


# Развёртывание и настройка oVirt 4.0. Часть 9.

## Интеграция oVirt с LDAP-каталогом Microsoft Active Directory

 [blog.it-kb.ru/2016/10/04/install-ovirt-4-0-part-9-ovirt-engine-aaa-integration-profile-for-users-authentication-via-ldap-microsoft-active-directory](http://blog.it-kb.ru/2016/10/04/install-ovirt-4-0-part-9-ovirt-engine-aaa-integration-profile-for-users-authentication-via-ldap-microsoft-active-directory)

Автор: Алексей Максимов

04.10.2016



В этой и последующей части серии записей о настройке среды управления виртуализацией **oVirt 4.0** будет рассмотрен практический пример интеграции функционала разграничения прав доступа oVirt с внешним **LDAP**-каталогом на базе домена **Microsoft Active Directory**. Сначала мы пошагово рассмотрим процедуру настройки профиля интеграции oVirt с LDAP-каталогом, а затем, в отдельной заметке, рассмотрим настройку автоматической аутентификации пользователей на веб-порталах oVirt, настроив **Single sign-on (SSO)** на базе протокола **Kerberos**.

Подготовительные мероприятия для интеграции с LDAP Active Directory

Далее по пунктам мы рассмотрим ряд подготовительных мероприятий, которые необходимо провести для того, чтобы в конечном итоге можно было настроить профиль интеграции oVirt с AD:

- Установка пакета расширения oVirt
- Проверка корректности разрешения имён в службе DNS
- Создание учётной записи для поиска в LDAP-каталоге
- Создание JKS-хранилища с корневыми сертификатами доменного ЦС

Обратите внимание на то, что это необходимый минимум действий лишь в том случае, если вы не планируете использовать **Kerberos SSO** (для обеспечения возможности прозрачной аутентификации пользователей без явного указания имени пользователя и пароля при входе на веб-порталы oVirt), а хотите ограничиться лишь простой **Form-based** аутентификацией (явный ввод имени пользователя и пароля на веб-странице входа порталов oVirt) на основе данных LDAP-каталога. Требования специфичные для Kerberos SSO будут рассмотрены отдельно в следующей заметке.

Установка пакета расширения oVirt

В ранних версиях для интеграции с разными реализациями LDAP-каталогов использовалась утилита **engine-manage-domains**, пример использования которой можно найти, например, [здесь](#). Теперь эта утилита считается устаревшей и вместо неё нужно использовать **ovirt-engine-extension-aaa-ldap-setup**. Установим на сервере oVirt Engine соответствующий пакет:

```
# yum install ovirt-engine-extension-aaa-ldap-setup
```

Проверка корректности разрешения имён в службе DNS

Убедимся в том, что в конфигурационном файле `/etc/resolv.conf` на нашем сервере **oVirt Engine** присутствуют записи о **DNS серверах**, способных разрешать **A** и **SRV** записи, относящиеся к нашему домену Active Directory. Примеры запросов на разрешение SRV-записей, относящихся к **LDAP**/глобальному каталогу AD в DNS-зоне нашего домена:

```
$ dig +noall +answer _ldap._tcp.{имя домена AD} SRV
$ dig +noall +answer {имя контроллера домена AD} _ldap._tcp.gc._msdcs.{имя леса AD} SRV
```

Например, в нашем случае, в качестве LDAP-каталога будет использоваться домен `ad.holding.com` в лесу (AD Forest) `holding.com`, поэтому запросы будут выглядеть следующим образом:

```
$ dig +noall +answer _ldap._tcp.ad.holding.com SRV
$ dig +noall +answer kom-dc01.ad.holding.com _ldap._tcp.gc._msdcs.holding.com SRV
```

Забегая вперёд, могу сказать, что правильность разрешения SRV-записей из DNS в нашем конкретном примере будет не важна, так как в конечном итоге мне придётся использовать явное указание LDAP-серверов (контроллеров домена AD) по причинам, которые я опишу далее. Ну а в большинстве типичных конфигураций AD всё-таки лучше отталкиваться от условия правильной работы механизма разрешения SRV-записей из DNS. Однако при любых сценариях наши DNS серверы, как минимум, без проблем должны разрешать в IP-адреса A-записи LDAP-серверов (контроллеров домена).

Создание учётной записи для поиска в LDAP-каталоге

Для того, чтобы oVirt Engine мог аутентифицировано подключаться и выполнять поиск объектов в LDAP-каталоге контроллеров домена Active Directory, нам потребуется создать в домене служебную учётную запись пользователя. Этому пользователю не нужно давать никаких особых привилегий в домене, а достаточно иметь лишь права рядового пользователя домена, так как у рядовых пользователей AD по умолчанию есть права на чтение большинства объектов в домене (права уровня **Read Access to Directory Services**). Согласно документа [RHEV 3.6 Administration Guide 14.2. Introduction to Directory Servers](#), расширенные права в домене могут потребоваться лишь в том случае, если в дальнейшем планируется использовать механизмы **Sysprep** при развёртывании из шаблона новых ВМ (права уровня **Join a computer to the domain** и **Modify the membership of a group**), однако это не наш случай и вообще это тема для отдельного разговора.

