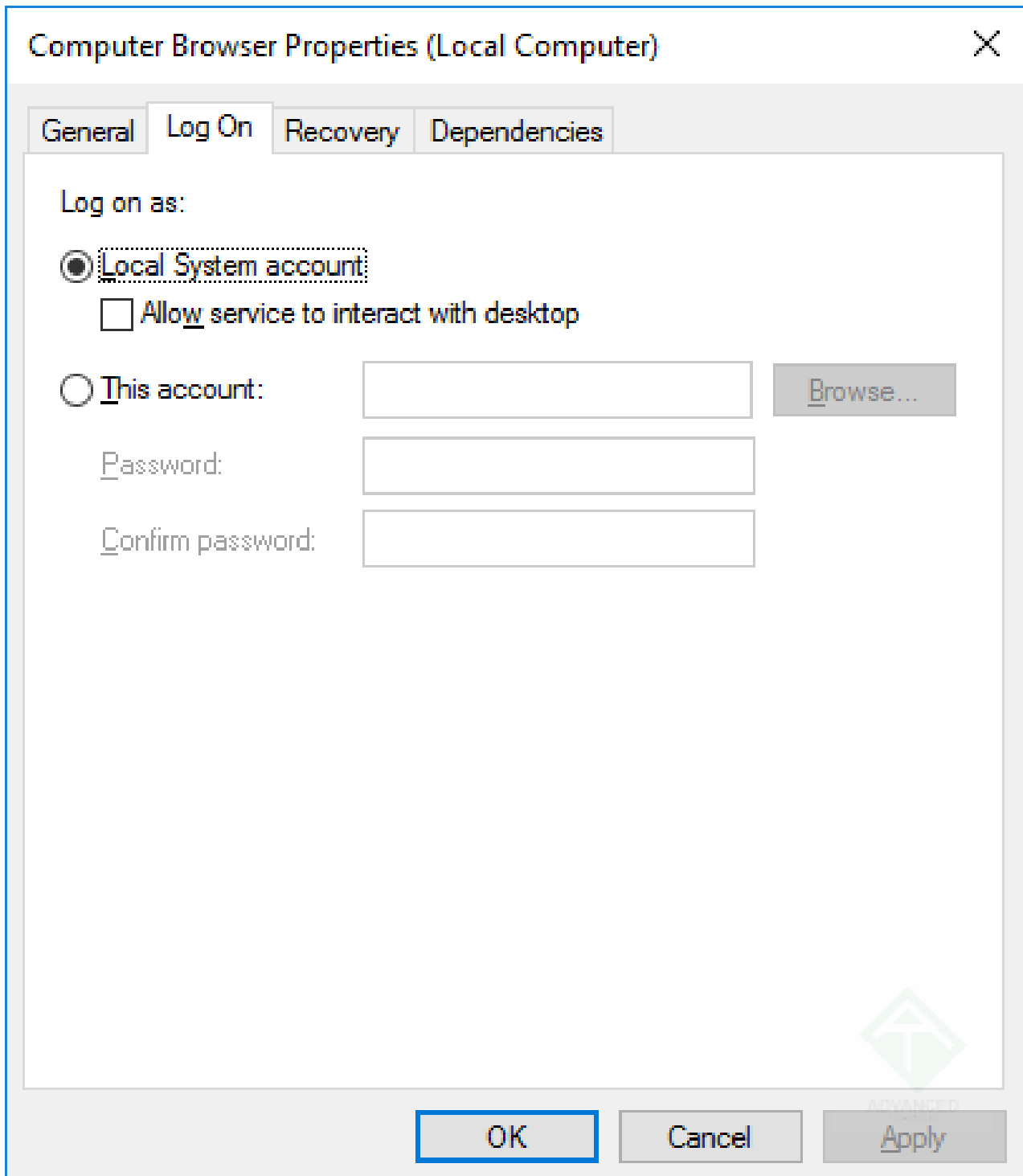


Управляемые учётные записи сервисов - Group Managed Service Accounts в Windows Server 2016

 atraining.ru/group-managed-service-accounts-msa-windows-server-2016

2013-10-03T14:49:06+08:00



Computer Browser Properties (Local Computer) X

General Log On Recovery Dependencies

Log on as:

☒ Local System account

☐ Allow service to interact with desktop

☐ This account: Browse...

Password:

Confirm password:

OK Cancel Apply

Привет.

Ранее я написал [обзорную статью про MSA в Windows Server 2008 R2](#). Теперь, так как вышли новые версии платформы Windows Server, надо добавлять. Ведь есть что, и очень даже интересное. Это – обновлённая версия статьи; оригинальная

была написана по Windows Server 2012 RTM, данная уже учитывает существование и Windows Server 2012 R2, и Windows Server 2016.

Работаем с Managed Service Accounts

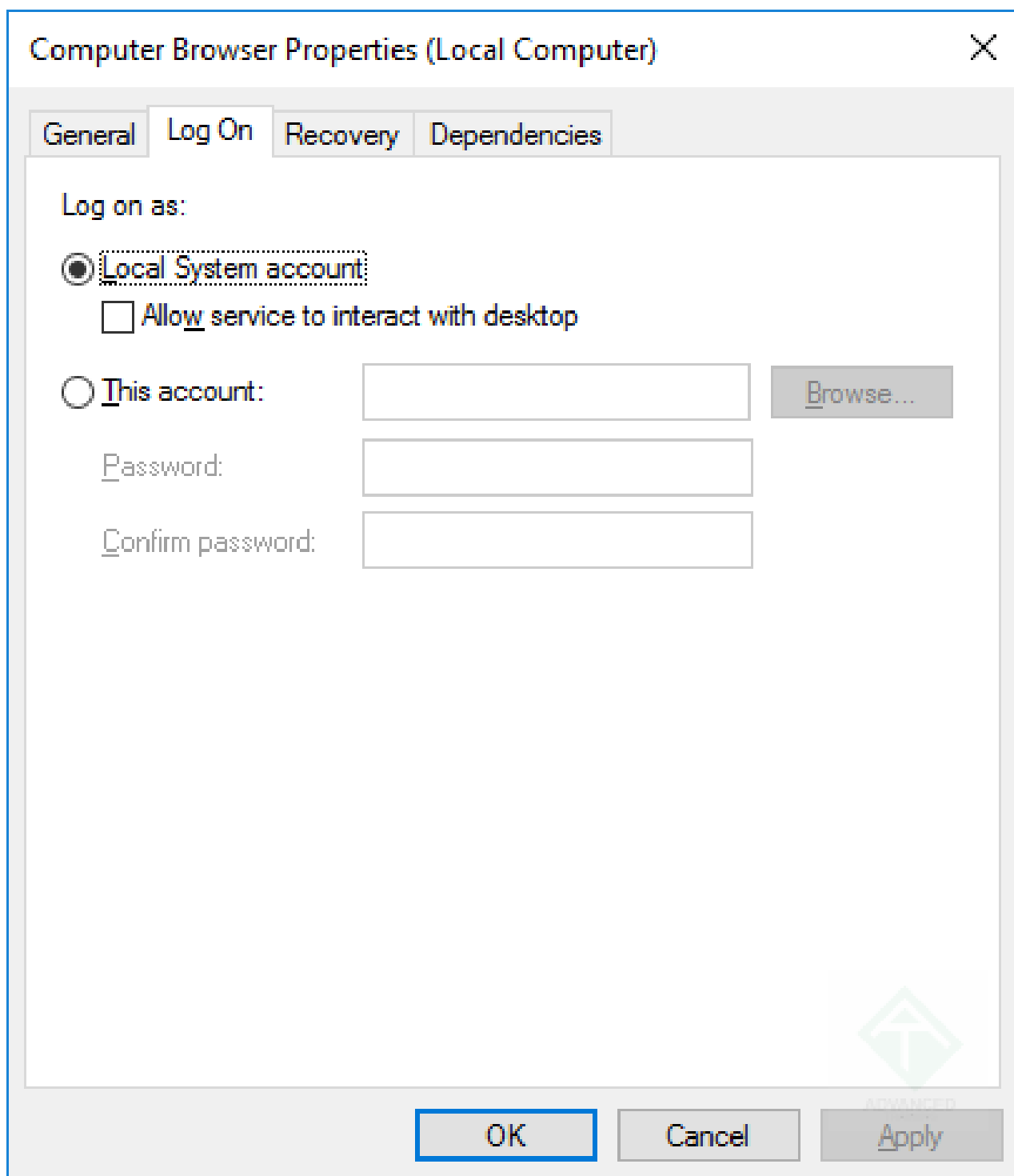
- Зачем нужны MSA
- Функционал MSA – обычных MSA и современных gMSA
- Подготавливаем лес Active Directory к работе с MSA и gMSA
- Включаем и настраиваем Key Distribution Services
- Создаём gMSA на Windows Server 2012 и выше
- Дополнительные возможности

