/etc/shadow».

Важно: все эти правила действуют на весь домен «httpd_t» – то есть на любой другой vhost/плагин, работающий под тем же Apache, а не только на keyvirt-ovn-firewall-plugin. Устанавливать эти правила стоит осознанно; при удалении пакета SELinux-модуль удаляется автоматически, булев «httpd_mod_auth_pam» не трогается (это общий флаг, как и «httpd_can_network_connect» – другие компоненты могут на него полагаться).

Добавление/изменение правил – по-прежнему полностью вручную:

1) Убедиться, что у роутера уже настроен External Gateway (раздел 7.6) – трансляция физически идёт через тот же gateway-порт.
2) Взять «id» роутера прямо из плагина (таблица «Логические маршрутизаторы» или панель роутера на Топологии) – он же и есть UUID нужного «Logical_Router» (см. врезку выше), можно свериться:

- ovn-nbctl list Logical_Router

3) Назначить gateway chassis порту роутера, смотрящему во внешнюю сеть – обязательный шаг, без которого следующее правило NAT создастся в базе, но не будет работать вообще (проверено вживую: «ovn-trace --ct=new» до этого шага не показывает никакой стадии «lr_out_snat» в конвейере – трафик просто минует NAT молча, без ошибок). Сначала найти имя нужного порта (тот, чьи «networks» совпадают с внешней подсетью):

- ovn-nbctl show <router>

4) Затем назначить хост, который будет централизованно обрабатывать NAT для этого роутера (в распределённой топологии OVN трансляция адресов требует consistent conntrack-состояния – оно должно жить на ОДНОМ хосте, поэтому просто

«распределённая» маршрутизация, которая и так работает без этого шага, для NAT недостаточна):

- ovn-nbctl lrp-set-gateway-chassis <lrp-имя-порта> <хост-полное-имя> 10

5) Проверить, что назначилось:

- ovn-nbctl lrp-get-gateway-chassis <lrp-имя-порта>

6) Добавить правило NAT – выполняется на узле, где реально доступна OVN Northbound DB (как правило – тот же хост, где «ovn-northd»/»ovirt-provider-ovn»; если БД не локальная – добавить к командам ниже флаг «--db=<connection-string><«):

- # Source NAT – все адреса внутренней подсети "прячутся" за один
  # внешний адрес (аналог MASQUERADE):
  ovn-nbctl lr-nat-add <router> snat <external-ip> <internal-subnet-cidr>
- # DNAT+SNAT – проброс конкретного внешнего адреса на конкретную ВМ
  # (аналог "проброса порта"/floating IP), трафик в обе стороны:
  ovn-nbctl lr-nat-add <router> dnat_and_snat <external-ip> <internal-vm-ip>

7) Проверить результат – вручную командой или кнопкой « 🔒 NAT» в плагине (обновит список):

- ovn-nbctl lr-nat-list <router>

8) Убедиться, что правило реально работает (не только создано в базе) – самый надёжный способ, без доступа к консоли гостя:

- ovn-trace --ct=new '<datapath-имя-логической-сети-ВМ>' \
  'inport == "<port-id-ВМ>" && eth.src==<mac-ВМ> && eth.dst==<mac-шлюза-подсети> && ip4.src==<IP-ВМ> && ip4.dst==<любой-внешний-IP> && ip.ttl==64 && icmp4.type==8 && icmp4.code==0'

9) В выводе должна появиться стадия «lr_out_snat» с «ct_snat_in_czone(<внешний-IP>)» – если её нет, gateway chassis не назначен или назначен не на тот порт.

