OpenLDAP и Ubuntu на практике

Установка и настройка OpenLDAP 2.4 в Ubuntu 14.04

Оглавление

Содержание

Введение

Это руководство является вольным переводом. Исходная конфигурация для **CentOS 6.2** адаптирована под **Ubuntu 14.04 server**. Выражаю благодарность автору оригинала и привожу ссылку:

Another IT blog: «HOWTO : CentOS 6.2 OpenLDAP 2.4 Setup» (EN)

В процессе написания использовались материалы проекта **pro-ldap.ru**. В том числе:

Погружение в OpenLDAP (RU)

Несколько весьма полезных ссылок для понимания сути описанного далее:

OpenLDAP Software 2.4 Administrator's Guide (EN)

Руководство администратора OpenLDAP 2.4 (RU)

LDAP для учёных-ракетчиков (RU)

Конфигурация используемых в нашем стенде машин:

На всех машинах при установке ОС заводится пользовательская учётная запись **user**.

Шлюз 192.168.122.1

Сервер DNS

hostname dns-srv.example.com
ip 192.168.122.140

Сервер каталогов и сервер Kerberos

hostname ldap-srv.example.com
ip 192.168.122.150

Клиентская машина

hostname: ldap-client.example.com
ip 192.168.122.151

Сервер NFS

hostname: nfs-srv.example.com
ip 192.168.122.154

Сервер репликации

hostname: ldap-srv-repl.example.com
ip 192.168.122.160

Начните с установки Ubuntu 14.04. Это довольно просто. Для этого примера я использовал виртуальные машины. Постарайтесь воздержаться от установки большого количества пакетов. Такой подход уменьшит количество обновляемых пакетов, а так же сузит перечень возможных атак на Ваши серверы. После

первоначальной установки каждой виртуальной машины обновите программное обеспечение командой:

```
#  apt-get update && apt-get upgrade
```

Для тех, кто не в курсе... В тексте обращайте внимание на символы **#** и **$**. Идущие после них команды выполняются от имени пользователя **root** и обычного пользователя соответственно. Выбор за Вами, открывать отдельный терминал с правами суперпользователя или предварять команды командой **sudo** в терминале обычного пользователя.

1. Настройка DNS

Для наших целей необходим настроенный DNS-сервер. Если в Вашей сети уже имеется DNS-сервер и он успешно разрешает имена для перечисленных выше машин, то пункт 1.1 можно пропустить. Помните об этом, если в последующих разделах Вы встретите упоминание о сервере DNS (**dns-srv**).

1.1 Настройка DNS-сервера

Где работаем: **dns-srv**

Настроим сеть в файле **/etc/network/interfaces**:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.168.122.140
        netmask 255.255.255.0
        gateway 192.168.122.1
        dns-nameservers 192.168.122.1
```

Настроим имя DNS-сервера. В Ubuntu 14.04 останов или перезапуск сервиса **networking** не поддерживается, поэтому придётся сделать так:

```
#  echo 'dns-srv.example.com' > /etc/hostname
#  reboot
```

Установим DNS сервер. Будем использовать простой **dnsmasq**:

```
#  apt-get install dnsmasq
```

Убедимся, что он работает:

```
$  dig ubuntu.com @localhost | grep ^ubuntu.com
```

Должны получить примерно это:

```
ubuntu.com.    310     IN      A       91.189.94.156
```

Настроим разрешение имён в файле **/etc/hosts**:

```
127.0.0.1            localhost
192.168.122.140      dns-srv.example.com

192.168.122.150      ldap-srv.example.com
192.168.122.151      ldap-client.example.com
```

1.2 Начальная настройка сервера и клиента LDAP

Где работаем: **ldap-srv**

Настроим сеть в файле **/etc/network/interfaces**:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.168.122.150
        netmask 255.255.255.0
        gateway 192.168.122.1
        dns-nameservers 192.168.122.140
```

Настроим имя LDAP-сервера. В Ubuntu 14.04 останов или перезапуск сервиса **networking** не поддерживается, поэтому придётся сделать так:

```
#  echo 'ldap-srv.example.com' > /etc/hostname
#  reboot
```