Приступим.

Зачем нужны MSA

Изначально в Windows NT такого понятия, как “специальный вид учётной записи для приложения-сервиса” не существовало. Была системная учётка **SYSTEM**, которая олицетворяла собой Одина, Будду, Ктулху и прочих интересных личностей в одном флаконе. Эта учётная запись была неявно включена в **BUILTIN\Administrators** и обладала всеми правами на системе, которые могут быть – ну а если чем-то не обладала, так от её лица можно было сделать **take ownership** на любом объекте, имеющем ACL, и в результате всё ж обладать всеми правами. От этой учётки работали все системные процессы, а также запускались сервисы – но подчеркну, сделана она была не именно для запуска сервисов, а чтобы олицетворять “всю систему”.

Вы могли запускать сервисы как от неё, так и от обычной учётной записи пользователя – это менялось в настройках примерно так же, как меняется сейчас:



[Пример сервиса, запускающегося от учётной записи SYSTEM](#)
(кликните для увеличения до 406 px на 468 px)

Да и сервисы свои вы тоже могли создавать, для этого была специальная утилита [srvany](#), входившая в состав Resource Kit.

Вроде как всё хорошо, но на самом деле – проблемы.

Первая и очевидная состояла в небезопасности выдачи всех-всех-всех прав в системе любому приложению, которому надо было запускаться на фоне. Допустим, берём сервис DHCP Client. Что ему нужно от системы? Возможность доступа к сети, притом небольшую совсем, да возможность записи конфигурации в некоторые ветки реестра (где хранятся настройки сетевых интерфейсов). Всё, что он делает –

обменивается несколькими видами ethernet-кадров, да пишет Надо ли ему доступ ко всем данным на всех дисках? К созданию пользователей и смене прав? К раздаче системных привилегий типа “shut down from remote system”? Нет конечно, зачем. Но всё это выдаётся, если работаешь от **SYSTEM**.

Вторая состояла в том, что даже если заводишь специальную учётную запись пользователя, от имени которой запускаешь этот сервис, то возникает новая пачка проблем, в том числе:

- Надо старательно выдать этой учётной записи только минимально нужные для этой операции права и запретить изначально разрешаемые ей, но не нужные в данном сценарии использования (например, запретить сетевой вход);
- Надо периодически менять у этой учётной записи пароль, а также сделать его стойким;
- Пароль к этой учётной записи будет храниться в системе в виде открытого текста (надо ж при запуске сервиса симитировать log on as a service, поэтому надо предоставить связку логин-пароль в исходном варианте);
- Пароль может использоваться в нескольких местах (например, когда данная служба на нескольких серверах работает), и доступ к нему возможен на любом из этих серверов от любой учётной записи с правами локального администратора;

Всё это неудобно, небезопасно, заставляет тратить лишнее время, и зачастую сводится к “разово сделаем пароль, поставим галочку, чтобы мозг не конопатил, выдадим права локального админа этой учётке и забудем”.

Managed Service Accounts – MSA (я буду говорить уже именно про новые, gMSA, появившиеся в Windows Server 2012, новая буква g – это Group) – позволяют решить все эти проблемы.

Функционал MSA – обычных MSA и современных gMSA

Managed Service Accounts будут различаться – самый первый вариант, появившийся в Windows Server 2008 R2 (про него [предыдущая статья про MSA](#)), будет уметь работать только с одним сервером, и, так как в следующей же операционной системе, Windows Server 2012, появились gMSA, лишённые этого ограничения, речь пойдёт сразу же про них. Называть я их буду MSA, без буквы g, так проще.

Итак, как MSA решают все вышеперечисленные вопросы?

- Проблема с минимизацией прав упростилась – MSA – это не пользовательская учётка, выкорчёвывать лишние полномочия не нужно, изначально на локальной системе никаких прав у MSA нет, громоздкий профиль, как пользователь, при логине, она тоже не создаёт. Вы просто добавляете её в нужные группы, чтобы предоставить соответствующие задаче права и полномочия.

- Периодичность смены пароля у MSA задаётся через групповые политики и проводится автоматически.
- Пароль у MSA генерится очень стойкий – 240 символов, половина латинские буквы, половина – цифры. Такой пароль нет смысла подбирать bruteforce'ом, потому что это займёт времени больше, чем линейный перебор всех значений хэша.
- При использовании MSA на нескольких серверах проблемы со сменой пароля также нет – всё делается автоматически.