Ограничения этого способа (важно понимать перед использованием):

- Плагин может только показать уже настроенные правила (кнопка « 🔒 NAT») – добавление, изменение и удаление правил по-прежнему только вручную командами выше, в интерфейсе этого нет;
- Не переживает удаление/пересоздание роутера через плагин – новый «Logical_Router» создаётся провайдером с нуля, без ваших nat-записей И без назначения gateway chassis;

после любого пересоздания роутера оба шага («lrp-set-gateway-chassis» и «lr-nat-add») нужно повторить вручную;

- Gateway chassis – это конкретный хост, который централизованно обрабатывает NAT для этого роутера. Если назначен только один хост и он выйдет из строя/уйдёт на обслуживание – NAT (а с ним и весь внешний доступ через SNAT) для этого роутера перестанет работать, пока хост не вернётся или не будет назначен другой (обычная маршрутизация без NAT при этом продолжит работать в распределённом режиме на других хостах). Для отказоустойчивости можно назначить несколько хостов разными вызовами «lrp-set-gateway-chassis» с разным приоритетом (меньшее число – выше приоритет) – тогда при отказе текущего активного OVN автоматически переключится на следующий по приоритету;
- Сделано так намеренно: бэкенд плагина не имеет прав на запись в OVN Northbound DB и не может изменить NAT-конфигурацию сам (принцип минимальных привилегий, ФСТЭК №187, см. раздел 1) – эта операция всегда остаётся полностью ручной, как и шаг 6 физической привязки (раздел 7.3); единственное разрешённое обращение к NBDB – одна read-only команда для кнопки просмотра.

7.8. Ограничение исходящего трафика (egress) – вручную поверх группы безопасности

Как объяснено в разделе 11 – обычные группы безопасности не могут ограничить исходящий (egress) трафик ВМ: провайдер сам автоматически вешает на каждую группу безусловное ACL "allow all egress", и Neutron Security Group Rules API в принципе не поддерживает запрещающие действия. Единственный способ реально запретить egress (или ingress, минуя логику групп безопасности) – создать отдельную OVN Port_Group с явным ACL действия «drop» напрямую в Northbound DB, в обход API провайдера. Бэкенд плагина не имеет прав на запись в NBDB (принцип минимальных привилегий) – эта операция всегда полностью ручная.

Кнопка «⛔ Egress-запрет» (страница «Группы безопасности и правила ACL») открывает мастер: имя новой группы, направление (исходящий и/или входящий), список ВМ (чекбоксы) – плагин сам резолвит выбранные ВМ в их реальные id портов OVN. Кнопка «Проверить и сформировать команды» проверяет параметры (имя не занято зарезервированными "Default"/"DropAll", выбрана хотя бы одна ВМ и хотя бы одно направление, текущее состояние группы, если она уже существует) и формирует готовые команды «ovn-nbctl» – выполнить нужно вручную от root там, где доступна OVN Northbound DB:

- ovn-nbctl pg-add <имя-группы> <port-id-1> <port-id-2> ...
- ovn-nbctl --type=port-group acl-add <имя-группы> from-lport 1002 "inport == @<имя-группы> && ip" drop
- ovn-nbctl --type=port-group acl-add <имя-группы> to-lport 1002 "outport == @<имя-группы> && ip" drop

Важно – приоритет ACL должен быть строго выше 1001 (порог автоматического allow-all-egress провайдера, раздел 11) – иначе правило создастся, но не будет иметь эффекта. Плагин всегда подставляет приоритет 1002 – этого достаточно, чтобы перебить автоматический allow, но оставляет запас для более специфичных allow-исключений (приоритет 1003+), если понадобится разрешить что-то конкретное поверх общего запрета – такие исключения мастер не формирует, добавляются вручную той же командой «acl-add» с действием «allow-related».

Если группа уже существует (повторное открытие мастера с другим набором ВМ) – вместо «pg-add» формируется:

- ovn-nbctl pg-set-ports <имя-группы> <port-id-1> <port-id-2> ...

Проверка – кнопка «Проверить конфигурацию» на Топологии теперь дополнительно ищет все пользовательские (не служебную DropAll) группы портов с drop-ACL и предупреждает, если у какой-то из них приоритет ≤ 1001 (правило создано, но неэффективно) – read-only.

