Развёртывание и настройка oVirt 4.0. Часть 10. Настройка Single sign-on (SSO) на базе Kerberos для упрощения аутентификации на веб-порталах oVirt

blog.it-kb.ru/2016/10/04/install-ovirt-4-0-part-10-ovirt-engine-apache-sso-single-sign-on-module-for-kerberos-authentication-with-ldap-active-directory

Автор:Алексей Максимов 04.10.2016

В этом части описания мы продолжим тему интеграции **oVirt 4.0** с внешним **LDAP**-каталогом на базе домена **Microsoft Active Directory** (**AD**) и поговорим о настройке механизма **Single sign-on** (**SSO**) средствами протокола **Kerberos** для веб-сервера **Apache** с целью облегчения процедуры аутентификации пользователей при входе на веб-порталы oVirt.

Опорным документом для наших действий будет: RHEV 3.6 Administration Guide - 14.4. Configuring LDAP and Kerberos for Single Sign-on. Действия, описываемые в данной заметке предполагают, что раннее нами уже создан работоспособный профиль интеграции oVirt с LDAP-каталогом на базе каталога Active Directory.

Подготовительные мероприятия

Подготовительные мероприятия, которые нам нужно будет выполнить перед настройкой SSO Kerberos на веб-сервере Apache, на базе которого работают веб-порталы oVirt:

- Настройка синхронизации времени между oVirt Engine и контроллерами домена AD
- Создание в домене AD сервисной учётной записи
- Создание keytab-файла для сервисной учётной записи
- Установка и настройка Kerberos-клиента на сервере oVirt Engine

Рассмотрим эти действия последовательно более подробно.

Настройка синхронизации времени между oVirt Engine и контроллерами домена AD

Наличие корректно работающей синхронизации времени между всеми участниками сетевого обмена (клиентские компьютеры, контроллеры домена, сервер oVirt Engine) необходимо для правильной работы протокола **Kerberos**. В инфраструктуре AD это требование реализуется довольно просто, так как контроллеры домена имеют функции **NTP-сервера** и используются в качестве источника точного времени, то есть предполагается, что как клиентские компьютеры, так и сервер oVirt Engine синхронизирует время с контроллерами домена.

Проверим состояние имеющейся по умолчанию в **CentOS 7** службы **chronyd**, которая реализует функции **NTP-клиента** для синхронизации времени с внешними источниками.

```
# systemctl status chronyd
```

Как правило, эта служба запускается и настраивается ещё в процессе развёртывания ОС. Если по какой-то причине служба не запущена и не настроена, выполним её настройку, указав в файле /etc/chrony.conf в качестве источников синхронизации времени контроллеры домена AD:

```
# cat /etc/chrony.conf | grep ^[^#\;]

server 10.1.0.9 iburst
server 10.1.6.8 iburst
stratumweight 0
driftfile /var/lib/chrony/drift
rtcsync
makestep 10 3
bindcmdaddress 127.0.0.1
keyfile /etc/chrony.keys
commandkey 1
generatecommandkey
noclientlog
logchange 0.5
logdir /var/log/chrony
```

После этого запустим службу и включим её автозапуск в процессе загрузки системы:

```
# systemctl start chronyd
# systemctl enable chronyd
```

Проверяем статус синхронизации времени:

```
# chronyc tracking

Reference ID    : 10.1.0.9 (kom-dc01.ad.holding.com)
Stratum         : 4
Ref time (UTC)  : Mon Sep 26 17:57:33 2016
System time     : 0.000602990 seconds slow of NTP time
Last offset     : -0.000166431 seconds
RMS offset      : 0.001643001 seconds
Frequency       : 13.628 ppm slow
Residual freq   : -0.004 ppm
Skew            : 0.157 ppm
Root delay      : 0.078126 seconds
Root dispersion : 0.125185 seconds
Update interval : 1036.9 seconds
Leap status     : Normal
```