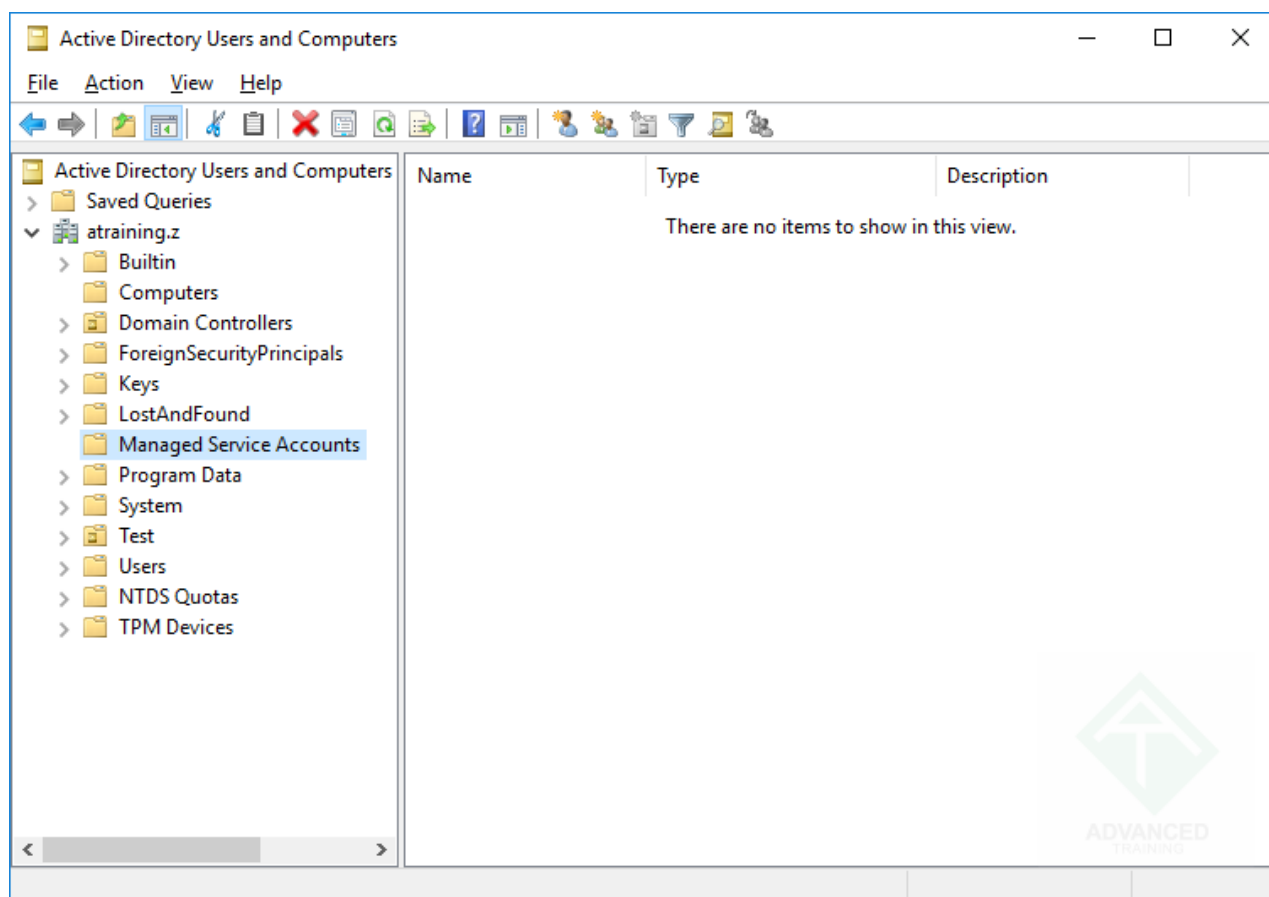
Из дополнительных преимуществ – MSA использует только Kerberos, поэтому хоть [NTLM дополнительно обезопасить можно](#), в случае с MSA это просто не нужно – проблем с безопасностью LM, NTLMv1 и NTLMv2 у этих учёток нет.

В результате получается очень удобный вид учётных записей для упрощения обслуживания и улучшения ситуации с безопасностью.

Давайте внедрять.

Подготавливаем лес Active Directory к работе с MSA и gMSA

Домиком для всех видов MSA – обычных, которые **msDS-ManagedServiceAccount**, и новых, которые **msDS-GroupManagedServiceAccount** – будет контейнер **CN=Managed Service Accounts, DC=контекст домена**, находящийся в корне domain partition вашего домена:

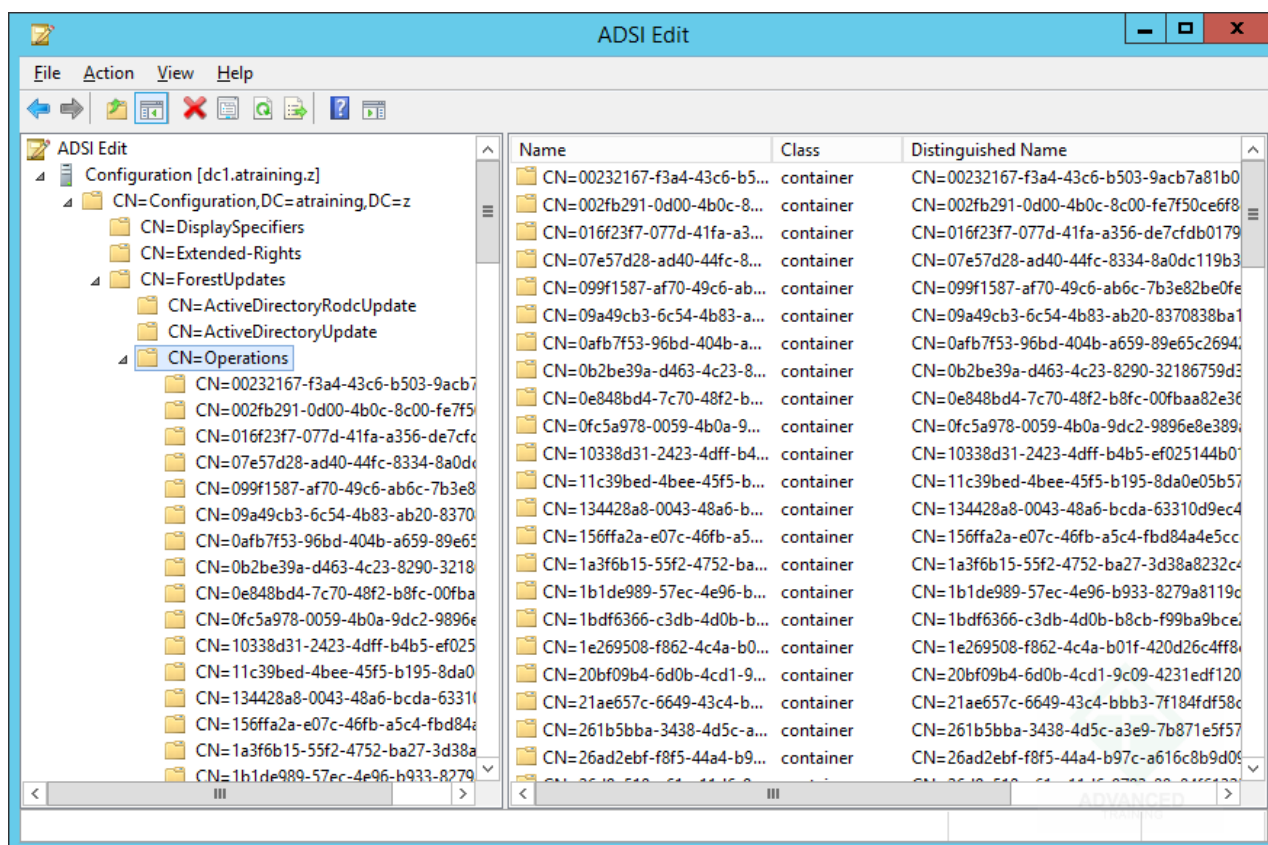


[Контейнер Managed Service Accounts в корне domain partition](#)

[\(кликните для увеличения до 754 px на 530 px\)](#)

Для работы MSA надо будет также включить специальный сервис на выбранном хосте, который будет отвечать за централизованное управление паролями. Этот сервис будет называться KDS – Key Distribution Services.