Убрать созданную группу целиком (со всеми её ACL) – тоже вручную:

- ovn-nbctl pg-del <имя-группы>

Для чтения текущего состояния (кнопка/мастер/health-check) пакет выдаёт пользователю «ovirt» ещё 2 read-only sudo-команды в дополнение к двум уже существующим для NAT (раздел 7.7, тот же sudoers-файл «/etc/sudoers.d/keyvirt-ovn-firewall-nat-readonly»): «ovn-nbctl list Port_Group» и «ovn-nbctl list ACL» – без аргументов, весь список сразу – но по-прежнему строго read-only, никакого «pg-add»/»acl-add»/»pg-set-ports»/»pg-del» пользователю «ovirt» не разрешено.

Ограничения этого способа:

- Эта Port_Group и её ACL полностью вне зоны видимости провайдера – Neutron API ничего о них не знает, поэтому их не показывают ни обычные группы безопасности, ни vNIC-профили ВМ. Управлять можно только вручную через «ovn-nbctl» или косвенно через кнопку «Проверить конфигурацию» (только видит и предупреждает, не правит);
- Если ВМ пересоздать или переподключить NIC – id её OVN-порта меняется, старое членство в группе осиротеет (будет ссылаться на несуществующий порт) – потребуется добавить новый id порта вручную («pg-set-ports» с полным новым списком);
- Изменение через 🔗 обычных групп безопасности (раздел 5.4) эту отдельную Port_Group не затрагивает и не трогает никак – это два независимых, не пересекающихся механизма.

8. Каталоги и файлы

- /usr/share/ovirt-engine/ovn-firewall/
  ovn_firewall_backend.cpython-*.so – скомпилированный backend
  ovn_firewall.wsgi – точка входа mod_wsgi

- /usr/share/ovirt-engine/ui-plugins/
  ovn-firewall.json – дескриптор UI-плагина
  ovn-firewall-resources/
    start.html – bootstrap/content iframe
    plugin.js – весь frontend
    main.css – стили
    fonts/Winston/ – шрифт Winston
- /etc/ovirt-engine/ui-plugins/
  ovn-firewall-config.json – пользовательская конфигурация плагина (включение/выключение)
- /etc/ovirt-engine/
  ovn-firewall.conf – основной конфиг бэкенда (права 640, root:ovirt)
- /etc/httpd/conf.d/
  keyvirt-ovn-firewall.conf – Apache mod_wsgi конфиг
- /usr/lib/systemd/system/
  keyvirt-ovn-firewall.service – резервный systemd-юнит (не нужен при наличии mod_wsgi)
- /etc/sudoers.d/
  keyvirt-ovn-firewall-nat-readonly – права 440, root:root; разрешает пользователю ovirt четыре read-only команды: «ovn-nbctl lr-nat-list» и «ovn-nbctl lrp-get-gateway-chassis» (кнопки «NAT» и «Мастер NAT», раздел 7.7), «ovn-nbctl list Port_Group» и «ovn-nbctl list ACL» (кнопка «Egress-запрет», раздел 7.8), никакой записи
- /usr/share/ovirt-engine/ovn-firewall/selinux/
  keyvirt-ovn-firewall-nat-sudo.te – исходник SELinux-модуля для кнопки «NAT» (раздел 7.7); компилируется и грузится автоматически в %post (checkmodule+semodule), не на этапе сборки RPM
- /etc/logrotate.d/
  keyvirt-ovn-firewall – ротация журнала аудита
- /var/log/ovirt-engine/
  ovn-firewall-audit.log – подписанный журнал аудита (ФСТЭК №187, хранение 365 дней)
- /var/lib/ovirt-engine/
  ovn-firewall-local-store.json – группы портов и наборы адресов (персистентно)
  ovn-firewall-incidents.json – учёт инцидентов
  ovn-firewall-gossopka.json – настройки интеграции с ГосСОПКА (из веб-формы)
  ovn-firewall-credentials.json – логин/пароль сервисной учётной записи (пароль зашифрован AES-256-GCM)
- /run/ovirt-engine/ovn-firewall/
  ovn-firewall.pid – PID-файл (для ротации логов через SIGHUP)

9. Управление сервисом

Основной путь обслуживания запросов – httpd/mod_wsgi, не отдельный процесс.
Перезапуск после изменения конфигурации:

- systemctl reload-or-try-restart httpd

Если по какой-то причине используется резервный systemd-юнит (когда «python3-mod_wsgi» недоступен):

- systemctl start keyvirt-ovn-firewall
- systemctl stop keyvirt-ovn-firewall
- systemctl restart keyvirt-ovn-firewall
- systemctl status keyvirt-ovn-firewal

Проверка версии:

- rpm -q keyvirt-ovn-firewall-plugin

Обновление:

- dnf install -y keyvirt-ovn-firewall-plugin-<новая-версия>-1.<dist>.x86_64.rpm
  ««««

Удаление:

- dnf remove keyvirt-ovn-firewall-plugin

Проверка состояния (health, без авторизации):

- curl -sk http://127.0.0.1/ovn-firewall-api/health | python3 -m json.tool

Просмотр журнала доступа к API плагина:

- tail -f /var/log/httpd/ovn-firewall-access.log

Просмотр журнала аудита (в формате JSON, по одной записи на строку):

- tail -f /var/log/ovirt-engine/ovn-firewall-audit.log

10.Типичные сценарии использования

10.1. Изолированная внутренняя сеть без внешнего доступа

Создать overlay-сеть → подсеть с шлюзом → группу безопасности с нужными правилами (готовые шаблоны портов/адресов) → назначить группу сети. Маршрутизатор и физическая привязка не требуются – сеть живёт только внутри OVN (Geneve-туннели между хостами).

10.2. Публикация сервиса наружу через прямую vlan/flat-сеть

Раздел 7, вариант А: создать vlan/flat-сеть → настроить физическую привязку кнопкой (шаги 1–5) → выполнить шаг 6 вручную по SSH → проверить «other_config» в Southbound DB → подключить к ВМ напрямую (без маршрутизатора) → назначить группу

безопасности с нужными входящими правилами (например «ssh-tcp» для управления, «web-tcp» для веб-сервиса).

10.3. Несколько внутренних сетей за одним внешним шлюзом

Раздел 7, вариант Б: создать одну vlan/flat-сеть с физической привязкой (как в 10.2) → создать маршрутизатор → подключить vlan/flat-сеть как External Gateway (раздел 7.6) → создать одну или несколько overlay-сетей с реально маршрутизируемыми (не приватными «для NAT») подсетями → подключить их к тому же маршрутизатору как interior-интерфейсы → согласовать с сетевыми администраторами маршрут на эти подсети через IP внешнего шлюза.

10.4. Изменение точки подключения существующей сети

Если сеть уже подключена к одному маршрутизатору как interior, а нужно переключить её на другой – использовать кнопку «⇄ Сменить маршрутизатор» непосредственно у подсети в панели маршрутизатора или сети (не кнопку «⇄ К роутеру» у самой сети – та предназначена только для первого, ещё не выполненного подключения; повторный вызов для уже подключённой сети всё равно будет отклонён провайдером с ошибкой «already connected to router»). Плагин сам выполняет отключение от старого маршрутизатора и подключение к новому одним действием.

10.5. Массовая проверка соответствия ФСТЭК №187 перед аудитом

Контроль соответствия → «Запустить проверку» → просмотреть отчёт → Журнал событий → экспорт в CSV за нужный период → при необходимости отправка сводки в ГосСОПКА (Настройки → ГосСОПКА → «Отправить тестовое событие» для предварительной проверки канала).

11.Ограничения

- «ovirt-provider-ovn» не делает NAT (ни SNAT, ни floating IP/DNAT) – поле «enable_snat» всегда отправляется как «false». Плагин умеет показать уже настроенные вручную NAT-правила роутера (кнопка «NAT», read-only) и проверить параметры нового правила плюс сформировать готовые команды (кнопка «⚙ Мастер NAT») – раздел 7.7, но не имеет права записи в NBDB: само добавление/изменение правила – всегда вручную командой «ovn-nbctl lr-nat-add».
- DHCP нельзя отключить ни при создании, ни при редактировании подсети – ограничение самого провайдера, не плагина;
- DNS-сервер подсети – провайдер поддерживает только один адрес (см. раздел 5.6), поэтому и в форме поле одно, не список;
- Разрешена только одна подсеть на одну логическую сеть;
- Подключить сеть сразу к другому маршрутизатору нельзя одним действием – сначала обязательно отключение от текущего (см. 10.4);

