KeyVirt

ИНСТРУКЦИЯ ПО РЕАЛИЗАЦИИ

Реализация поддержки многофакторной аутентификации
(Radius, ADFS)
для доступа в Web-интерфейс платформы виртуализации
KeyVirt

Оглавление

1. Вводная

Настоящий документ предназначен для настройки поддержки многофакторной аутентификации для доступа в Web-интерфейс интерфейса управления платформой виртуализации KeyVirt (С использованием Radius или ADFS).

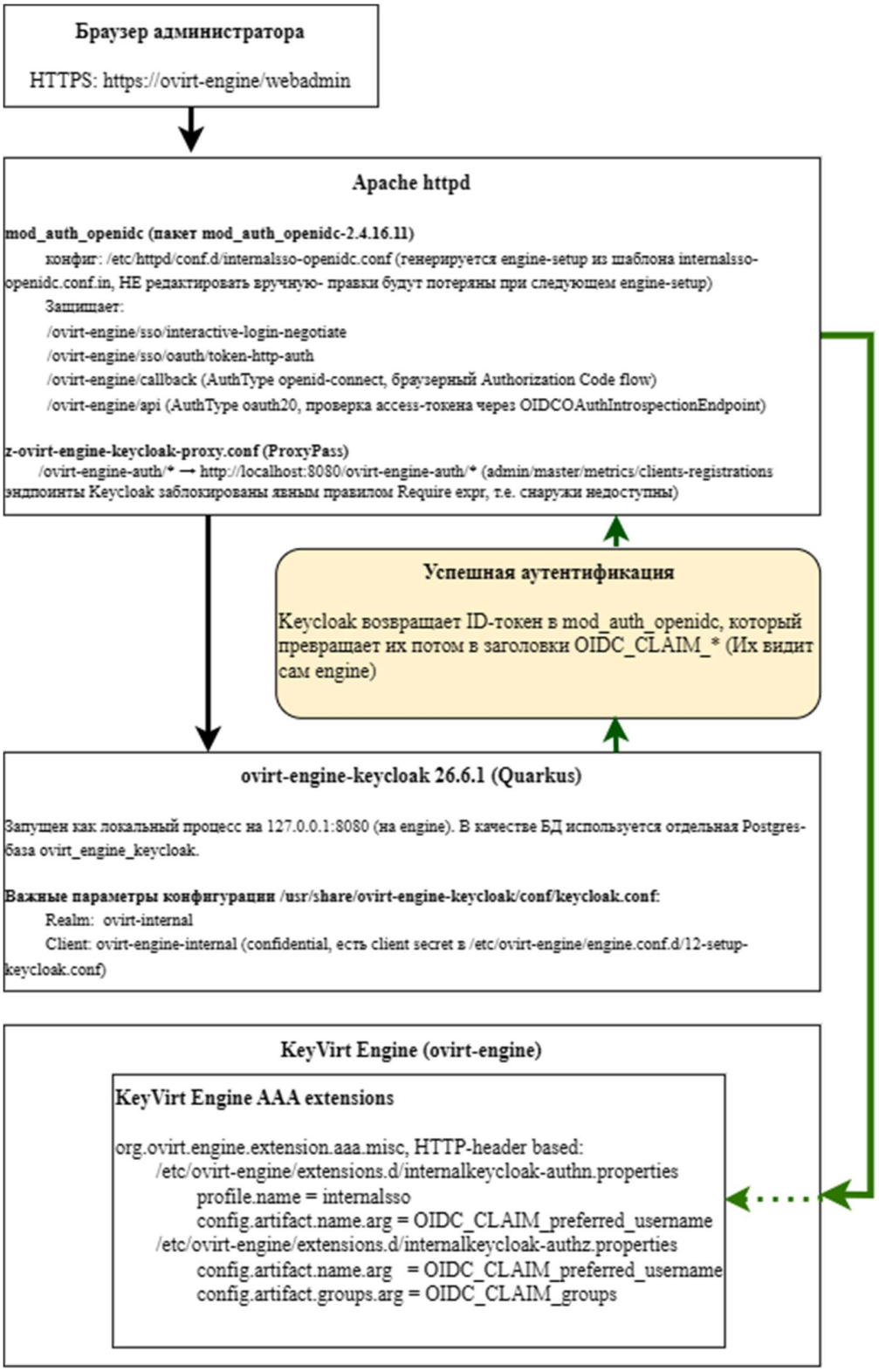
Платформа поддерживает два основных сценария:

Вариант А: RADIUS как второй фактор внутри собственного флоу аутентификации Keycloak. Пользователь по-прежнему аутентифицируется логином и паролем в Keycloak (LDAP/AD/локальная БД), а следующим шагом Keycloak сам обращается к RADIUS-серверу для проверки OTP/токена.