Необходимые для работы этого сервиса объекты в Active Directory будут создаваться либо сразу, когда поднимается новый лес, либо когда идёт /forestprep при установке первого DC на базе Windows Server 2012. Проверить, что данные операции на уровне леса прошли удачно, можно отследить по трекеу Forest Updates – зайдите в ADSI-редактор и откройте вот такой контейнер:



[Просматриваем Forest Updates в лесу Active Directory](#)

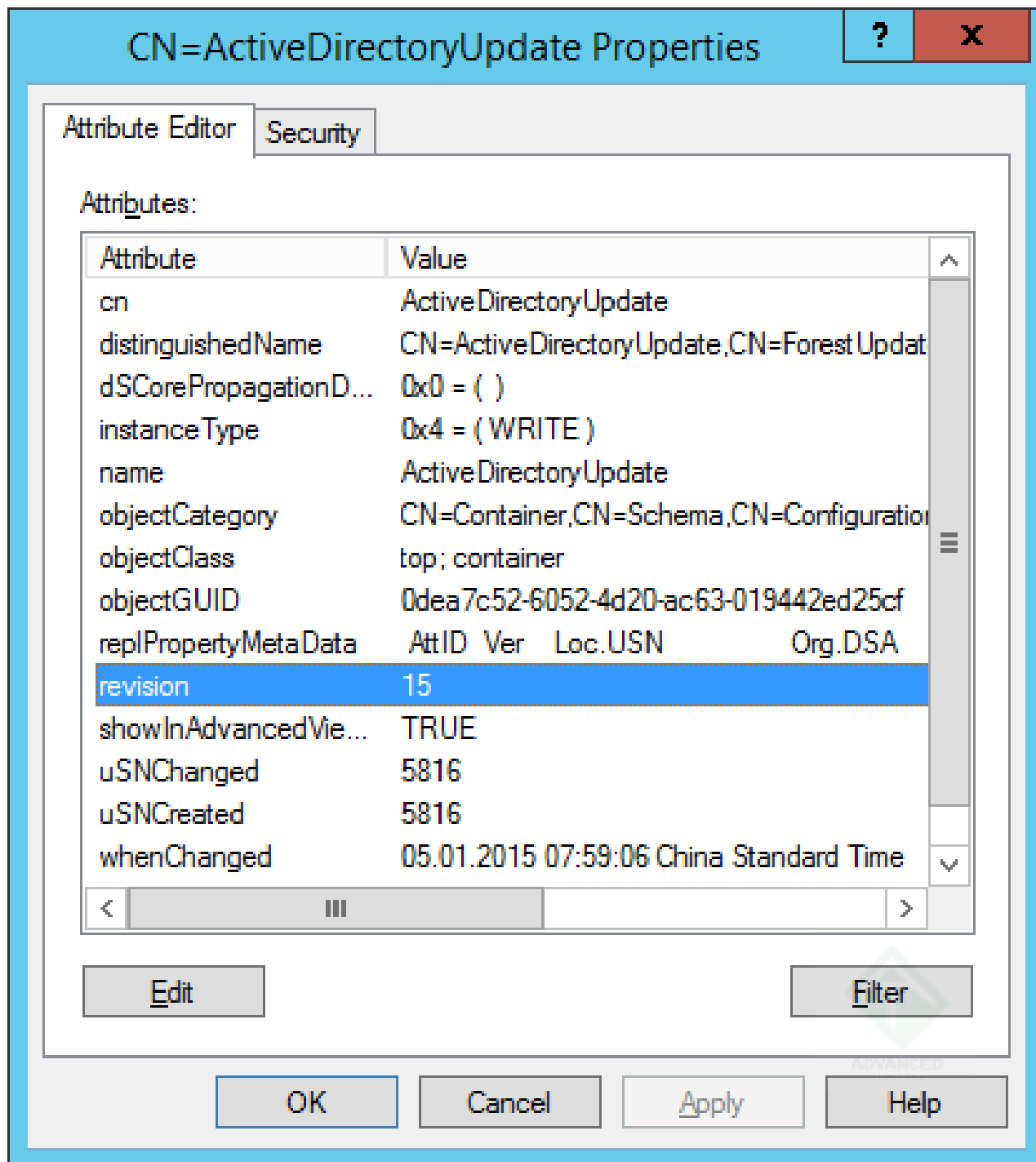
[\(кликните для увеличения до 876 px на 579 px\)](#)

В нём должны (если всё хорошо) быть GUID'ы выполненных операций, 4 штуки:

- {defc28cd-6cb6-4479-8bcb-aabfb41e9713}
- {d6bd96d4-e66b-4a38-9c6b-e976ff58c56d}
- {bb8efc40-3090-4fa2-8a3f-7cd1d380e695}
- {2d6abe1b-4326-489e-920c-76d5337d2dc5}

Там, конечно, много всяких других, благо с 2000го Windows Server в Active Directory изменения есть, но нам надо, чтобы были все эти 4 – это будет обозначать, что дефолтные объекты для работы Key Distribution Services были созданы.

Дополнительно можно глянуть рядом расположенный объект, **CN=ActiveDirectoryUpdate, CN=ForestUpdates, CN=Configuration, DC=atraining, DC=z**. Он будет содержать атрибут **revision** – нам надо, чтобы этот атрибут был как минимум **11**.