- Провайдер не поддерживает «stateful» для групп безопасности и расширенные поля ACL («action», «priority», «log»);
- Egress-трафик ВМ разрешён всегда, независимо от настроенных правил группы безопасности – проверено вживую («ovn-nbctl list ACL» + «ovn-sbctl lflow-list»): при создании любой группы безопасности сам «ovirt-provider-ovn» (не плагин) автоматически добавляет безусловное ACL «allow-related, direction: from-lport, priority: 1001, match: inport==@<группа> && ip4/ip6» – найдено в исходнике провайдера («/usr/share/ovirt-provider-ovn/ovndb/acls.py», «create_default_allow_egress_acls»). Это значит: трафик ОТ ВМ (к другой ВМ, через маршрутизатор, наружу через SNAT) всегда проходит, какие бы правила ни были настроены – правила группы безопасности в этой модели реально ограничивают только ingress (см. раздел 5.3). Средствами плагина (и вообще Neutron Security Group Rules API, который поддерживает только allow-действия) запретить исходящий трафик невозможно;
- Физическая привязка внешней (vlan/flat) сети к NIC хоста автоматизирована только частично: шаги создания несущей сети, добавления в кластер, Non-VM Network, VLAN-тега и привязки к NIC – одной кнопкой; последний шаг (OVS-мост и «ovn-bridge-mappings» на самом хосте) выполняется вручную по SSH – плагин намеренно не получает SSH/root-доступ к гипервизорам;
- Параметр «ALLOWED_ROLES» в конфиге в текущей версии не применяется бэкендом как фактическое ограничение доступа к API – единственная проверка это валидный SSO-токен oVirt, независимо от роли пользователя. Контроль видимости пункта меню в WebAdmin обеспечивается самим дескриптором плагина;
- Группы портов и наборы адресов – локальная абстракция плагина, не сущности OVN (сделано намеренно ради принципа минимальных привилегий, см. раздел 1).

12.Диагностика и устранение неисправностей

12.1. В WebAdmin вообще нет пункта «OVN Firewall»

Причина: страница браузера закэширована, либо плагин только что установлен и WebAdmin ещё не подхватил новый дескриптор.
Решение: обновить страницу браузера (Ctrl+F5). Перезапуск «ovirt-engine» не требуется – движок читает дескрипторы плагинов при каждом HTTP-запросе к странице WebAdmin.

12.2. health = status:ok, но списки сетей/маршрутизаторов пустые, хотя объекты реально существуют

Причина: «OVN_PROVIDER_PASSWORD» не задан или неверен – эндпоинт health проверяет только TCP-доступность порта провайдера, не корректность учётных данных.
Решение: Настройки → Подключение к OVN → задать/перепроверить пароль, нажать «Проверить соединение». Либо проверить топологию: кнопка «Проверить конфигурацию» покажет реальную ошибку авторизации при попытке получить список сетей.

12.3. «Проверить соединение» в Настройках → Подключение к OVN даёт ошибку авторизации

Причина: неверное имя AAA-профиля в имени пользователя – на разных стендах реально встречается и «admin@ovirt@internalsso» (Keycloak), и «admin@ovirt@internal» (legacy AAA) – угадать одним значением нельзя.
Решение: свериться со списком реальных профилей на сервере:

- ls /etc/ovirt-engine/aaa/

И с именем, которое ожидает сам «ovirt-provider-ovn»:

- grep ovirt-admin-user-name /etc/ovirt-provider-ovn/conf.d/10-setup-ovirt-provider-ovn.conf

Указать в поле «Имя пользователя» значение, реально совпадающее с профилем, а не по аналогии с другим стендом.