Вариант Б: ADFS как внешний Identity Provider. В этом варианте Keycloak не проверяет пароль и MFA сам, а перенаправляет пользователя на ADFS, и ADFS сам решает, что считать успешной аутентификацией (пароль + Azure MFA, смарт-карта, сертификат и т.п.). Keycloak в этой схеме только посредник (SAML/OIDC Service Provider), MFA полностью на стороне ADFS.

2. Базовая архитектура аутентификации

В настоящем разделе рассмотрен процесс аутентификации в интерфейсе KeyVirt «Из коробки», без дополнительных настроек.



Важный существующий запасной путь (break-glass), который нельзя трогать при внедрении MFA:

- /etc/ovirt-engine/aaa/internal.properties
- /etc/ovirt-engine/extensions.d/internal-authn.properties
    - profile.name = internal

3. Общие правила безопасного внедрения MFA

1) Не редактировать Authentication Flow, привязанный к realm по-умолчанию (Browser), пока он используется боевым клиентом ovirt-engine-internal. Порядок работ:
   a) Клонировать существующий «Browser» flow (Keycloak Admin Console → Authentication → список flows→ выбрать Browser → кнопка "Duplicate") например, в «Browser with MFA»;
   b) Вносить все изменения (RADIUS-execution, редирект на ADFS) только в копии;
   c) Протестировать копию на тестовом клиенте/тестовом пользователе (Keycloak позволяет привязать конкретный flow к конкретному клиенту через Client → Advanced → Authentication flow overrides → Browser Flow, не трогая realm-default);
   d) Только после успешного теста переключить flow override на боевом клиенте ovirt-engine-internal, либо (если нужно для всех клиентов сразу) сделать «Browser with MFA» flow realm по-умолчанию (Authentication → Bindings → Browser Flow).

2) Держать наготове минимум одну активную сессию/окно браузера, уже залогиненную как administrator, ПЕРЕД переключением flow по-умолчанию, на случай ошибки конфигурации, чтобы было чем откатить через Admin Console, не имея доступных опций для входа.

3) Не трогать профиль internal (admin@internal). Держать пароль этой учётной записи в безопасном месте (Например, сейфе) как процедуру аварийного восстановления доступа.

4) Риск апгрейда KeyVirt Engine

   ovirt-engine-keycloak- управляемый инструментами engine-setup компонент. Нет гарантии, что custom SPI-провайдер (RADIUS, раздел 4) и ручные правки realm (Identity Provider для ADFS, Authentication Flow) переживут `engine-setup` при следующем крупном обновлении Engine (Keycloak может быть переустановлен/обновлён как часть пакета ovirt-engine-keycloak). Обязательно:

   a) Сохранить экспорт realm (Admin Console → Realm settings → Action → Partial export, включая клиентов и Identity Providers) и сам SPI .jar в отдельном месте (не только на диске сервера);
   b) Включить восстановление этой конфигурации в процедуру обновления платформы как отдельный контрольный пункт;

c) После каждого engine-setup upgrade проверять процедуру живым тестом, чтобы убедиться что MFA по-прежнему работает, не полагаясь только на то, что WebAdmin в принципе открылся (по-умолчанию, Keycloak может тихо откатиться на flow без MFA, и вход всё ещё будет работать, но без второго фактора, что является скрытым отказом, не сразу заметным).

5) Логи для диагностики любого из вариантов:

/var/log/keycloak/keycloak.log: сам Keycloak (SPI-ошибки, отказ соединения с RADIUS/ADFS);
/var/log/httpd/error_log: mod_auth_openidc, несовпадение redirect_uri, ошибки метаданных
/var/log/ovirt-engine/engine.log: если заголовки OIDC_CLAIM_* дошли, но авторизация в Engine не прошла (например, не тот groups claim)

4. MFA «ВАРИАНТ А»: RADIUS как второй фактор

4.1. Принцип работы

Browser flow Keycloak состоит из последовательности "execution":

Cookie → Kerberos (опционально) → Identity Provider Redirector → Username Password Form → (здесь добавляется RADIUS-execution, Requirement = REQUIRED) → успешный логин.