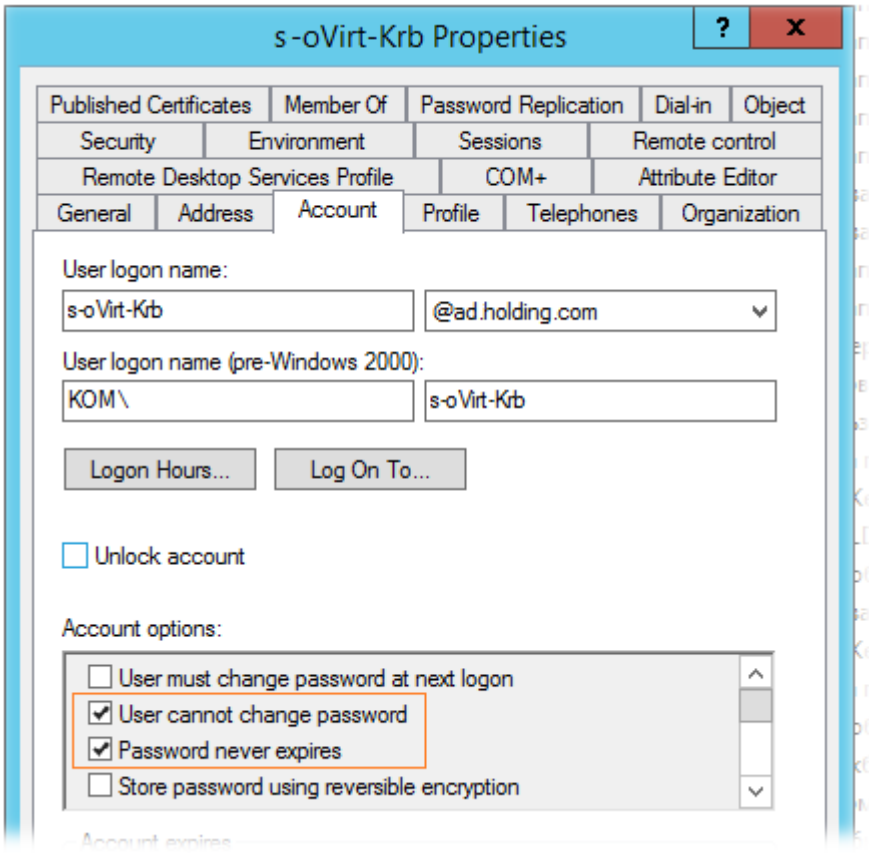
В нашем случае время в формате UTC успешно синхронизировано с одного из указанных ранее в конфигурации контроллеров домена. Если в поле **Stratum** отображается **16**, значит синхронизация с источником работает неправильно.

Дополнительную информацию по работе chrony можно получить, например, в документе RHEL7 System Administrators Guide - Using chrony.

Дополнительно у меня была мысль о том, что можно отключить синхронизацию времени виртуальной машины с хостом виртуализации, однако здесь меня заверили в том, что при настроенном и правильно работающем NTP-клиенте можно не опасаться проблемы рассинхронизации времени.

Создание в домене AD сервисной учётной записи

Создадим в домене AD сервисную учетную запись пользователя для работы Kerberos аутентификации в веб-сервере **Apache**, на котором работают веб-порталы oVirt. В нашем примере это будет учетная запись s-oVirt-Krb. Отключим для учетной записи требование периодической смены пароля (**Password never expires**):



Для данной учётной записи в домене не нужны никакие особенные права. Более того, я предпочитаю все сервисные учётные записи исключать из группы **Domain Users**, в которую учётная запись включается по умолчанию при создании.

Создание keytab-файла для сервисной учётной записи

Для созданной учетной записи нам нужно сгенерировать **keytab**-файл на контроллере домена AD (в нашем случае на базе **Windows Server 2012 R2**) с помощью утилиты **ktpass** в следующем порядке:

```
ktpass -princ HTTP/{FQDN сервера oVirt Engine}@{FQDN домена в верхнем регистре} –
mapuser {имя сервисного пользователя} -pass "{пароль сервисного пользователя}" -
crypto All -ptype KRB5_NT_PRINCIPAL –out {полный путь к создаваемому keytab-файлу}
```

В нашем примере команда будет выглядеть так:

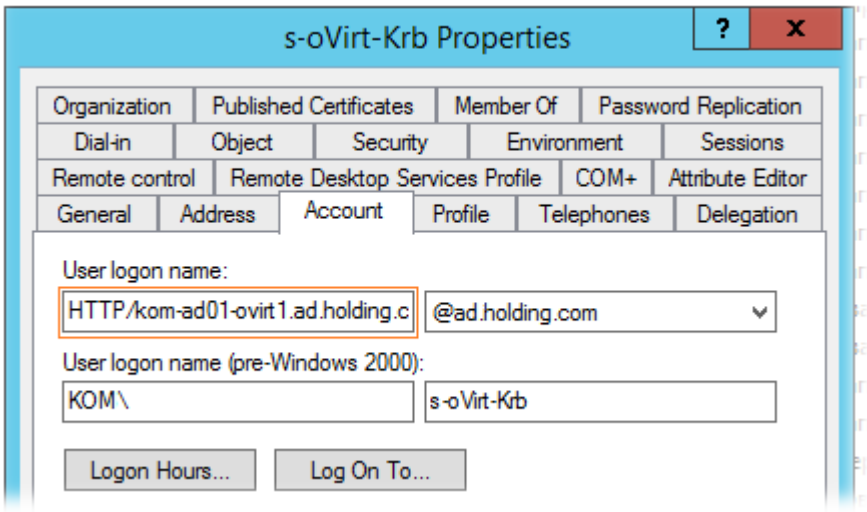
```
ktpass -princ HTTP/kom-ad01-ovirt1.ad.holding.com@AD.HOLDING.COM -mapuser KOM\s-oVirt-Krb -pass "sTr0ngP&ssw0rD" -crypto All -ptype KRB5_NT_PRINCIPAL -out C:\Temp\s-oVirt-Krb.keytab
```

Результат выполнения команды будет выглядеть примерно так:

```
Targeting domain controller: KOM-DC01.ad.holding.com
Successfully mapped HTTP/kom-ad01-ovirt1.ad.holding.com to s-oVirt-Krb.
Password successfully set!
Key created.
Key created.
Key created.
Key created.
Key created.
Output keytab to C:\Temp\s-oVirt-Krb.keytab:
Keytab version: 0x502
keysize 85 HTTP/kom-ad01-ovirt1.ad.holding.com@AD.HOLDING.COM ptype 1
(KRB5_NT_PRINCIPAL) vno 5 etype 0x1 (DES-CBC-CRC) keylength 8 (0x...)
keysize 85 HTTP/kom-ad01-ovirt1.ad.holding.com@AD.HOLDING.COM ptype 1
(KRB5_NT_PRINCIPAL) vno 5 etype 0x3 (DES-CBC-MD5) keylength 8 (0x...)
keysize 93 HTTP/kom-ad01-ovirt1.ad.holding.com@AD.HOLDING.COM ptype 1
(KRB5_NT_PRINCIPAL) vno 5 etype 0x17 (RC4-HMAC) keylength 16 (0x....)
keysize 109 HTTP/kom-ad01-ovirt1.ad.holding.com@AD.HOLDING.COM ptype 1
(KRB5_NT_PRINCIPAL) vno 5 etype 0x12 (AES256-SHA1) keylength 32 (0x.....)
keysize 93 HTTP/kom-ad01-ovirt1.ad.holding.com@AD.HOLDING.COM ptype 1
(KRB5_NT_PRINCIPAL) vno 5 etype 0x11 (AES128-SHA1) keylength 16 (0x....)
```

Обратите внимание на то, что для имени веб-сервера **oVirt Engine** kom-ad01-ovirt1.ad.holding.com в **DNS** должна присутствовать статическая **A**-запись (не PTR-запись), чтобы это имя могли правильно разрешать в IP адрес все клиенты Kerberos.

В процессе генерации keytab-файла в домене в свойствах учетной записи пользователя изменится имя для входа, а также будет изменён атрибут **servicePrincipalName** (будет добавлено значение из параметра -princ ранее выполненной команды **ktpass**)



Копируем получившийся файл s-oVirt-Krb.keytab с компьютера под управлением Windows на сервер oVirt Engine в каталог /etc/httpd/, например с помощью утилиты PSCP:

```
pscp -scp C:\Temp\s-oVirt-Krb.keytab root@kom-ad01-ovirt1:/etc/httpd/
```

Далее выставляем разрешения на keytab-файл таким образом, чтобы ограниченный доступ к файлу имела только служба веб-сервера Apache:

```
# chown apache /etc/httpd/s-oVirt-Krb.keytab
# chmod 400 /etc/httpd/s-oVirt-Krb.keytab
```