12.4. 400 «У сети нет vNIC-профилей» при попытке назначить группу безопасности на сеть (значок 🔗)

Причина: сеть провайдера не синхронизирована в собственные объекты oVirt Engine (нет ни одной engine-сети с этим именем и признаком «external_provider=true») – без этого у сети физически нет vNIC-профиля. Обычно вызвано одной из двух причин на стороне Engine (не провайдера и не плагина):

- Engine не доверяет TLS-сертификату «ovirt-provider-ovn» (в журнале «/var/log/ovirt-engine/engine.log» повторяются ошибки «SyncNetworkProviderCommand»: «unable to find valid certification path to requested target»);
- Неверный логин/пароль в самой регистрации провайдера в Engine (Compute → Providers → ovirt-provider-ovn) – это отдельное от «OVN_PROVIDER_USER»/«PASSWORD» плагина место, подверженное той же путанице профилей, что и в п. 12.3.

Диагностика и решение:

- # 1. Проверить журнал:
  grep SyncNetworkProviderCommand /var/log/ovirt-engine/engine.log | tail -20
- # 2. Посмотреть текущую регистрацию провайдера:
  TOKEN=<SSO-токен администратора>
  curl -sk https://<fqdn>/ovirt-engine/api/openstacknetworkproviders \
  -H "Authorization: Bearer $TOKEN" -H "Accept: application/json" \
  -H "Prefer: persistent-auth"
  # обратить внимание на поле "username"
- # 3. Если ошибка сертификата – импортировать сертификат провайдера в доверенное хранилище Engine:
  PROVIDER_ID=<id из предыдущего шага>
  CERT=$(curl -sk https://<fqdn>/ovirt-

```
engine/api/openstacknetworkproviders/$PROVIDER_ID/certificates \
  -H "Authorization: Bearer $TOKEN" -H "Accept: application/json" -H "Prefer: persistent-auth" \
  | python3 -c 'import sys,json; print(json.load(sys.stdin)["certificate"][0]["content"])')
curl -sk -X POST https://<fqdn>/ovirt-engine/api/openstacknetworkproviders/$PROVIDER_ID/importcertificates \
  -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" -H "Prefer: persistent-auth" \
  -d "{\"certificates\":{\"certificate\":[{\"content\":\"$CERT\"}]}}"
# (в Administration Portal – тот же результат даёт кнопка
#  Compute → Providers → ovirt-provider-ovn → Import Certificates)
```

- `# 4. Если ошибка "unauthorized" – исправить логин/пароль регистрации:`
  `curl -sk -X PUT https://<fqdn>/ovirt-engine/api/openstacknetworkproviders/$PROVIDER_ID \`
  `-H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" -H "Prefer: persistent-auth" \`
  `-d '{"username":"<реальный профиль AAA>","password":"<пароль>"}'`
- `# 5. Проверить соединение (ожидаемый ответ: {"status": "complete"}):`
  `curl -sk -X POST https://<fqdn>/ovirt-engine/api/openstacknetworkproviders/$PROVIDER_ID/testconnectivity \`
  `-H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" -H "Prefer: persistent-auth" -d '{}'`
- `# 6. Не дожидаясь фонового цикла синхронизации (обычно ~5 минут) – импортировать сеть немедленно:`
  `curl -sk https://<fqdn>/ovirt-engine/api/openstacknetworkproviders/$PROVIDER_ID/networks \`
  `-H "Authorization: Bearer $TOKEN" -H "Accept: application/json" -H "Prefer: persistent-auth"`
  `curl -sk https://<fqdn>/ovirt-engine/api/datacenters \`
  `-H "Authorization: Bearer $TOKEN" -H "Accept: application/json" -H "Prefer: persistent-auth"`
  `curl -sk -X POST https://<fqdn>/ovirt-engine/api/openstacknetworkproviders/$PROVIDER_ID/networks/<network_id>/import \`
  `-H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" -H "Prefer: persistent-auth" \`
  `-d '{"data_center":{"id":"<dc-id>"}}'`
- `# 7. Проверить: кнопка «🩺 Проверить конфигурацию» на Топологии –`
  `#  проверка engine_provider_sync должна вернуть pass.`
  `««««`

12.5. 400 «Attaching by subnet requires the subnet to have a default gateway specified» при подключении подсети к маршрутизатору

Причина: у подсети не задан «gateway_ip».
Решение: отредактировать подсеть (значок ✎ Подсеть), указать шлюз.