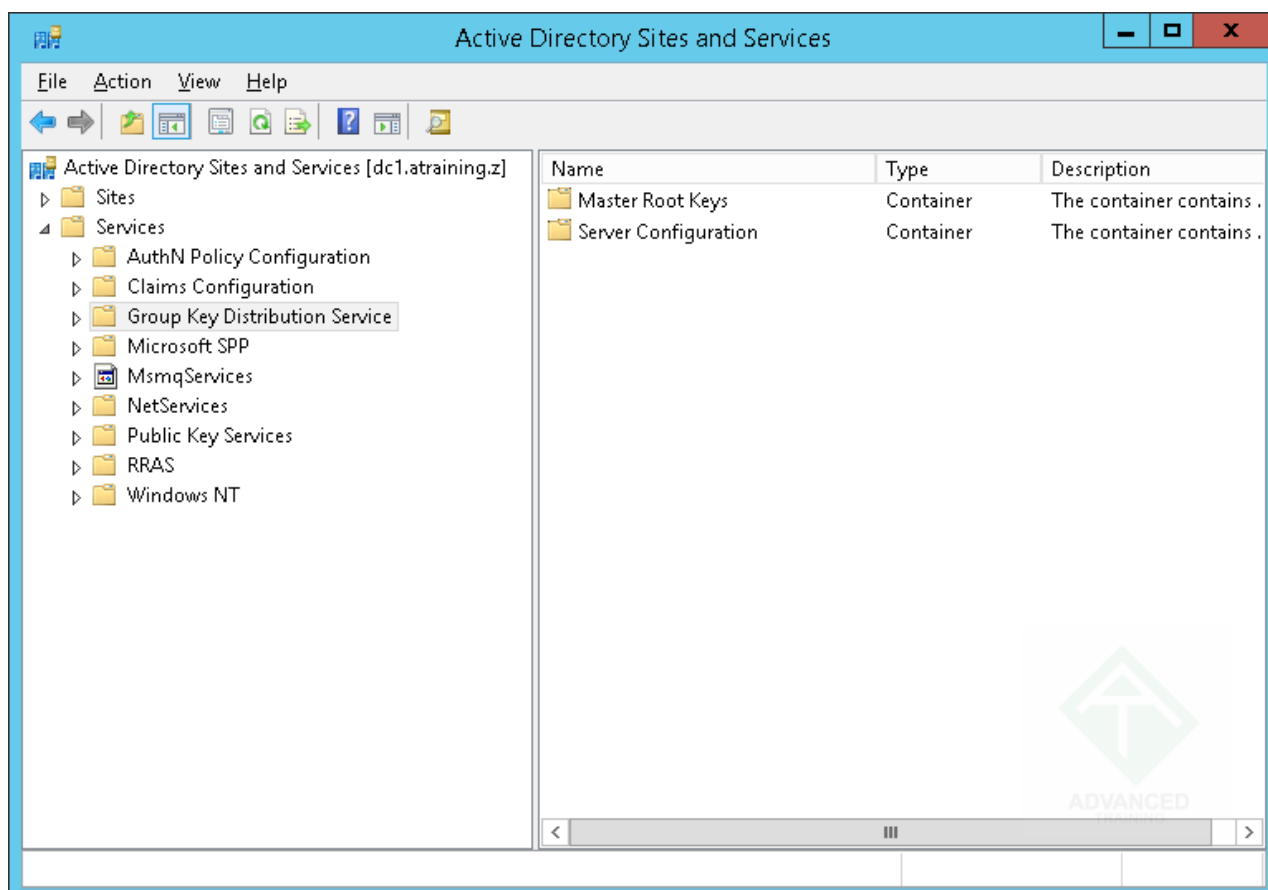


[Проверяем версию Forest Updates в Active Directory.](#)
(кликните для увеличения до 414 px на 462 px)

Если всё ОК, то продолжаем.

Включаем и настраиваем Key Distribution Services

Начиная с Windows Server 2012, в Active Directory появляется новый контейнер для хранения данных, находящийся в лесном разделе Configuration:

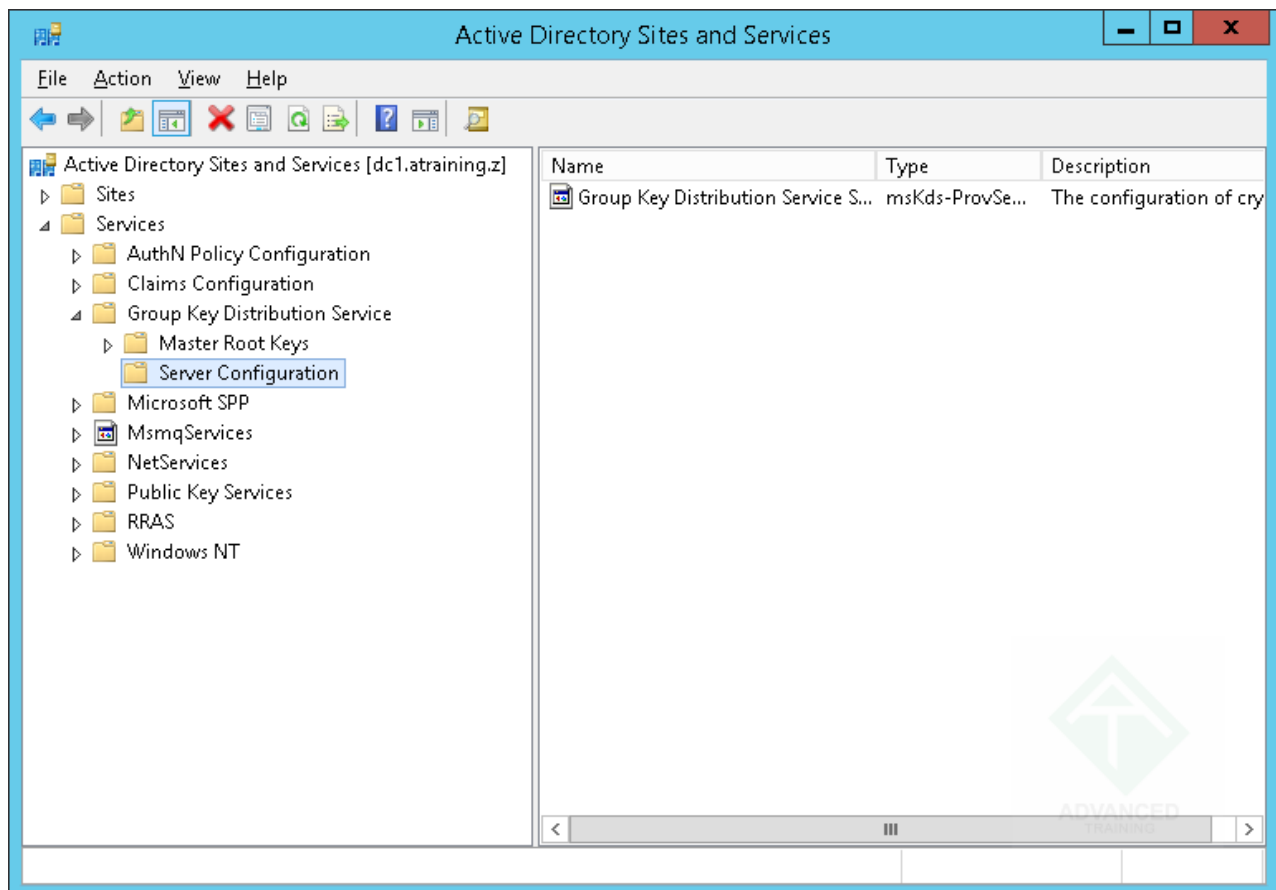


[Контейнер Group Key Distribution Services в Configuration](#)

([кликните для увеличения до 768 px на 537 px](#))

Обращающиеся к нему сервисы работают на всех DC под управлением Windows Server 2012 и старше, технически выглядят как вызываемые из библиотеки `kdssvc.dll` функции по управлению ключевой информацией. Эти функции и будут составлять собой Microsoft Key Distribution Services.