Установка и настройка Kerberos-клиента на сервере oVirt Engine

Установим клиентскую часть поддержки реализации **MIT Kerberos 5** на сервере oVirt Engine:

```
# yum info krb5-workstation
...
Available Packages
Name        : krb5-workstation
Arch        : x86_64
Version     : 1.13.2
Release     : 12.el7_2
Size        : 765 k
Repo        : updates/7/x86_64
Summary     : Kerberos 5 programs for use on workstations
URL         : http://web.mit.edu/kerberos/www/
License     : MIT
Description : Kerberos is a network authentication system. The krb5-workstation
            : package contains the basic Kerberos programs (kinit, klist,
kdestroy,
            : kpasswd). If your network uses Kerberos, this package should be
            : installed on every workstation.

# yum -y install krb5-workstation
```

Настроим конфигурационный файл для работы Kerberos-клиента:

```
# nano /etc/krb5.conf
```

В моём случае настроенный файл будет выглядеть следующим образом:

```
[libdefaults]
    default_realm = AD.HOLDING.COM
    dns_lookup_kdc = no
    dns_lookup_realm = no
    ticket_lifetime = 24h
    renew_lifetime = 7d
    forwardable = true
    rdns = false
    default_ccache_name = KEYRING:persistent:%{uid}

    # Encryption types for Windows 2008 and newer
    default_tgs_enctypes = aes256-cts-hmac-sha1-96 rc4-hmac des-cbc-crc des-cbc-
md5
    default_tkt_enctypes = aes256-cts-hmac-sha1-96 rc4-hmac des-cbc-crc des-cbc-
md5
    permitted_enctypes = aes256-cts-hmac-sha1-96 rc4-hmac des-cbc-crc des-cbc-md5

[realms]
    AD.HOLDING.COM = {
        kdc = kom-dc01.ad.holding.com
        kdc = kom-dc02.ad.holding.com
        admin_server = kom-dc01.ad.holding.com
        default_domain = ad.holding.com
    }

[domain_realm]
    .ad.holding.com = AD.HOLDING.COM
    ad.holding.com = AD.HOLDING.COM
```

Проверим, правильно ли настроен наш Kerberos-клиент - попробуем аутентифицировать в домене AD какого-нибудь доменного пользователя с помощью утилиты **kinit**:

```
# kinit kuzya

Password for kuzya@AD.HOLDING.COM:
```

Утилита kinit запросит пароль указанного пользователя, введём его. Никаких ошибок при аутентификации быть не должно.

Теперь с помощью утилиты **klist** проверим кэш билетов Kerberos и убедимся в наличии билета для аутентифицированного пользователя:

```
# klist

Ticket cache: KEYRING:persistent:0:0
Default principal: kuzya@AD.HOLDING.COM

Valid starting       Expires              Service principal
09/29/2016 20:30:04  09/30/2016 06:30:04  krbtgt/AD.HOLDING.COM@AD.HOLDING.COM
        renew until 10/06/2016 20:29:59
```

После этого очистим кэш командой **kdestroy**:

```
# kdestroy
# klist

klist: Credentials cache keyring 'persistent:0:0' not found
```

Будем считать, что наш Kerberos-клиент работает.

***

Теперь выполним проверку возможности Kerberos-аутентификации служебного пользователя в домене с помощью keytab-файла (команда должна отработать без ошибок):

```
# kinit -V -k -t /etc/httpd/s-oVirt-Krb.keytab HTTP/kom-ad01-ovirt1.ad.holding.com

Using default cache: persistent:0:0
Using principal: HTTP/kom-ad01-ovirt1.ad.holding.com@AD.HOLDING.COM
Using keytab: /etc/httpd/s-oVirt-Krb.keytab
Authenticated to Kerberos v5
```

Как видим, аутентификация прошла успешно. Снова заглянем в кэш билетов Kerberos и убедимся в наличии билета для аутентифицированного с помощью keytab-файла служебного пользователя:

```
# klist

Ticket cache: KEYRING:persistent:0:0
Default principal: HTTP/kom-ad01-ovirt1.ad.holding.com@AD.HOLDING.COM

Valid starting       Expires              Service principal
09/29/2016 23:18:26  09/30/2016 09:18:26  krbtgt/AD.HOLDING.COM@AD.HOLDING.COM
        renew until 10/06/2016 23:18:26
```

Необходимые проверки выполнены успешно и теперь можно переходить к настройке поддержки Kerberos в конфигурации веб-сервера Apache.