RADIUS-execution это Java-класс, реализующий SPI-интерфейс Keycloak org.keycloak.authentication.Authenticator. После того как пользователь ввёл логин/пароль и они прошли проверку (LDAP/AD/локальная БД Keycloak), Keycloak показывает дополнительную форму (обычно поле «Введите код»), значение из которой SPI-провайдер отправляет как RADIUS Access-Request (протокол PAP или CHAP, поле User-Name = логин пользователя, User-Password = введённый OTP/PIN) на внешний RADIUS-сервер. Ответ Access-Accept означает что второй фактор пройден. Если ответ Access-Reject/Access-Challenge, то форма показывает ошибку, либо запрашивает дополнительный ввод (Access-Challenge, если сервер поддерживает multi-step, например запрос PIN отдельно от OTP).

RADIUS-сервером на другой стороне может быть:

- FreeRADIUS с модулем OTP (например rlm_otp, Google Authenticator PAM-модуль через rlm_pam, либо коммерческий модуль);
- Любое коммерческое MFA-решение (в т.ч. отечественные средства двухфакторной аутентификации), предоставляющее RADIUS-интерфейс (это стандартный и самый частый способ интеграции корпоративных MFA-платформ с внешними приложениями именно потому, что RADIUS поддерживают все, у кого нет прямой интеграции по API).

4.2. Получение/подготовка SPI-провайдера

Готового провайдера от самого проекта Keycloak нет- потребуется использовать существующий open-source SPI-провайдер, реализующий RADIUS-аутентификацию для нужной версии Keycloak (26.x, SPI Keycloak между мажорными версиями иногда меняется. Обязательно требуется проверить совместимость выбранной реализации именно с 26-й веткой перед использованием в продуктивной среде, либо скомпилировать/собрать провайдер под ту же версию keycloak-server-spi/keycloak-services, которая установлена в вашей инфраструктуре).

Зависимости провайдера (RADIUS-клиентская библиотека и т.п.) следует собирать в один "fat jar" (shaded/uber jar), поскольку Keycloak подхватывает provider'ы из своего classpath, а сторонние транзитивные зависимости отдельно не резолвит.

4.3. Установка SPI в ovirt-engine-keycloak

Каталог провайдеров находится по пути: /usr/share/ovirt-engine-keycloak/providers/

Для провайдера нужно выполнить:

- cp radius-authenticator-<version>.jar /usr/share/ovirt-engine-keycloak/providers/
- chown ovirt:ovirt /usr/share/ovirt-engine-keycloak/providers/radius-authenticator <version>.jar

После копирования .jar в providers/ Keycloak должен пересобрать внутренний кэш провайдеров. Для этого нужно выполнить:

- /usr/share/ovirt-engine-keycloak/bin/kc.sh build

Затем перезапустить сервис:

- systemctl restart ovirt-engine-keycloak *# либо фактическое имя unit-файла — проверить* systemctl status | grep -i keycloak *# на конкретном сервере, имя юнита может отличаться от версии к версии сборки*

Проверить, что провайдер загрузился без ошибок:

- grep -i "radius|provider" /var/log/keycloak/keycloak.log | tail -50

4.4. Настройка Authentication Flow

Admin Console → выбрать realm «ovirt-internal» → Authentication → клонировать «Browser» (см. раздел 3.1) → в скопированном flow, на шаге после "Username Password Form", нажать "Add step" → выбрать установленный RADIUS-authenticator из списка (появится под тем именем, которое провайдер регистрирует в AuthenticatorFactory.getDisplayType()) → Requirement = REQUIRED.

Настройки самого шага (адрес/порт RADIUS-сервера, shared secret, таймаут, NAS-Identifier) задаются через «Config» у этого execution. Конкретный набор полей зависит от того, что реализует выбранный/написанный SPI-провайдер (обычно: radius.server.host, radius.server.port (1812 по-умолчанию для Access-Request), radius.shared.secret, radius.timeout.ms).

Привязать flow к клиенту ovirt-engine-internal для тестирования (без влияния на остальные клиенты realm):

Clients → ovirt-engine-internal → Advanced → Authentication flow overrides → Browser Flow → выбрать «Browser with MFA».

После успешного теста нужно либо оставить override только на этом клиенте (если MFA нужен только для WebAdmin, а не для остальных потребителей realm), либо сделать flow default для всего realm (раздел 3.1, п.d).