Итак, в нашем примере в домене AD создана сервисная учётная запись рядового пользователя домена с именем `s-oVirt-LS`.

Создание JKS-хранилища с корневыми сертификатами доменного ЦС

Для возможности установки безопасных соединений по протоколу LDAP с контроллерами домена AD, нам потребуется защищать такие соединения с помощью **SSL/TLS**. То есть LDAP-сервер при соединении будет предоставлять серверу oVirt Engine некий цифровой сертификат для защиты соединения. Чтобы сервер oVirt Engine мог быть уверен в том, что это правильный LDAP-сервер с правильным сертификатом, нам потребуется скопировать на сервер oVirt Engine **корневой сертификат** Центра Сертификации (ЦС), которым был подписан сертификат LDAP-сервера.

Создадим на сервере oVirt Engine в домашней папке временный каталог для операций с файлами:

```
$ mkdir ~/AD-LDAP-Files
```

Чтобы "выудить" корневой сертификат LDAP-сервера, воспользуемся утилитой **openssl** и выполним команду типа:

```
$ openssl s_client -connect {FQDN контроллера домена}:636 -showcerts < /dev/null
```

В результате выполнения команды мы должны получить в консоль две секции с содержимым сертификатов в формате **PEM** вида:

```
...
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
...
```

Первая секция будет содержать сертификат самого LDAP-сервера (контроллера домена), вторая - корневой сертификат, которым подписан сертификат LDAP-сервера. Скопируем контент первого сертификата в файл `end.pem`, а контент второго сертификата в файл `myrootca.pem`. Затем попробуем проверить получившийся файл `myrootca.pem`, как корневой сертификат для `end.pem`

```
$ openssl verify -CAfile ~/AD-LDAP-Files/myrootca.pem ~/AD-LDAP-Files/end.pem
```

В моём случае была получена ошибка:

```
error 2 at 1 depth lookup:unable to get issuer certificate
```

Связана она с тем, что корневой сертификат, который был сохранён в файл `myrootca.pem` на самом деле не является конечным в цепочке корневых сертификатов и есть ещё один корневой сертификат более верхнего уровня. Если в вашем случае проверка прошла успешно и корневой сертификат у вас действительно всего один, то в этом разделе описания сразу можно пропустить "пляски" с получением цепочки корневых сертификатов и сразу переходить к использованию утилиты **keytool** для генерации JKS-файла из файла `myrootca.pem`

\*\*\*

Мне же дополнительно пришлось запросить всю цепочку корневых сертификатов у ЦС, сертификатом которого подписан сертификат LDAP-сервера, с доменной клиентской машины на базе **Windows** с помощью утилиты **certutil**:

```
certutil -f -config "kom-subca.ad.holding.com\KOM-SUBCA" -ca.chain  
C:\Temp\RootCACertificateChain.p7b
```

Если вам не известно имя доменного ЦС (то, что указывается в параметре -config), то, как правило, в доменной инфраструктуре AD на любом доменном компьютере можно получить имя ЦС, вызвав команду **certutil** без параметров. Ну или же, просто на просто, можно запросить цепочку корневых сертификатов в формате **PKCS #7** у администратора доменной **PKI**. Переносим файл RootCACertificateChain.p7b с Windows на сервер oVirt Engine, например, с помощью утилиты [pscp.exe](#):

```
cd /d C:\Tools\PuTTY  
pscp.exe d:\Temp\CertificateRootCAChain.p7b root@kom-ad01-ovirt1:/root/AD-LDAP-Files
```

Затем на сервере oVirt Engine конвертируем файл в формат **PEM** следующей командой:

```
$ cd ~/AD-LDAP-Files  
$ openssl pkcs7 -inform DER -in CertificateRootCAChain.p7b -print_certs -outform PEM -out myrootca_chain.pem
```

После этого откроем в текстовом редакторе получившийся файл myrootca\_chain.pem и удалим весь контент не входящий в границы секций вида -----BEGIN CERTIFICATE----- и -----END CERTIFICATE-----.

Теперь, имея в файле myrootca\_chain.pem все корневые сертификаты, снова пробуем выполнить проверку:

```
$ openssl verify -CAfile ~/AD-LDAP-Files/myrootca_chain.pem ~/AD-LDAP-Files/end.pem
```

```
/root/AD-LDAP-Files/end.pem: OK
```

Как видим, проверка проходит успешно, значит наш файл myrootca\_chain.pem действительно содержит всю цепочку корневых сертификатов и может быть использован для генерации JKS-хранилища.

\*\*\*

Теперь нам нужно создать специальный файл в формате **JKS (Java Key Store)**, в который будут помещены корневые PEM-сертификаты из файла myrootca\_chain.pem (или myrootca.pem, если корневой сертификат всего один).