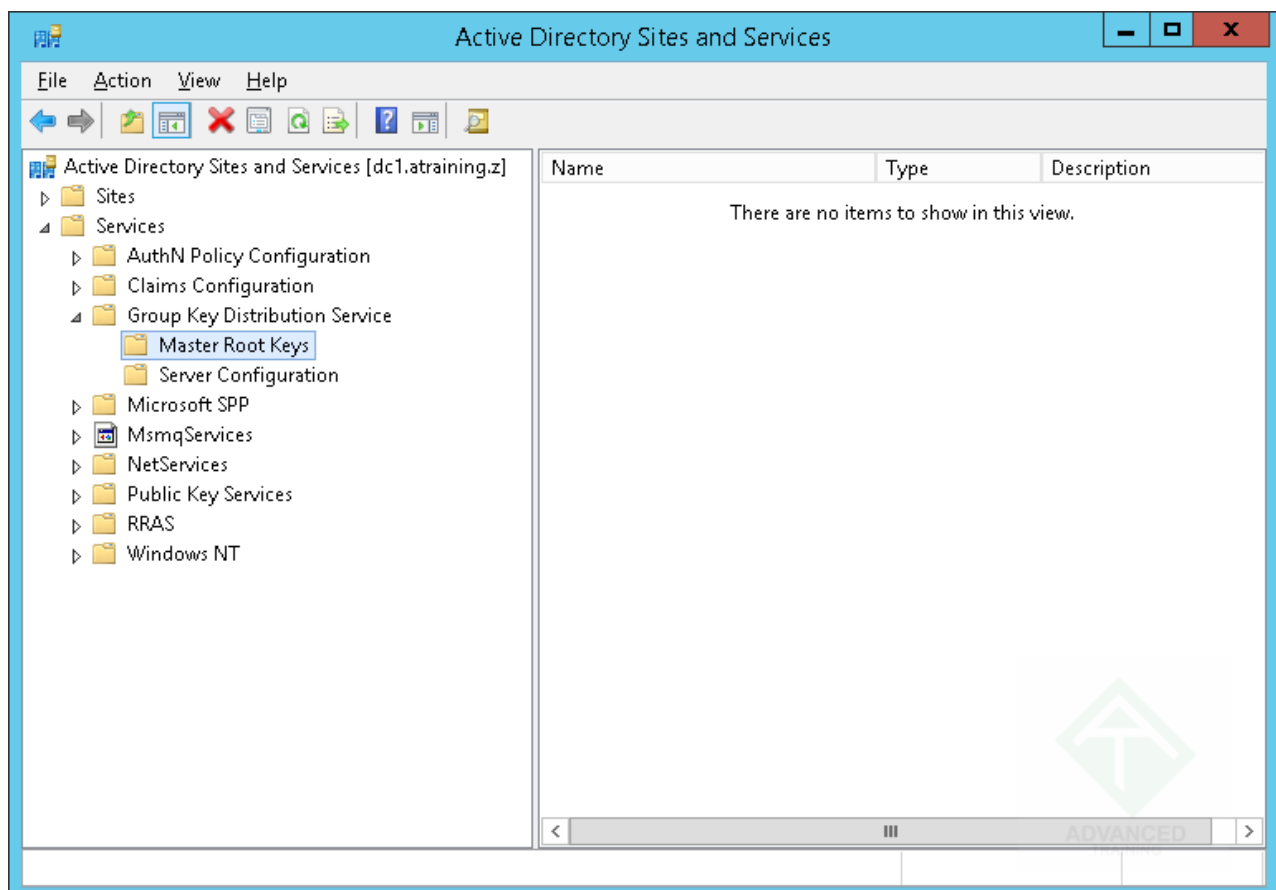
Внутри контейнера `CN=Group Key Distribution Service` есть два других – один для общих настроек Microsoft Key Distribution Services, называется `Server Configuration`:



[Контейнер Server Configuration в Group Key Distribution Services](#)

([кликните для увеличения до 768 px на 537 px](#))

а второй – для самих ключей, он будет называться **Master Root Keys**:



[Контейнер Master Root Keys в Group Key Distribution Services](#)

([кликните для увеличения до 768 px на 537 px](#))

Что в них?

Server Configuration

Здесь будет находиться единственный объект с названием **Group Key Distribution Service Server Configuration**, класса **msKds-ProvServerConfiguration**. Из интересного в нём разве что атрибут **msKds-Version**, равный единице со времён Windows Server 2012 – двойка пока нигде, включая Windows Server 2016 TP5, не обнаружена.

Attribute	Value
msDS-ObjectSoa	<not set>
msDS-SourceAnchor	<not set>
msKds-KDFAlgorithmID	<not set>
msKds-KDFParam	<not set>
msKds-PrivateKeyLength	<not set>
msKds-PublicKeyLength	<not set>
msKds-SecretAgreementAlgorithmID	<not set>
msKds-SecretAgreementParam	<not set>
msKds-Version	1
name	Group Key Distribution Service
objectCategory	CN=ms-Kds-Prov-ServerConfig
objectClass	top; msKds-ProvServerConfigu
objectGUID	882b0008-a7c3-4bde-bc49-85
objectVersion	<not set>

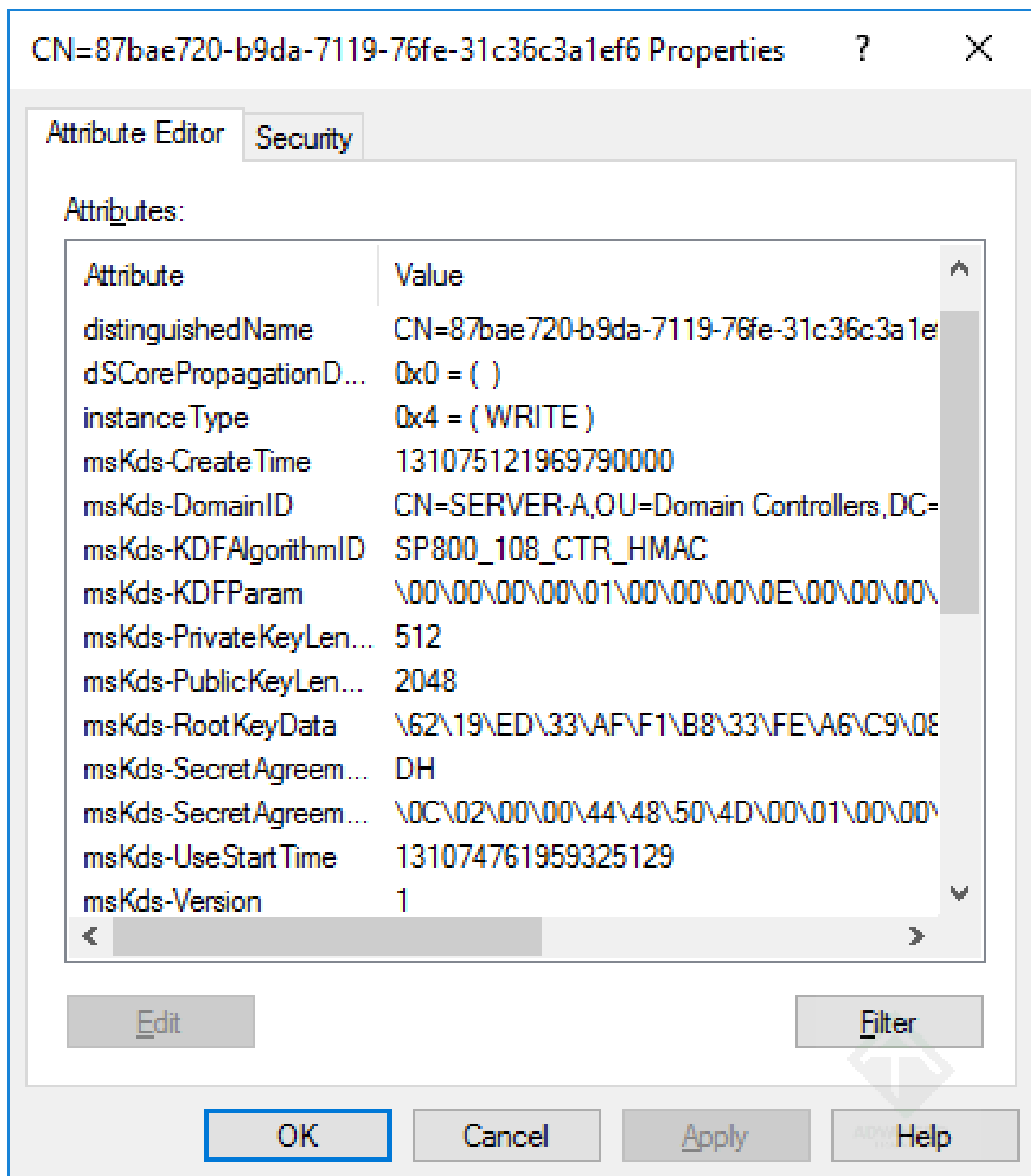
Buttons: Edit, Filter, OK, Cancel, Apply, Help