Подключаем SSO-модуль к веб-серверу Apache

К ранее установленному пакету **ovirt-engine-extension-aaa-ldap** дополнительно установим пару необходимых пакетов:

```
# yum -y install ovirt-engine-extension-aaa-misc mod_auth_kerb
```

Скопируем рекурсивно шаблонные файлы конфигурации oVirt и модуля SSO для веб-сервера Apache в каталог рабочих конфигурационных файлов:

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

Создадим символическую ссылку на SSO модуль для веб-сервера Apache в каталоге /etc/httpd/conf.d, подключив тем самым модуль к Apache:

```
# ln -s /etc/ovirt-engine/aaa/ovirt-sso.conf /etc/httpd/conf.d
```

Отредактируем содержимое SSO модуля для веб-сервера Apache:

```
# nano /etc/ovirt-engine/aaa/ovirt-sso.conf
```

В файле, как минимум, нужно изменить параметры **Krb5Keytab** и **KrbAuthRealms**. В нашем примере результирующий файл будет выглядеть следующим образом:

```
<LocationMatch ^/ovirt-engine/sso/(interactive-login-negotiate|oauth/token-http-auth)|^/ovirt-engine/api>
  <If "req('Authorization') !~ /^(Bearer|Basic)/i">
    RewriteEngine on
    RewriteCond %{LA-U:REMOTE_USER} ^(.*)$
    RewriteRule ^(.*)$ - [L,NS,P,E=REMOTE_USER:%1]
    RequestHeader set X-Remote-User %{REMOTE_USER}s
    AuthType Kerberos
    AuthName "Kerberos Login"
    Krb5Keytab /etc/httpd/s-oVirt-Krb.keytab
    KrbAuthRealms AD.HOLDING.COM
    KrbMethodK5Passwd off
    Require valid-user
    ErrorDocument 401 "<html><meta http-equiv=\"refresh\" content=\"0; url=/ovirt-engine/sso/login-unauthorized\"/><body><ahref=\"/ovirt-engine/sso/login-unauthorized\">Here</a></body></html>"
  </If>
</LocationMatch>
```

Здесь ещё стоит отдельное внимание обратить на то, что на данный момент шаблонный файл /usr/share/ovirt-engine-extension-aaa-ldap/examples/ad-sso/aaa/ovirt-sso.conf, который мы взяли за основу адаптирован под использование в **oVirt 3.6**, но не будет работать в **oVirt 4.0**. Вышеприведённый же пример уже "причёсан" под версию **oVirt 4.0**.

Переименуем другие два файла, которые мы ранее скопировали из каталога /usr/share/ovirt-engine-extension-aaa-ldap/examples/ad-sso/extensions.d/, заменив в имени файла "profile1" на имя нашего профиля интеграции с LDAP, который мы создавали ранее:

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

Отредактируем файл конфигурации аутентификации (не путать с ранее созданным файлом ad.holding.com-authn.properties):

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

По аналогии заменим значения "profile1" на имя ранее созданного профиля интеграции:

```
ovirt.engine.extension.name = ad.holding.com-http-authn
ovirt.engine.extension.bindings.method = jbossmodule
ovirt.engine.extension.binding.jbossmodule.module = org.ovirt.engine-
extensions.aaa.misc
ovirt.engine.extension.binding.jbossmodule.class =
org.ovirt.engineextensions.aaa.misc.http.AuthnExtension
ovirt.engine.extension.provides = org.ovirt.engine.api.extensions.aaa.Authn
ovirt.engine.aaa.authn.profile.name = ad.holding.com-http
ovirt.engine.aaa.authn.authz.plugin = ad.holding.com-authz
ovirt.engine.aaa.authn.mapping.plugin = ad.holding.com-http-mapping
config.artifact.name = HEADER
config.artifact.arg = X-Remote-User
```

Отредактируем mapping-файл:

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

Здесь достаточно заменить значение "profile1" лишь в первой строке:

```
ovirt.engine.extension.name = ad.holding.com-http-mapping
ovirt.engine.extension.bindings.method = jbossmodule
ovirt.engine.extension.binding.jbossmodule.module = org.ovirt.engine-
extensions.aaa.misc
ovirt.engine.extension.binding.jbossmodule.class =
org.ovirt.engineextensions.aaa.misc.mapping.MappingExtension
ovirt.engine.extension.provides = org.ovirt.engine.api.extensions.aaa.Mapping
config.mapAuthRecord.type = regex
config.mapAuthRecord.regex.mustMatch = true
config.mapAuthRecord.regex.pattern = ^(?.*?)((\\\\(?@)(?.*?)@.*)|(?@.*))$
config.mapAuthRecord.regex.replacement = ${user}${at}${suffix}${realm}
```

Установим права доступа на файлы:

```
# chmod 600 /etc/ovirt-engine/extensions.d/ad.holding.com-http-authn.properties
# chmod 600 /etc/ovirt-engine/extensions.d/ad.holding.com-http-mapping.properties
# chown ovirt:ovirt /etc/ovirt-engine/extensions.d/ad.holding.com-http-
authn.properties
# chown ovirt:ovirt /etc/ovirt-engine/extensions.d/ad.holding.com-http-
mapping.properties
```

Удалим лишние файлы, которые остались после рекурсивного копирования из каталога /usr/share/ovirt-engine-extension-aaa-ldap/examples/ad-sso/ и не требуются в нашем случае, так как фактически такие файлы уже были настроены ранее:

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

Перезапускаем службы веб-сервера Apache и oVirt Engine:

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

Перезапуск служб должен пройти без ошибок, после чего можно попробовать проверить результат. Если всё сделано верно, то клиенты смогут прозрачно (без явного указания учётных данных) проходить по ссылкам **User Portal** и

Administration Portal с главной стартовой веб-страницы oVirt

***

В ходы выяснения причины неработоспособности изначально настроенной мной конфигурации SSO в мэйл-группе oVirt мне подсказали, что для настройки Kerberos SSO в **Apache** на **CentOS 7** вместо устаревшего модуля **mod_auth_krb** можно использовать более "модную" связку модулей **mod_auth_gssapi**/**mod_sessions**. Для того, чтобы это сделать достаточно сначала установить сами модули из репозитория **EPEL**:

```
# yum install mod_session mod_auth_gssapi
```

А затем подправить модуль конфигурации веб-сервера /etc/ovirt-engine/aaa/ovirt-sso.conf следующим образом:

```
<LocationMatch ^/ovirt-engine/sso/(interactive-login-negotiate|oauth/token-http-auth)|^/ovirt-engine/api>
  <If "req('Authorization') !~ /^(Bearer|Basic)/i">
    RewriteEngine on
    RewriteCond %{LA-U:REMOTE_USER} ^(.*)$
    RewriteRule ^(.*)$ - [L,NS,P,E=REMOTE_USER:%1]
    RequestHeader set X-Remote-User %{REMOTE_USER}s
    #AuthType Kerberos
    AuthType GSSAPI
    AuthName "Kerberos Login"
    #Krb5Keytab /etc/httpd/s-oVirt-Krb.keytab
    #KrbAuthRealms AD.HOLDING.COM
    #KrbMethodK5Passwd off
    GssapiCredStore keytab:/etc/httpd/s-oVirt-Krb.keytab
    GssapiUseSessions On
    Session On
    SessionCookieName ovirt_gssapi_session path=/private;httponly;secure;
    Require valid-user
    ErrorDocument 401 "<html><meta http-equiv=\"refresh\" content=\"0; url=/ovirt-engine/sso/login-unauthorized\"/><body><a href=\"/ovirt-engine/sso/login-unauthorized\">Here</a></body></html>"
</If>
</LocationMatch>
```

После этого также необходимо выполнить перезапуск служб веб-сервера Apache и oVirt Engine и можно проверять результат.

Как я смог убедиться на практике в том, что и вариант с использованием модуля **mod_auth_krb** и вариант с использованием связки модулей **mod_auth_gssapi**/**mod_sessions** работают одинаково ровно с веб-браузерами **Internet Explorer v11**, **Mozilla Firefox ESR v38**, **Google Chrome v49**. Вопрос о выборе какого-либо из вариантов пока остался для меня открытым. Если кто-то приведёт серьёзные аргументы в пользу того или иного варианта, было бы интересно это услышать.