Сделать это можно одной командой с помощью утилиты **keytool**. Создаём JKS-хранилище с именем файла равным имени нашего домена AD и паролем "changeit":

```
# keytool -importcert -noprompt -trustcacerts -alias myrootcachain -file ~/AD-LDAP-Files/myrootca_chain.pem -keystore /etc/ovirt-engine/aaa/ad.holding.com.jks -storepass changeit
```

Certificate was added to keystore

Получившийся файл `ad.holding.com.jks` мы будем использовать в дальнейшем при настройке профиля интеграции oVirt с Active Directory.

\*\*\*

Дополнительную информацию о получении корневых сертификатов и создании JKS-файла можно найти в разделе X.509 CERTIFICATE TRUST STORE в файле встроенной справки README:

```
# locate README | grep ovirt-engine-extension-aaa-ldap
```

```
/usr/share/doc/ovirt-engine-extension-aaa-ldap-1.2.1/README
```

\*\*\*

На данном этапе можно сказать, что необходимые подготовительные мероприятия для успешного создания профиля интеграции oVirt с LDAP-каталогом на базе AD выполнены.

Создание профиля интеграции oVirt с LDAP Active Directory

В случае, если лес Active Directory (AD Forest) имеет простую структуру с единственным одноимённым доменом и небольшим количеством контроллеров домена, то создать профиль интеграции oVirt с LDAP-сервером (контроллер домена AD с ролью Global Catalog) можно будет с помощью утилиты **ovirt-engine-extension-aaa-ldap-setup**. И скорее всего у вас это не вызовет особых проблем.

Однако в моём случае ситуация осложнилась тем, что в лесу AD имеется несколько под-доменов разного назначения с множеством территориально разрозненных сайтов (AD Sites) и контроллеров домена, а к контроллерам корневого домена доступ ограничен, как на сетевом, так и на административном уровне. Поэтому, забегая вперёд, скажу, что в моём случае процедура создания и настройки профиля интеграции будет представлять собой ритуал "мануальной терапии", для которого в качестве основы будет взят документ [RHEV 3.6 Administration Guide - 14.3.3. Configuring an External LDAP Provider \(Manual Method\)](#).

Если в вашей инфраструктуре AD всё достаточно примитивно, то самым простым и удобным вариантом создания профиля интеграции всё-таки будет вызов утилиты **ovirt-engine-extension-aaa-ldap-setup**. В ходе выполнения этой утилиты вам будет в удобной форме задано несколько вопросов о вашем LDAP-каталоге, затем сразу будет предложено выполнить ряд тестов на предмет поиска, аутентификации и авторизации в каталоге...

```
root@KOM-AD01-OVIRT1:~  
[root@KOM-AD01-OVIRT1 ~]# ovirt-engine-extension-aaa-ldap-setup  
[ INFO ] Stage: Initializing  
[ INFO ] Stage: Environment setup  
Configuration files: ['/etc/ovirt-engine-extension-aaa-ldap-setup.conf.d/10-packaging.conf']  
Log file: /tmp/ovirt-engine-extension-aaa-ldap-setup-20160927162427-z2gphc.log  
Version: otopi-1.5.2 (otopi-1.5.2-1.el7.centos)  
[ INFO ] Stage: Environment packages setup  
[ INFO ] Stage: Programs detection  
[ INFO ] Stage: Environment customization  
Welcome to LDAP extension configuration program  
Available LDAP implementations:  
1 - 389ds  
2 - 389ds RFC-2307 Schema  
3 - Active Directory  
4 - IPA  
5 - Novell eDirectory RFC-2307 Schema  
6 - OpenLDAP RFC-2307 Schema  
7 - OpenLDAP Standard Schema  
8 - Oracle Unified Directory RFC-2307 Schema  
9 - RFC-2307 Schema (Generic)  
10 - RHDS  
11 - RHDS RFC-2307 Schema  
12 - iPlanet  
Please select: 3
```

В завершении своей работы утилита настройки сообщит вам о создании трёх конфигурационных файлов:

- /etc/ovirt-engine/aaa/{имя профиля}.properties
- /etc/ovirt-engine/extensions.d/{имя профиля}-authz.properties
- /etc/ovirt-engine/extensions.d/{имя профиля}-authn.properties

По вышеописанным причинам, я не буду пошагово рассматривать работу этой утилиты, так как в моём практическом примере с помощью этой утилиты я так и не смог добиться желаемого результата. К тому же, после двух дней экспериментов с расширением ovirt-engine-extension-aaa-ldap, я пришёл к однозначному выводу о том, что если вы хотите настроить интеграцию oVirt с LDAP именно так, как вам это нужно, то единственный результативный вариант – отказаться от использования вышеупомянутой утилиты и создать профиль интеграции вручную.

\*\*\*

Итак, приступим.

После установки пакета **ovirt-engine-extension-aaa-ldap-setup**, возможные примеры конфигурационных файлов настройки профилей интеграции для разных реализаций LDAP-каталогов можно найти в каталоге: /usr/share/ovirt-engine-extension-aaa-ldap/examples/. Действующие же профили интеграции располагаются в каталоге /etc/ovirt-engine/.