12.6. 400 «already connected to router ...»

Причина: сеть/подсеть уже подключена к другому (или тому же) маршрутизатору – провайдер не поддерживает переключение подключения одним действием.
Решение: использовать кнопку «⇄ Сменить маршрутизатор» непосредственно у самой подсети (см. раздел 10.4), не кнопку «⇄ К роутеру» у сети.

12.7. Кнопка «⇄ К роутеру» на сети неактивна

Причина (ожидаемое поведение, не ошибка): сеть уже подключена к маршрутизатору как interior либо уже используется как External Gateway. Подсказка (title) на самой кнопке называет причину.
Решение: для смены маршрутизатора – раздел 10.4; для сети-шлюза переключение на interior не предусмотрено намеренно.

12.8. 400 «Unable to create more than one subnet for network ...»

Причина: провайдер разрешает только одну подсеть на сеть.
Решение: кнопка на сети автоматически превращается в «✎ Подсеть» (редактирование), если подсеть уже есть – создавать вторую не нужно.

12.9. 400 при попытке выключить DHCP у подсети

Причина: провайдер не поддерживает «enable_dhcp=false» ни при каких условиях.
Решение: не пытаться отключать – в форме этой настройки нет намеренно, включён всегда.

12.10. 400 «Router \<id\> still has ports» при удалении маршрутизатора

Причина: корректная защита провайдера – у маршрутизатора ещё есть подключённые interior-сети.
Решение: сообщение плагина само называет конкретную сеть/CIDR – сначала отключить её (панель маршрутизатора на Топологии, значок ✕ у нужной подсети), затем повторить удаление.

12.11. 409 при удалении сети/подсети

Причина: она используется как внешний шлюз активного маршрутизатора – эта проверка реализована на стороне плагина (сам провайдер её не делает и без неё удаление сломало бы «GET /routers» сразу для всех маршрутизаторов).

Решение: сначала снять шлюз с маршрутизатора (кнопка «⚔ Убрать шлюз»), либо, если точно уверены – передать параметр «force=true».

12.12. «physical_network_binding: warning» в «🩺 Проверить конфигурацию»

Причина: несущая сеть для «provider:physical_network» не создана, не добавлена в кластер или не «operational» ни на одном хосте.
Решение: кнопка «Настроить физическую привязку» на этой сети → «Проверить» – сообщение укажет точную причину (см. раздел 7.2).

12.13. Физическая привязка настроена (мастер прошёл успешно, «physical_network_binding: pass»), но реального трафика нет

Возможные причины (проверять по порядку):

- Шаг 7.3 (OVS-мост) ещё не выполнен вручную на этом хосте;
- При проверке «ovn-bridge-mappings» смотрели не в то поле – нужно «other_config», не «external_ids» (см. раздел 7.4);
- Хоста вообще нет в списке «ovn-sbctl list Chassis» – служба «ovn-controller» не запущена или не подключена (раздел 7.4: «systemctl status ovn-controller» / «systemctl enable --now ovn-controller»);
- Несущая сеть осталась VM Network вместо Non-VM Network – при наличии реальных ВМ на ней автоматическая конвертация мастером была заблокирована как небезопасная, нужно создать новую несущую сеть на другом, гарантированно свободном VLAN.

12.14. RPM ставится, но плагин отдаёт 500 на все запросы сразу после установки

Причина (известный класс бага, исправлен начиная с версии 1.4.30): конфигурационный файл ставился с владельцем «root:root», а бэкенд работает от пользователя «ovirt» и не мог его прочитать.
Решение: обновиться до текущей версии пакета – «%post» теперь сам выставляет «root:ovirt» на конфигурационном файле при каждой установке (и первой, и при обновлении).