[Атрибуты Group Key Distribution Service Server Configuration](#)

([кликните для увеличения до 400 px](#) на [455 px](#))

Master Root Keys

Здесь будут находиться сами ключи. Идентифицироваться они будут по CN, равному GUID ключа, класс объекта у них будет **msKds-ProvRootKey**, а интересных атрибутов у них будет много:



[Атрибуты Master Root Key](#)

([кликните для увеличения до 400 px](#) на [455 px](#))

Два параметра `msKds-CreateTime` и `msKds-UseStartTime` будут содержать время создания объекта и время начала возможности его использования. Логика такова, что после создания объекта ему даётся время на то, чтобы среплицироваться на все DC в лесу. Это время стандартно и составляет 10 часов. До наступления `msKds-UseStartTime` данный ключ не будет использоваться, DC просто не будут находить его по запросам со стороны функций, работающих с gMSA.

Криптографические параметры будут задавать длины закрытого (`msKds-PrivateKeyLength`) и открытого (`msKds-PublicKeyLength`) ключа RSA, алгоритм совместной генерации общего секретного ключа (`msKds-SecretAgreementAlgorithmID`, в нашем случае DH) и сами блобы этих криптографических параметров. В атрибуте `msKds-Version` будет версия – такая же, как и в предыдущем объекте, а в `msKds-KDFAlgorithmID` будет название KDF-функции – сейчас там значение `SP800_108_CTR_HMAC`, пока оно стандартное, но API допускает и возможность других значений из подмножества поддерживаемых CNG KDF-функций (например, `SP80056A_CONCAT`, или `PBKDF2`). Посмотреть фактическую поддержку KDF на хосте (контроллере домена Active Directory или просто Windows-машине – неважно), можно командой `show crypto algo`:

```
C:\Users\Administrator\Desktop\atcmd.exe

 4      DESX      Symmetric cipher
 5      DES       Symmetric cipher
 6      RC2       Symmetric cipher
 7      RC4       Symmetric cipher

Crypto algorithms, filtered by: random number generators, total 3.
Num    Name      Type
 0      RNG       Random number generator
 1      FIPS186DSARNG Random number generator
 2      DUALECRNG Random number generator

Crypto algorithms, filtered by: asymmetric ciphers, total 1.
Num    Name      Type
 0      RSA       Asymmetric encryption

Crypto algorithms, filtered by: key agreement, total 5.
Num    Name      Type
 0      DH        Secret agreement
 1      ECDH_P256 Secret agreement
 2      ECDH_P384 Secret agreement
 3      ECDH_P521 Secret agreement
 4      ECDH       Secret agreement

Crypto algorithms, filtered by: key derivation functions, total 6.
Num    Name      Type
 0      SP800_108_CTR_HMAC KDF
 1      SP800_56A_CONCAT  KDF
 2      PBKDF2            KDF
 3      CAPI_KDF          KDF
 4      TLS1_1_KDF        KDF
 5      TLS1_2_KDF        KDF

SERVER-A(config)#
```

[Просмотр списка KDF](#)

[\(кликните для увеличения до 979 px на 599 px\)](#)

KDF, если что – это аббревиатура от Key Derivation Function, т.е. способа генерации ключа нужной длины из не-криптографического материала (например, из пароля).

Эти параметры можно также посмотреть через командлет `Get-KdsConfiguration`:

```
Administrator: Windows PowerShell

PS C:\Users\Administrator> Get-KdsConfiguration

AttributeOfWrongFormat      :
KdfParameters               : {0, 0, 0, 0...}
SecretAgreementParameters   : {12, 2, 0, 0...}
IsValidFormat                : True
SecretAgreementAlgorithm     : DH
KdfAlgorithm                 : SP800_108_CTR_HMAC
SecretAgreementPublicKeyLength : 2048
SecretAgreementPrivateKeyLength : 512
VersionNumber                : 1

PS C:\Users\Administrator> _
```

[Просмотр параметров службы KDS в Windows Server 2016](#)

[\(кликните для увеличения до 859 px на 249 px\)](#)

Он, в общем-то, эти же атрибуты и выводит. Парный к нему командлет, **Set-KdsConfiguration**, позволит изменять параметры – алгоритмы, длины ключей, и всё остальное. Например, можно сменить стандартную KFD-функцию на другую:

```
Set-KdsConfiguration -KdfAlgorithm "PBKDF2"
```

По идее, можно ещё сменить сам алгоритм генерации ключевого материала со старого DH на ECDH, или хотя бы увеличить количество бит у DH с 2048 до 4096, но только вот это сейчас (в Windows Server 2016 TP5) не работает по неизвестной причине – надеюсь, к RTM доделают. Меняется т.е. пока только KDF-функция – список доступных вы можете просмотреть через **ATcmd**, пример есть выше.

Давайте теперь создадим ключ.

Создание нового Root Key

Это несложно – используется командлет **Add-KdsRootKey**, из параметров у которого только разве что время начала действия ключа. Самый простой способ – создать так:

```
Add-KdsRootKey -EffectiveImmediately
```

Тогда ключ создастся и начнётся отсчёт 10 часов – для подстраховки, чтобы среплицировалось во всём лесу Active Directory. Если не хотите ждать – не вопрос. Можно сделать хитро – сгенерить ключ с “минус десятью часами начала работы”:

```
Add-KdsRootKey -EffectiveTime ((get-date).addhours(-10))
```

В этом случае ключ будет работоспособен сразу же. После создания Вы увидите GUID ключа:

```
Administrator: Windows PowerShell

Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> Add-KdsRootKey -EffectiveTime ((get-date).addhours(-10))

Guid
----
87bae720-b9da-7119-76fe-31c36c3a1ef6
```

[Создание KDS-ключа через PowerShell в Windows Server 2016](#)

[\(кликните для увеличения до 859 px на 150 px\)](#)

Можно зайти в контейнер с ключами и посмотреть его параметры – опять же, из интересного там пока что будет разве что сервер, на котором Вы создали этот ключ – в атрибуте `msKds-DomainID`.