Например, чтобы вручную настроить профиль интеграции с Active Directory, скопируем рекурсивно файлы из подкаталога ../examples/ad/ в каталог /etc/ovirt-engine/:

```
# cp -r /usr/share/ovirt-engine-extension-aaa-ldap/examples/ad/. /etc/ovirt-engine
```

Тем самым мы создадим 3 файла:

```
/etc/ovirt-engine/aaa/profile1.properties  
/etc/ovirt-engine/extensions.d/profile1-authn.properties  
/etc/ovirt-engine/extensions.d/profile1-authz.properties
```

Теперь нам нужно переименовать эти файлы так, чтобы в их имени вместо части "profile1" было использовано имя создаваемого нами профиля интеграции. Это имя будет отображаться в выпадающем списке профилей на веб-страницах входа на порталы oVirt - **Administration Portal** и **User Portal**. Логичнее всего задать в качестве имени профиля имя нашего домена Active Directory, например:

```
# mv /etc/ovirt-engine/aaa/profile1.properties /etc/ovirt-  
engine/aaa/ad.holding.com.properties  
# mv /etc/ovirt-engine/extensions.d/profile1-authn.properties /etc/ovirt-  
engine/extensions.d/ad.holding.com-authn.properties  
# mv /etc/ovirt-engine/extensions.d/profile1-authz.properties /etc/ovirt-  
engine/extensions.d/ad.holding.com-authz.properties
```

Далее нам нужно отредактировать каждый из этих трёх файлов. Первым будет файл свойств конфигурации LDAP-подключения:

```
# nano -Y sh /etc/ovirt-engine/aaa/ad.holding.com.properties
```

Пример готового работоспособного в моём случае файла:



```

include = <ad.properties>

# Active Directory domain name.
#
vars.domain = ad.holding.com

# Search user and its password.
#
pool.default.auth.simple.bindDN = s-oVirt-LS@${global:vars.domain}
pool.default.auth.simple.password = SuPeRbP4ssW0rd

# LDAP-servers autosearch deactivating
#
pool.default.dc-resolve.enable = false
search.default.dc-resolve.enable = false

# BaseDN for LDAP search
#
search.ad-resolve-upn.search-request.baseDN = DC=ad,DC=holding,DC=com

# LDAP-servers
#
pool.default.serverset.type = failover
pool.default.serverset.failover.00.server = kom-dc01.${global:vars.domain}
pool.default.serverset.failover.01.server = kom-dc02.${global:vars.domain}
pool.default.serverset.failover.port = 636
pool.default.serverset.failover.domain = ${global:vars.domain}

# Settings for startTLS over TCP 389
#
#pool.default.ssl.startTLS = true
#pool.default.ssl.startTLSProtocol = TLSv1.2
#pool.default.ssl.truststore.file = ${local:_basedir}/${global:vars.domain}.jks
#pool.default.ssl.truststore.password = changeit

# Settings for TLS over TCP 636
#
pool.default.ssl.enable = true
pool.default.ssl.protocol = TLSv1.2
pool.default.ssl.truststore.file = ${local:_basedir}/${global:vars.domain}.jks
pool.default.ssl.truststore.password = changeit

```

Несколько комментариев по поводу содержимого:

- В параметре **vars.domain** - FQDN имя нашего домена AD. В моём случае используется один из под-доменов леса AD.
- В параметрах **pool.default.auth.simple** – учётные данные доменного пользователя для подключения к LDAP-каталогу и операций поиска. Этого пользователя мы сделали ранее. В конечном итоге, с учётом доменной части, полное значение этого параметра будет формироваться в виде UPN учётной записи сервисного пользователя.



- Параметры **pool.default.dc-resolve.enable** и **search.default.dc-resolve.enable** отвечают за отключение назойливого механизма, пытающегося самостоятельно получить имена LDAP-серверов из сервисных учётных записей в DNS. Несмотря на то, что после нескольких попыток мне удалось заставить работать механизм сужения списка контроллеров домена до тех, которые можно считать ближайшими по топологии сайтов AD, встроенные механизмы авто-обнаружения продолжали чинить мне всяческие препятствия. Например, в пул LDAP-серверов при обработке запросов упорно добавлялись контроллеры корневого домена, что в результате ломало весь поиск, так как в моём случае поиск учётных записей должен был выполняться исключительно в конкретном ресурсном под-домене.
  - Параметр **search.ad-resolve-upn.search-request.baseDN** во первых позволяет сузить границы поиска в крупных доменах, во вторых решает проблему при использовании поиска в под-домене (см.предыдущий пункт).
  - В параметрах **pool.default.serverset** указываем конкретные удобные для нас LDAP-серверы (контроллеры домена с ролью Global Catalog), порт подключения и имя нужного домена. Явное указание LDAP-серверов в данном случае может показаться не самым изящным решением, однако, как я заметил выше, позволяет однозначно избежать проблем авто-обнаружения.
  - В параметрах **pool.default.ssl** при необходимости задаём настройки для защиты LDAP соединения. Несмотря на то, что из документации Red Hat можно понять, что рекомендуемым вариантом для защиты LDAP-соединений является **startTLS**, практические упражнения со сниффером показали, что в случае выбора **LDAPS** с подключением на выделенный для SSL соединений порт (по умолчанию TCP 636) на стороне контроллера домена AD, трафик соединений шифруется более полно и работают такие соединения субъективно пошустрей.
- Здесь же, обратите внимание на то, что параметр **pool.default.ssl.truststore.file** должен указывать на ранее созданный нами **jks** файл с корневыми сертификатами доменного ЦС. В параметре **pool.default.ssl.truststore.password** соответственно указывается пароль, который был ранее использован нами при генерации этого jks файла.