12.15. Кнопка «🔒 NAT» показывает ошибку прав (sudo) вместо правил

Причина №1 – не установлен/не подхватился sudoers-файл «/etc/sudoers.d/keyvirt-ovn-firewall-nat-readonly» (ставится самим RPM, раздел 8) – либо у него испорчены права/владелец (sudo молча игнорирует файлы «sudoers.d», если они не «0440 root:root»), либо «ovn-nbctl» отсутствует на этом хосте.
Решение:

- ls -la /etc/sudoers.d/keyvirt-ovn-firewall-nat-readonly
  # ожидается: -r--r-----. root root
- visudo -cf /etc/sudoers.d/keyvirt-ovn-firewall-nat-readonly
- which ovn-nbctl

Причина №2 (на хостах с SELinux enforcing – Platform V SberLinux, RHEL 8+) – не загружен SELinux-модуль «keyvirt_ovn_nat_sudo» и/или выключен булев «httpd_mod_auth_pam» (оба должны ставиться автоматически в «%post», см. раздел 7.7). Сообщение об ошибке в этом случае обычно содержит «PAM»/«audit system»/«database connection failed», а не прямое «password is required».
Решение:

- semodule -l | grep keyvirt_ovn_nat_sudo
- getsebool httpd_mod_auth_pam # второе значение должно быть "on"

Если модуля нет или булев выключен – переустановить пакет («dnf reinstall keyvirt-ovn-firewall-plugin»), «%post» гарантированно выполняет оба шага заново. Если денайлы всё равно продолжаются – диагностировать через временное отключение «dontaudit» (показывает скрытые денайлы, ничего не меняет в enforcement):

- semodule -DB # повторить запрос через UI/curl
- ausearch -m avc -ts recent | grep denied
- semodule -B

Если файла нет вообще – переустановить пакет («dnf reinstall keyvirt-ovn-firewall-plugin»), «%files» гарантированно его ставит.

Причина №3 – переустановлен ovn-provider/OVN, обслуживающий процесс сокета «/var/run/ovn/ovnnb_db.sock», теперь работает в ДРУГОМ домене SELinux (не «unconfined_service_t», как на проверенном стенде) – модуль «keyvirt_ovn_nat_sudo» разрешает «connectto» только к «unconfined_service_t». Проверить:

- ps -Z -C ovsdb-server

Если домен другой – потребуется править «.te» (требует пересборки RPM/пересоздания модуля вручную).

12.16. Общая диагностика «с нуля» при любых непонятных проблемах

««««bash

- # 1. Проверить, что сервис вообще отвечает:
  curl -sk http://127.0.0.1/ovn-firewall-api/health | python3 -m json.tool
- # 2. Проверить владельца и права конфига (ожидается: -rw-r-----. root ovirt):
  ls -la /etc/ovirt-engine/ovn-firewall.conf

- # 3. Проверить журнал доступа Apache:
  tail -50 /var/log/httpd/ovn-firewall-access.log
- # 4. Проверить журнал самого mod_wsgi процесса (ошибки Python видны через общий журнал httpd):
  journalctl -u httpd -n 100 --no-pager

5. Открыть Топологию → «Проверить конфигурацию» – это самый полный самодиагностический инструмент плагина, покрывающий большинство сценариев из этого раздела одним нажатием.

12.17. ВМ имеет доступ к другим ВМ/в интернет, даже если в её группе безопасности нет ни одного разрешающего egress-правила

Причина: не баг – ожидаемое поведение самого «ovirt-provider-ovn» (см. раздел 11). Egress-трафик от ВМ разрешён безусловно автоматическим ACL-правилом, которое провайдер создаёт для каждой группы безопасности сам – правила, настроенные через плагин, реально ограничивают только ingress (входящий трафик).
Решение: если нужно ограничить именно исходящий трафик ВМ – средствами этого плагина (и вообще «ovirt-provider-ovn»/Neutron Security Group Rules API) это невозможно в принципе, не только в текущей версии. Нужен другой механизм (например iptables/nftables внутри самой ВМ) – раздел 11 объясняет причину подробнее.