Теперь можно и gMSA создать.

Создаём gMSA

Для начала, выберем хост, на котором мы создадим gMSA. Хост должен быть строго на Windows Server 2012 и выше и обладать установленным .NET Framework 4.5 (это будет само собой, если там развёрнуты службы AD DS).

Создание будет несложным – используется тот же командлет, что и раньше:

```
New-ADServiceAccount
```

В качестве обязательных параметров будет имя, которое можно указать сразу после командлета (например, `msa1`):

```
New-ADServiceAccount msa1
```

И надо выбрать – создаём ли “обычный” MSA, привязанный к одному серверу – тогда это выглядит так:

```
New-ADServiceAccount msa2 -RestrictToSingleComputer
```

Или создаём именно gMSA, пригодную для использования в кластерах, и тогда надо указать FQDN хоста, на котором мы заводили KDS-ключ (например, `server-a.atraining.z`):

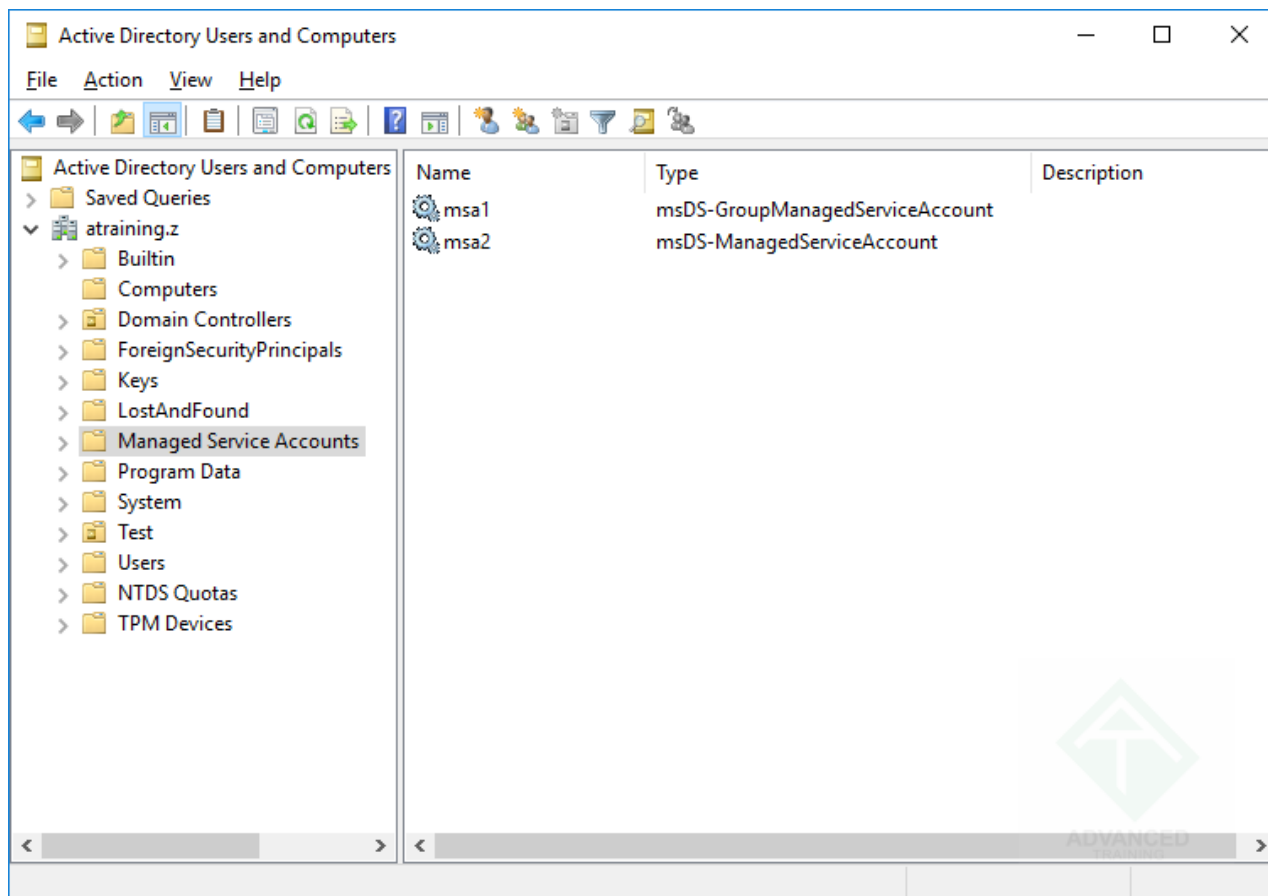
```
New-ADServiceAccount msa1 -DNSHostName server-a.atraining.z
```

А после – указать все те учётки computer’ов, которые смогут получать доступ к security-информации (паролю, например) новорожденной gMSA, задав их перечень в параметре `-PrincipalsAllowedToRetrieveManagedPassword` (например, разрешим учётке `DC1`):

```
New-ADServiceAccount msa1 -DNSHostName server-a.atraining.z -  
PrincipalsAllowedToRetrieveManagedPassword DC1$
```

Можно ещё добавить в конец `-Enabled $true`, и тогда будет полный вариант.

Вот так выглядят созданные учётки (я сделал `msa1` нормальной, gMSA, а `msa2` – ‘устаревшей’ MSA, чтобы было видно, что в зависимости от параметров командлета `New-ADServiceAccount` создаются разные типы объектов Active Directory):



MSA и gMSA

([кликните для увеличения до 754 px на 530 px](#))

Дальнейшее использование простое – надо пойти на hosts, на которых планируется использование этой gMSA, и выполнить на них **Install-ADServiceAccount**. Всё идентично обычному MSA, только можно установить больше чем на 1 host.

Дополнительные возможности

Вы можете задать криптоалгоритмы, используемые для тикетов Kerberos, выдаваемых этой MSA. Это делается, используя параметр - **KerberosEncryptionTypes**. Задавать можно несколько, выбор есть из DES, RC4, AES128 и AES256. Например, так:

```
New-ADServiceAccount msa1 -KerberosEncryptionType AES128|AES256
```

Можно (и нужно) явно задать промежуток смены пароля для gMSA – теперь же нет явного “хоста-монопольного-владельца-MSA”, от настроек которого это было зависимо:

```
New-ADServiceAccount msa1 -ManagedPasswordIntervalInDays 60
```

60, как понятно – это в днях.

Интересный момент – Вы не сможете получить полный доступ к вкладке Attributes у gMSA, если подключаетесь по обычному LDAP. Только если по LDAPS. Ну, это тема [отдельного разговора о безопасности Active Directory](#). Намекну, что сможете через

ATcmd.exe :).

В качестве заключения

Очередная технология, ощутимо поднимающая безопасность и упрощающая управление сервисами стала только лучше. Категорически рекомендуется к использованию – более того, ранее заведённые MSA проще переделать как gMSA – чтобы был единый тип объектов, а не два разных. Конечно, это потребует, чтобы в лесу Active Directory были как минимум NT 6.2-системы, но плюсы очевидны – проще делать gMSA, привязанные к 1й машине, чем специальный устаревший тип объектов с неполным функционалом.

Удач!