Дополнительную информацию о массе других поддерживаемых параметров и возможных вариантах настройки файла свойств конфигурации LDAP-подключения можно найти в файле справки README.profile:

```
# locate README.profile
/usr/share/doc/ovirt-engine-extension-aaa-ldap-1.2.1/README.profile
```

Либо в онлайн-варианте по ссылке [GitHub - oVirt - ovirt-engine-extension-aaa-ldap - README.profile](#).

\*\*\*

Отредактируем другие два файла параметров профиля интеграции. Изменения, которые нужно сделать в этих файлах, сводятся к замене всех названий профиля "profile1" на заданное нами ранее название профиля (в нашем случае это "ad.holding.com").

Редактируем файл параметров аутентификации профиля интеграции:

```
# nano -Y sh /etc/ovirt-engine/extensions.d/ad.holding.com-authn.properties
```

Пример готового файла:

```
ovirt.engine.extension.name = ad.holding.com-authn
ovirt.engine.extension.bindings.method = jbossmodule
ovirt.engine.extension.binding.jbossmodule.module = org.ovirt.engine-
extensions.aaa.ldap
ovirt.engine.extension.binding.jbossmodule.class =
org.ovirt.engineextensions.aaa.ldap.AuthnExtension
ovirt.engine.extension.provides = org.ovirt.engine.api.extensions.aaa.Authn
ovirt.engine.aaa.authn.profile.name = ad.holding.com
ovirt.engine.aaa.authn.authz.plugin = ad.holding.com-authz
config.profile.file.1 = ../aaa/ad.holding.com.properties
```

Редактируем файл параметров авторизации профиля интеграции:

```
# nano -Y sh /etc/ovirt-engine/extensions.d/ad.holding.com-authz.properties
```

Пример готового файла:

```
ovirt.engine.extension.name = ad.holding.com-authz
ovirt.engine.extension.bindings.method = jbossmodule
ovirt.engine.extension.binding.jbossmodule.module = org.ovirt.engine-
extensions.aaa.ldap
ovirt.engine.extension.binding.jbossmodule.class =
org.ovirt.engineextensions.aaa.ldap.AuthzExtension
ovirt.engine.extension.provides = org.ovirt.engine.api.extensions.aaa.Authz
config.profile.file.1 = ../aaa/ad.holding.com.properties
```

\*\*\*

Теперь изменим права доступа на файлы профиля интеграции следующим образом:

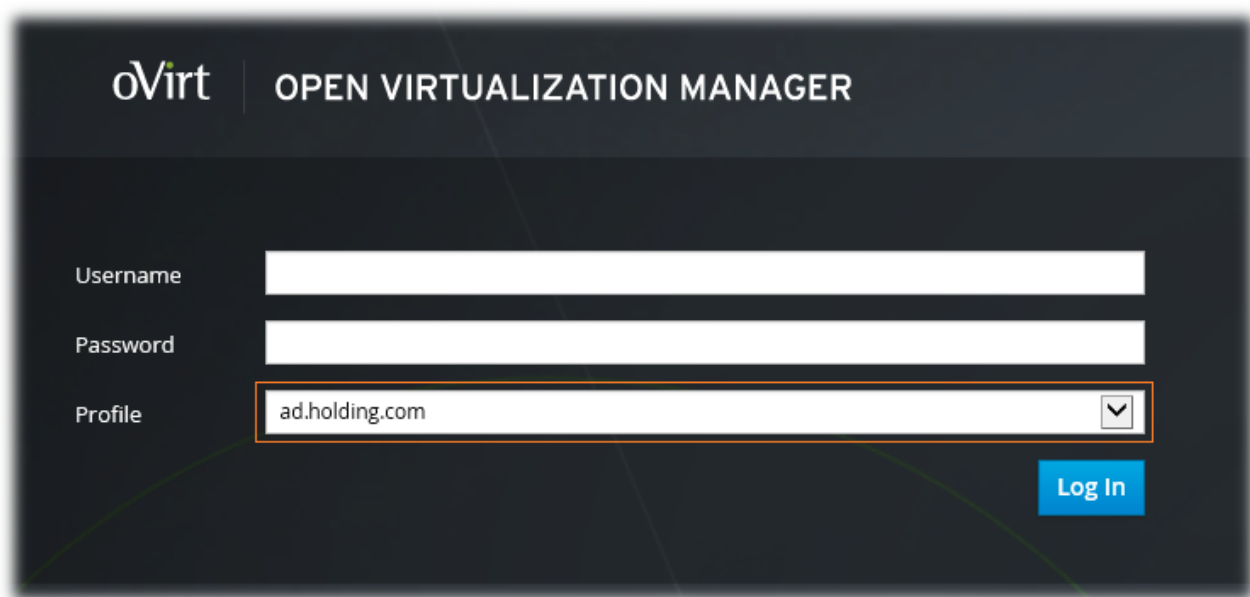
```
# chown ovirt:ovirt /etc/ovirt-engine/extensions.d/ad.holding.com-authn.properties
# chown ovirt:ovirt /etc/ovirt-engine/extensions.d/ad.holding.com-authz.properties
# chown ovirt:ovirt /etc/ovirt-engine/aaa/ad.holding.com.properties

# chmod 600 /etc/ovirt-engine/extensions.d/ad.holding.com-authn.properties
# chmod 600 /etc/ovirt-engine/extensions.d/ad.holding.com-authz.properties
# chmod 600 /etc/ovirt-engine/aaa/ad.holding.com.properties
```