4.5. Тестирование

- curl -k -s -X POST
  -H "Accept: application/json"
  -H "Content-Type: application/x-www-form-urlencoded"
  "https://<engine-fqdn>/ovirt-engine-auth/realms/ovirt-internal/protocol/openid-connect/token"
  -d "grant_type=password&client_id=ovirt-engine-internal&client_secret=<secret>&username=<user>&password=<pass>"

ВНИМАНИЕ: прямой grant_type=password (Resource Owner Password Credentials) в Keycloak НЕ проходит через Browser flow и, следовательно, НЕ проверяет RADIUS-шаг. Это ожидаемо и не является уязвимостью в MFA, если только Direct Access Grants не включены для этого клиента и не используются реальными потребителями (То есть, нужно проверить проверить: Clients → ovirt-engine-internal → Capability config → Direct access grants. Если планируется MFA именно для браузерного входа в WebAdmin, этот флаг стоит выключить, иначе RADIUS-проверку можно обойти прямым вызовом token endpoint). Полноценно протестировать RADIUS-шаг можно только через настоящий браузерный вход в https://<engine-fqdn>/ovirt-engine/webadmin (полный редирект на Keycloak login page).

5. MFA «ВАРИАНТ Б»: ADFS как внешний identity provider

5.1. Принцип работы

В этом варианте Keycloak выступает Service Provider (SP) по протоколу SAML 2.0 (либо Service Provider/Relying Party по OpenID Connect, если версия ADFS поддерживает OIDC (AD FS 2016+ поддерживает OAuth2/OIDC-эндпоинты, но классическая и наиболее совместимая схема для ADFS- именно SAML 2.0). ADFS выступает Identity Provider (IdP),

то есть именно на стороне ADFS настраивается фактическая политика MFA (Azure MFA, сертификат, смарт-карта или то что уже развёрнуто в организации для ADFS).

Пользователь на экране логина Keycloak видит либо кнопку «Войти через <название ADFS>», либо (если настроен принудительный редирект, см. 5.4) автоматически перенаправляется на ADFS, минуя форму Keycloak вообще.

5.2. Настройка на стороне ADFS: Relying Party Trust

На сервере ADFS (оснастка "AD FS Management" либо PowerShell):

- Add-AdfsRelyingPartyTrust -Name "Keycloak-KeyVirt" -MetadataUrl "https://<engine-fqdn>/ovirt-engine-auth/realms/ovirt-internal/broker/adfs/endpoint/descriptor"

Метаданные SP Keycloak отдаёт по этому пути только ПОСЛЕ того, как Identity Provider с alias «adfs» создан в Keycloak- см. 5.3. Реальный порядок действий на практике- сначала частично создать IdP в Keycloak, скопировать его SP metadata URL/entity ID, затем зарегистрировать Relying Party Trust в ADFS, затем вернуться в Keycloak и указать метаданные ADFS.

Если MetadataUrl недоступен ADFS-серверу по сети (типичная ситуация- ADFS в закрытом периметре без прямого доступа к Engine), то можно использовать вариант с локальным файлом метаданных:

- Add-AdfsRelyingPartyTrust -Name "Keycloak-KeyVirt" -MetadataFile "sp-metadata.xml"

Настроить в ADFS Claim Rules (обязательно передать группы пользователя- см. критичное замечание в 5.5):

- LDAP Attribute: SAM-Account-Name --> Outgoing: Name ID / preferred_username;
- LDAP Attribute: Token-Groups - Unqualified Names --> Outgoing: группы (имя исходящего claim согласовать с маппингом на стороне Keycloak, раздел 5.5).

Настроить Access Control Policy для этого Relying Party Trust так, чтобы требовалась MFA (Azure MFA/дополнительный фактор). Конкретные шаги зависят от того, какой MFA-адаптер уже подключён к ADFS (Azure MFA Adapter, сторонний ADFS MFA adapter и т.п.)- эта часть вне зоны ответственности KeyVirt, она настраивается на стороне службы каталогов силами администраторов ADFS.

5.3. Настройка Identity Provider в Keycloak

- Admin Console → realm "ovirt-internal" → Identity Providers → Add provider → SAML v2.0.
  a) Alias: adfs
  b) Service provider entity ID: https://<engine-fqdn>/ovirt-engine-auth/realms/ovirt-internal
  c) Single Sign-On Service URL: https://<adfs-fqdn>/adfs/ls/ (стандартный SAML SSO endpoint ADFS)

d) NameID Policy Format: urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified (либо persistent — согласовать с настройками NameID в Claim Rules ADFS)
e) Signature Algorithm: RSA_SHA256
f) Want AuthnRequests Signed: On (рекомендуется)
g) Validate Signatures: On
h) Signing Certificate: вставить сертификат подписи токенов ADFS (Get-AdfsCertificate -CertificateType Token-Signing на сервере ADFS)

После сохранения Keycloak генерирует собственные SP-метаданные по адресу вида:

«https://<engine-fqdn>/ovirt-engine-auth/realms/ovirt-internal/broker/adfs/endpoint/descriptor» - именно этот URL/файл нужен для регистрации Relying Party Trust в ADFS (раздел 5.2).