Ну и наконец, чтобы oVirt Engine подхватил созданный нами профиль в работу, перезапустим его службу:

```
# service ovirt-engine restart
```

Теперь созданный профиль интеграции должен появиться в качестве дополнительного на веб-страницах входа на порталы oVirt - **Administration Portal** и **User Portal**:



При входе на тот или иной веб-портал, выбранный пользователем профиль будет сохраняться в cookie-файлах его веб-браузера, и поэтому выбор нужного профиля не придётся делать каждый раз.

Перед тем как, начать использование созданного профиля интеграции, желательно выполнить процедуру его проверки.

Проверка профиля интеграции

Для отладки настроек профиля может помочь утилита **ovirt-engine-extensions-tool**, с помощью которой можно выполнять некоторые виды LDAP-запросов и изучать порядок их обработки и получения результатов.

Чтобы протестировать процедуру проверки учётных данных для аутентификации какого-либо доменного пользователя с помощью созданного профиля интеграции выполним команду типа:

```
# ovirt-engine-extensions-tool --log-level={уровень вывода информации} aaa login-user --profile={имя профиля интеграции} --user-name={имя пользователя для проверки}
```

Например, в нашем случае эта команда будет выглядеть вот так:

```
# ovirt-engine-extensions-tool --log-level=INFO aaa login-user --profile=ad.holding.com --user-name=petya
```

В результате выполнения этой команды будет запрошен пароль указанного пользователя и, в случае успешной аутентификации, будет возвращён перечень доменных групп безопасности, участником которых является этот пользователь.

В случае возникновения проблем, для поиска их причины, можно включить режим расширенного логирования, добавив соответствующие параметры вызова утилиты:

```
# ovirt-engine-extensions-tool --log-level=FINEST --log-file=/var/log/mytest.log  
aaa ...
```

Если первый тест проверки учётных данных не прошёл успешно, то стоит начать с проверки работоспособности процедуры поиска объектов в LDAP-каталоге.

Пример проверки поиска учётной записи доменного пользователя:

```
# ovirt-engine-extensions-tool --log-level=INFO aaa authz-fetch_principal_record -  
-extension-name=ad.holding.com-authz --principal-name=petya
```

Пример проверки поиска доменной группы безопасности:

```
# ovirt-engine-extensions-tool --log-level=INFO aaa search --extension-  
name=ad.holding.com-authz --entity=group --entity-name="Domain Admins"
```

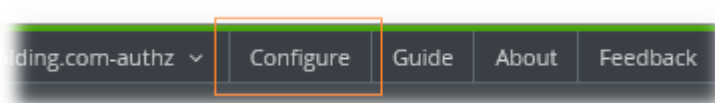
В выводе этих запросов нужно обращать внимание на наличие возвращаемых значений типа **PrincipalRecord** и **GroupRecord**. Дополнительную информацию об использовании этой утилиты можно получить во встроенной справке, либо здесь: [oVirt - Develop - Extension tester tool](#)

После отладки и изменения конфигурации профиля, необходимо перезапускать службу **ovirt-engine**, чтобы профиль интеграции начал свою работу с новым набором параметров. При этом также нужно убедиться в том, что служба перезапускается без ошибок

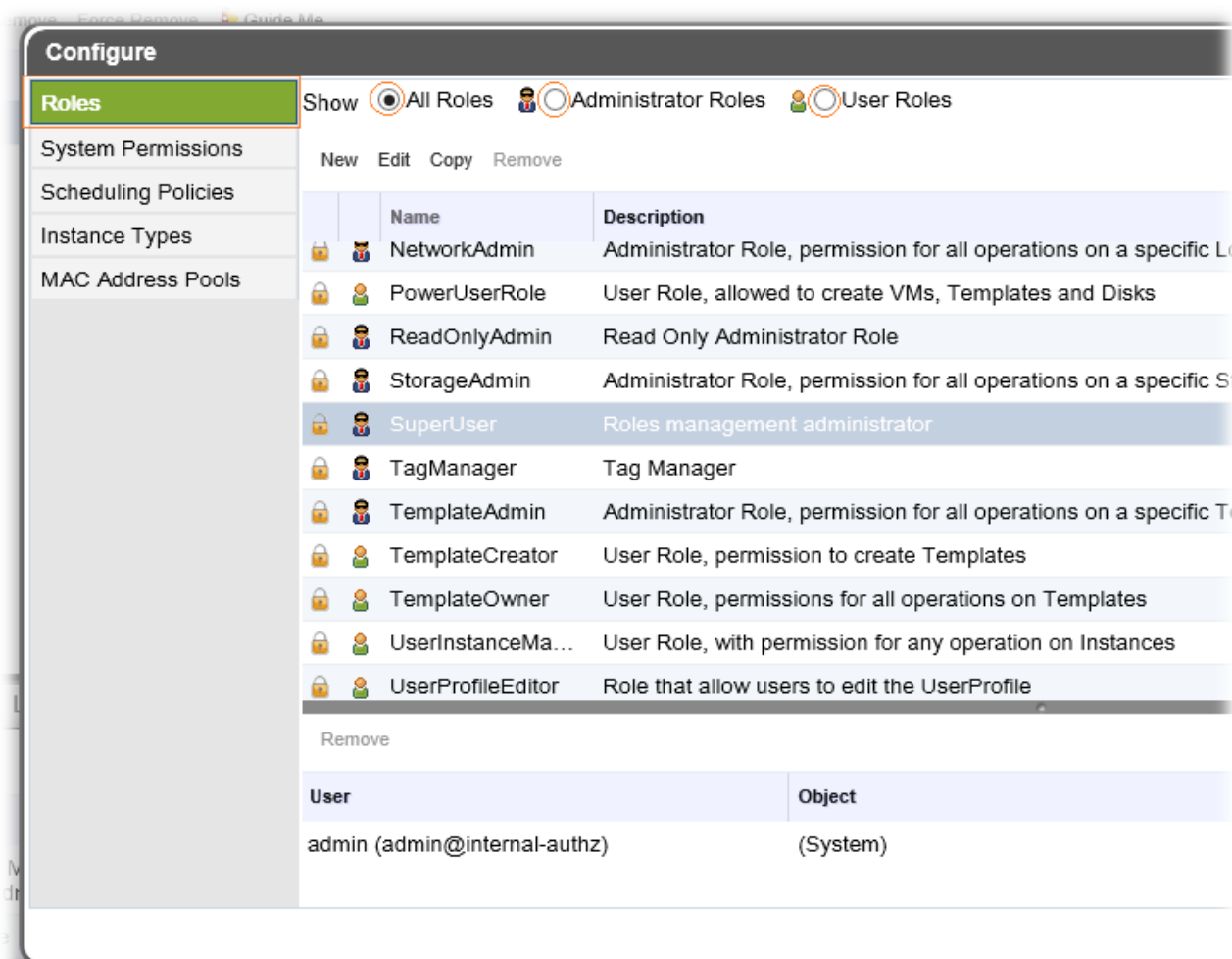
```
# service ovirt-engine restart
```

Настройка ролевой модели прав доступа

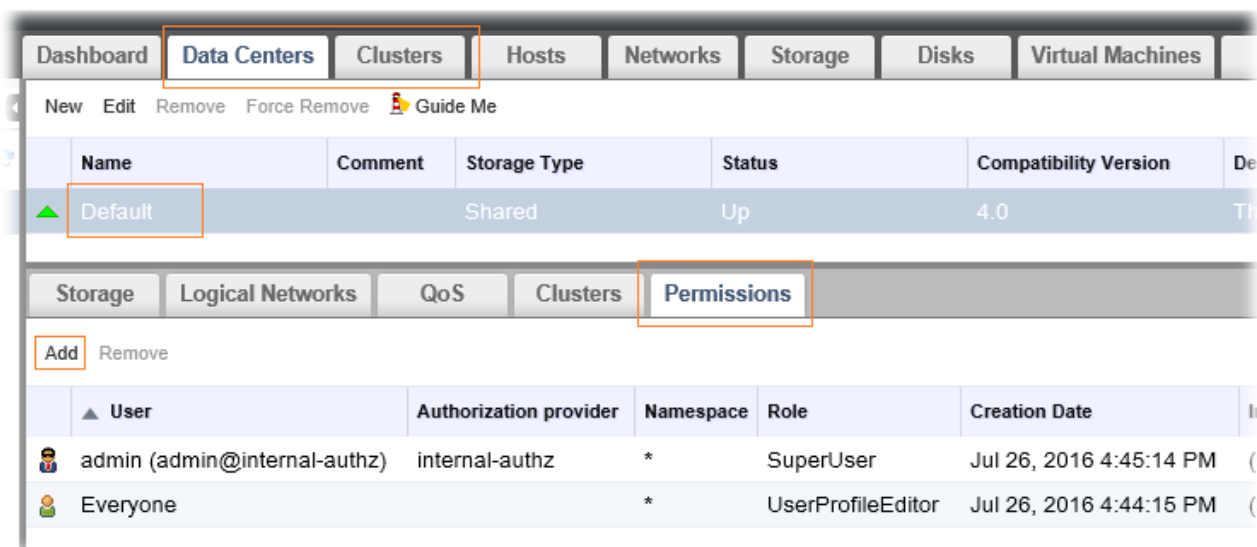
Ранее мы создали профиль интеграции, настроив тем самым транспорт между контроллерами домена AD и oVirt. Для того, чтобы oVirt Engine понимал то, какому залогинившемуся через Active Directory пользователю, какой нужно выдать набор привилегий внутри инфраструктуры oVirt, нам потребуется выдать этому пользователю какой-либо набор прав. Наборы прав в oVirt представляют собой ряд сконфигурированных ролей. При необходимости мы можем создавать собственные роли с нужными тонкими настройками. Изучить список доступных пред-настроенных ролей и создать собственные роли можно в веб-консоли портала администрирования oVirt - в правом верхнем углу пункт **Configure**