5.4. Принудительный редирект на ADFS (опционально)

Если ADFS должен быть ЕДИНСТВЕННЫМ способом входа (без формы логин/пароль самого Keycloak), то нужно добавить в Browser flow (в копию, см. раздел 3.1) execution «Identity Provider Redirector» ПЕРЕД «Username Password Form», Requirement = ALTERNATIVE, и в его Config указать Default Identity Provider = adfs. Тогда любой пользователь, дошедший до экрана логина Keycloak, будет немедленно перенаправлен на ADFS, минуя локальную форму.

Если нужно оставить возможность выбора (локальный вход или через ADFS, например для того же break-glass администратора), то Identity Provider Redirector не добавлять. Кнопка входа через ADFS появится на форме логина автоматически, наравне с полями логин/пароль, при условии что Identity Provider включён и «Hide on login page» выключен в его настройках.

5.5. Передача групп для авторизации в KeyVirt Engine

Как зафиксировано в разделе 2, «internalkeycloak-authz» читает claim groups из ID-токена, который Keycloak выдаёт клиенту ovirt-engine-internal. При обычном локальном логине этот claim формируется из членства пользователя в группах самого Keycloak. При входе через ADFS группы приходят в SAML-ассершне от ADFS и не становятся автоматически группами Keycloak – для них нужен явный маппинг:

- Identity Providers → adfs → Mappers → Add mapper:
  a) Mapper type: Attribute Importer (или Advanced Attribute to Role, если нужна конвертация groups → Role)
  b) Attribute Name: имя исходящего claim из ADFS Claim Rules (5.2), например http://schemas.xmlsoap.org/claims/Group
  c) User Attribute: groups
- И затем на самом клиенте ovirt-engine-internal: Client scopes → (dedicated scope клиента) → Add mapper → By configuration → User Attribute →
  a) User Attribute = groups
  b) Token Claim Name = groups

c) Add to ID token = On
d) Multivalued = On (чтобы это значение реально попало в ID-токен, а не осталось только атрибутом пользователя внутри Keycloak)

Без этого шага пользователи, вошедшие через ADFS, будут аутентифицированы успешно, но окажутся без каких-либо прав в KeyVirt Engine (groups придёт пустым), что является типичной ошибкой первого внедрения такой федерации. Стоит проверить это отдельно и в первую очередь при тестировании (раздел 6).

5.6. First Broker Login flow

При первом входе нового (ранее не встречавшегося Keycloak) пользователя через ADFS Keycloak по умолчанию покажет форму «подтверждения email» или создания локальной учётной записи (First Broker Login flow). Если требуется, чтобы пользователи ADFS автоматически становились полноценными пользователями Keycloak без дополнительного шага, то нужно:

- Identity Providers → adfs → First Login Flow
  a) выбрать (или создать копию) flow, где execution "Create User If Unique" стоит REQUIRED, а "Review Profile" — DISABLED.

6. Проверка результата (Обязательный чек-лист)

1) Вход через основной браузерный путь WebAdmin (https://<engine-fqdn>/ovirt-engine/webadmin) требует пройденного второго фактора (RADIUS) либо полностью проходит через ADFS- проверить вживую в браузере, не только curl'ом.
2) При неверном OTP/RADIUS-ответе Access-Reject- вход отклонён с понятной ошибкой, а не «тихим» пропуском дальше.
3) Пользователь, вошедший через ADFS, реально видит те разделы WebAdmin, которые соответствуют его правам в KeyVirt Engine (не «Access Denied» на всё)- прямая проверка выполнения раздела 5.5.
4) grant_type=password (Direct Access Grants) для клиента ovirt-engine-internal либо выключен, либо осознанно оставлен включённым с пониманием, что это обходной путь мимо MFA (раздел 4.5).
5) Учётная запись admin@internal (профиль internal, раздел 2) по-прежнему работает без каких-либо изменений- контрольный вход под ней должен быть выполнен и задокументирован до того, как новый flow становится default для всего realm.
6) Экспорт realm (раздел 3.4) сохранён вне сервера, дата снятия зафиксирована.
7) Полный тестовый прогон API-доступа (/ovirt-engine/api)- он защищён отдельным путём (OIDCOAuthIntrospectionEndpoint, раздел 2) и MFA-flow браузера на него не действует. Убедиться, что это ожидаемое поведение (обычно MFA требуется для «Web-интерфейса», то есть речь про интерактивный вход, а не про программный доступ по

токену. Если требование шире, то этот пункт нужно перепроверять отдельно, т.к. текущая архитектура REST API не проходит через Browser flow вообще).