На закладке **Roles** мы увидим, что существует два типа ролей – **Administrators Roles** (регулируют доступ к portalу администрирования) и **User Roles** (регулируют доступ к пользовательскому portalу)



Соответствующие роли могут быть использованы для назначения прав доступа как на уровне Дата-Центров, так и на уровне отдельных кластеров oVirt.



При настроенном профиле интеграции oVirt с LDAP-каталогом AD, помимо встроенных в oVirt учётных записей и групп, для назначения прав доступа нам будет доступна возможность использования, как отдельных учётных записей пользователей домена AD, так и групп безопасности. Например, давайте назначим

доменной группе безопасности SRV-ADM-oVirt-Administrators права администраторов oVirt. Для этого выберем тип назначения **Group**, выберем область поиска и зададим маску поиска имени интересующей нас доменной группы

**Add Permission to User**

☐ User ☒ Group ☐ Everyone ☐ My Groups

Search:  Namespace:

Group Name	Display Name
<input checked="" type="checkbox"/> SRV-ADM-oVirt-Administrators	Администраторы инфраструктуры oVirt

Role to Assign:

Отметим найденную группу и в низу форму в выпадающем списке выберем роль oVirt, которую мы хотим назначить для найденной доменной группы безопасности. Как видите, всё предельно просто.

Проверка авторизации на веб-порталах oVirt

Выдав необходимые права доступа для портала администрирования и для пользовательского портала проведём ряд тестов, чтобы убедиться в том, что авторизация работает так, как надо. Например, верно укажем доменные учётные данные доменного пользователя, входящего в ранее назначенную доменную группу администраторов oVirt.

Username:

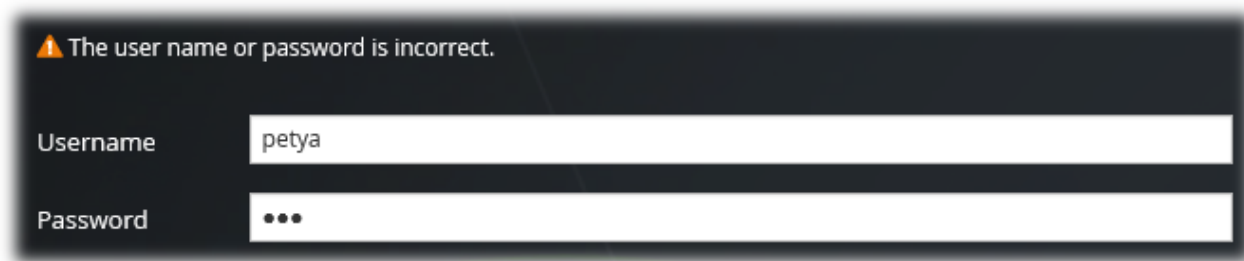
Password:

Profile:

Если все предыдущие настройки выполнены правильно, то пользователь успешной войдёт на портал администрирования, а в верхнем правом углу будет видна информация о его учётной записи с в виде UPN с именем файла авторизации для профиля интеграции:

petya@ad.holding.com@ad.holding.com-authz  [Configure](#) [Guide](#) [About](#) [Feed](#)

Также стоит проверить то, что oVirt правильно обрабатывает ситуации ограничения доступа в случае использования неверных или несуществующих учётных данных, например не пускает пользователя указавшего существующее имя доменного пользователя и заведомом неверный пароль



The screenshot shows a dark-themed login interface. At the top, a red error message with a warning icon reads: "The user name or password is incorrect." Below this, there are two input fields. The first field is labeled "Username" and contains the text "petya". The second field is labeled "Password" and contains three black dots, indicating a masked password.

Или например можно проверить попытку указания правильных учётных данных пользователя, который при этом не имеет никакого доступа в oVirt и т.п.

\*\*\*

На данном этапе после всех успешных проверок можно говорить, что с интеграция oVirt с LDAP-каталогом на базе Microsoft Active Directory выполнена. В следующей части мы рассмотрим процедуру настройки **Single sign-on (SSO)** на базе протокола **Kerberos** для прозрачной аутентификации пользователей на веб-порталах oVirt (без необходимости явного указания имени пользователя и пароля).