Проверим разрешение имени будущего сервера LDAP:

```
$  dig ldap-srv.example.com +short
192.168.122.150
```

Проверим обратное разрешение имени:

```
$  dig -x `dig ldap-srv.example.com +short` +short
ldap-srv.example.com.
```

Где работаем: **ldap-client**

Настроим сеть в файле **/etc/network/interfaces**:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.168.122.151
        netmask 255.255.255.0
        gateway 192.168.122.1
        dns-nameservers 192.168.122.140
```

Настроим имя LDAP-клиента. В Ubuntu 14.04 останов или перезапуск сервиса **networking** не поддерживается, поэтому придётся сделать так:

```
#  echo 'ldap-client.example.com' > /etc/hostname
#  reboot
```

Проверим разрешение имени будущего клиента LDAP:

```
$  dig ldap-client.example.com +short
192.168.122.151
```

Проверим обратное разрешение имени:

```
$  dig -x `dig ldap-client.example.com +short` +short
ldap-client.example.com.
```

2. Установка OpenLDAP

2.1 Установка пакетов

Где работаем: **ldap-srv**

Прежде чем установить пакет **sudo-ldap** необходимо задать пароль для учетной записи **root**. Пакеты **sudo** и **sudo-ldap** взаимоисключающие, поэтому нужно перестраховаться, если с их установкой или работой случится беда.

```
#  passwd
Введите новый пароль UNIX:
Повторите ввод нового пароля UNIX:
```

Теперь установим все необходимые пакеты. Мы сразу устанавливаем **krb5-kdc-ldap** и **sudo-ldap**, даже несмотря на то, что настроим их позже. Идея заключается в том, чтобы сразу получить необходимые схемы наборов данных OpenLDAP.

```
#  apt-get install slapd ldap-utils krb5-kdc-ldap sudo-ldap
```

Во время установки Вам будет предложено задать некоторые настройки:

- Пароль учётной записи **admin** для доступа к конфигурации OpenLDAP;
- Название области Kerberos (realm);
- Сервер Kerberos для нашей области;
- Управляющий сервер области.

На данном этапе эти настройки не важны. Мы последовательно опишем их в дальнейшем.

Прежде чем продолжить, убедитесь, что нужные пакеты установились:

```
$  dpkg --get-selections|egrep '(slapd|ldap-utils|krb5-kdc-ldap|sudo-ldap)'
krb5-kdc-ldap                                   install
ldap-utils                                      install
slapd                                           install
sudo-ldap                                       install
```

2.2 Инициализация конфигурации каталога

Где работаем: **ldap-srv**

Инициализацию конфигурации каталога мы произведём с нуля с использованием нового подхода, OLC (**cn=config**).

Создайте каталог для работы со службой каталогов. У нас будет много конфигурационных файлов, неплохо бы класть их в одно место:

```
$  mkdir ~/ldap
$  cd ~/ldap
```

Избавимся от установленных по-умолчанию конфигурации **slapd** и его базы данных:

```
#  service slapd stop
#  rm -rf /etc/ldap/slapd.d
```

```
#  rm -rf /var/lib/ldap
```

Создадим пустой каталог для нашей новой конфигурации:

```
#  mkdir /etc/ldap/slapd.d
```

Создадим файл **2.2-init-config.ldif** с новой конфигурацией и запишем в него:

```
dn: cn=config
objectClass: olcGlobal
cn: config
olcPidFile: /var/run/slapd/slapd.pid
olcArgsFile: /var/run/slapd/slapd.args

dn: olcDatabase={0}config,cn=config
objectClass: olcDatabaseConfig
olcDatabase: {0}config
olcAccess: to *
  by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" manage

dn: cn=schema,cn=config
objectClass: olcSchemaConfig
cn: schema
```

В этом файле первым делом мы определяем корневую запись DIT (Directory Information Tree), **cn=config**. С помощью директив **olcPidFile** и **olcArgsFile** мы указали, куда необходимо записать ID процесса службы каталогов и аргументы его запуска соответственно.

Во втором разделе задаём служебную базу данных конфигурации **cn=config**. Мы так же добавили правило доступа (ACL, Access Control List), разрешающее манипулировать ей от имени пользователя **root** (uid=0, gid=0) с помощью механизма SASL EXTERNAL и идентификационной сущности IPC. Помните, что в конце каждого ACL, если не задан модификатор « **break** », подразумевается наличие правила « **by * none** ». То есть остальной доступ к объекту в условии « **to** » — запрещён.

Последним разделом мы добавляем в конфигурацию контейнер для наборов схем данных.

Инициализируем конфигурацию:

```
#  slapadd -n 0 -F /etc/ldap/slapd.d -l 2.2-init-config.ldif
_#################### 100.00% eta   none elapsed            none fast!
Closing DB...
```

Модификатор « **-n 0** » говорит о том, что мы добавляем данные в базу данных с индексом 0, который зарезервирован для **cn=config**.

Проверим, всё ли в порядке с нашей конфигурацией:

```
#  slaptest -uF /etc/ldap/slapd.d
config file testing succeeded
```

Поправим права доступа, разрешив пользователю **openldap** заправлять в каталоге **/etc/ldap/slapd.d**:

```
#  chown -R openldap:openldap /etc/ldap/slapd.d
#  chmod 750 /etc/ldap/slapd.d
```

2.3 Запуск службы каталогов

Где работаем: **ldap-srv**

Разрешим нашей службе каталогов использовать только IPv4. Для этого установим **SLAPD_OPTIONS="-4"** в файле **/etc/default/slapd**. В остальном конфигурация стандартная:

```
SLAPD_CONF=
SLAPD_USER="openldap"
SLAPD_GROUP="openldap"
SLAPD_PIDFILE=
SLAPD_SERVICES="ldap:/// ldapi:///"
SLAPD_SENTINEL_FILE=/etc/ldap/noslapd
SLAPD_OPTIONS="-4"
```

Настроим **rsyslog**, чтобы он писал события службы каталогов в отдельный файл. Для этого достаточно добавить три строки в его конфигурацию после глобальных директив (**/etc/rsyslog.conf**):

```
$ModLoad imuxsock # provides support for local system logging
$ModLoad imklog   # provides kernel logging support

$KLogPermitNonKernelFacility on

$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

$RepeatedMsgReduction on

$FileOwner syslog
$FileGroup adm
$FileCreateMode 0640
$DirCreateMode 0755
$Umask 0022
$PrivDropToUser syslog
$PrivDropToGroup syslog

$WorkDirectory /var/spool/rsyslog

$IncludeConfig /etc/rsyslog.d/*.conf

# Пишем журнал демона slapd в /var/log/slapd.log
if $programname == 'slapd' then /var/log/slapd.log
& ~
```

Создадим файл для журнала службы каталогов и зададим для него права доступа. Затем перезапустим **rsyslog**, чтобы изменения вступили в силу:

```
#  touch /var/log/slapd.log
#  chmod 0640 /var/log/slapd.log
#  chown syslog:adm /var/log/slapd.log
#  service rsyslog restart
```

Настроим **logrotate** для управления этим журналом. Создадим файл конфигурации **/etc/logrotate.d/slapd** и запишем в него:

```
/var/log/slapd.log {
 daily
 missingok
 rotate 7
 compress
 delaycompress
 notifempty
}
```

Проверим настройку **logrotate**:

```
# logrotate -df /etc/logrotate.conf
```

Наконец запускаем нашу службу каталогов:

```
#  service slapd start
 * Starting OpenLDAP slapd         [ OK ]
```

Заглянем в файл журнала **slapd.log**. Всё ли в порядке?

```
#  tail /var/log/slapd.log
...
slapd starting
```

Проверим, активен ли TCP порт 389:

```
$  ss -tan | grep 389
LISTEN        0      128    *:389         *:*
```

Убедимся, что UNIX сокет тоже активен:

```
$  ss -xa | grep ldap
u_str  LISTEN        0      128    /var/run/slapd/ldapi        44345         * 0
```

Для пущей убедительности проверим текущую конфигурацию каталога с помощью **ldapsearch**:

```
#  ldapsearch -QLLL -Y EXTERNAL -H ldapi:/// -b 'cn=config' dn
dn: cn=config
dn: cn=schema,cn=config
dn: olcDatabase={-1}frontend,cn=config
dn: olcDatabase={0}config,cn=config
```

Заметьте, что **slapadd** (из предыдущего пункта) добавил в наш каталог описание служебной базы данных **frontend** (в ней можно определить опции, которые будут применяться ко всем базам данных в текущей конфигурации OLC). Однако, если те же самые опции (в том числе правила ACL) определены в конкретной базе данных, то будут применяться именно они.

2.4 Подключение динамических модулей

Где работаем: **ldap-srv**

Для работы нам понадобится два модуля. Один — для механизма базы данных **mdb**. На данный момент он рассматривается как основной для нормальных баз данных и должен прийти на смену **bdb** и **hdb**. Второй модуль — **monitor**, для создания и динамической поддержки ветки с информацией о текущем статусе демона **slapd**.

Чтобы их подключить нам понадобится всего один короткий LDIF-файл и специальная запись **cn=module,cn=config**. Назовём файл **2.4-add-modules.ldif** и запишем в него:

```
dn: cn=module,cn=config
objectClass: olcModuleList
cn: module
olcModulePath: /usr/lib/ldap
olcModuleLoad: back_mdb.la
olcModuleLoad: back_monitor.la
```

Каталог с модулями в нашем случае - **/usr/lib/ldap**.

Добавим наш LDIF-файл в конфигурацию:

```
#  ldapadd -QY EXTERNAL -H ldapi:/// -f 2.4-add-modules.ldif
adding new entry "cn=module,cn=config"
```

2.5 Добавление наборов схем данных

Где работаем: **ldap-srv**

Прежде чем добавлять наборы схем данных в каталог, необходимо подготовить наборы схем из пакетов **krb5-kdc-ldap** и **sudo-ldap** к загрузке (перевести их в формат LDIF).

В используемой нами версии Ubuntu искомые наборы схем находятся здесь:

```
/usr/share/doc/sudo-ldap/schema.OpenLDAP
/usr/share/doc/krb5-kdc-ldap/kerberos.schema.gz
```

Скопируем их во временный каталог:

```
$  zcat /usr/share/doc/krb5-kdc-ldap/kerberos.schema.gz > ~/ldap/kerberos.schema
$  cp /usr/share/doc/sudo-ldap/schema.OpenLDAP ~/ldap/sudo.schema
```

На просторах сети был найден несложный скрипт для преобразования файлов **schema** в формат LDIF. Немного откорректируем его (для универсальности) и сохраним в том же каталоге **~/ldap** под именем **2.5-schema-ldif.sh**:

```
#!/bin/bash

SCHEMAD=~/ldap

tmpd=`mktemp -d`
pushd ${tmpd} >>/dev/null

echo '' > convert.dat

for schema in ${SCHEMAS} ; do
    echo include ${SCHEMAD}/${schema} >> convert.dat
done

slaptest -f convert.dat -F .

if [ $? -ne 0 ] ; then
    echo "slaptest conversion failed"
    exit
fi

for schema in ${SCHEMAS} ; do
    fullpath=${SCHEMAD}/${schema}
    schema_name=`basename ${fullpath} .schema`
    schema_dir=`dirname ${fullpath}`
    ldif_file=${schema_name}.ldif

    find . -name *${schema_name}.ldif -exec mv '{}' ./${ldif_file} \;
    sed -i "/dn:/ c dn: cn=${schema_name},cn=schema,cn=config" ${ldif_file}
    sed -i "/cn:/ c cn: ${schema_name}" ${ldif_file}
    sed -i '/structuralObjectClass/ d' ${ldif_file}
    sed -i '/entryUUID/ d' ${ldif_file}
    sed -i '/creatorsName/ d' ${ldif_file}
    sed -i '/createTimestamp/ d' ${ldif_file}
    sed -i '/entryCSN/ d' ${ldif_file}
    sed -i '/modifiersName/ d' ${ldif_file}
    sed -i '/modifyTimestamp/ d' ${ldif_file}
    sed -i '/^ *$/d' ${ldif_file}

    mv ${ldif_file} ${schema_dir}
done

popd >>/dev/null
rm -rf $tmpd
```

Сделаем его исполняемым и запустим, передав через переменную **SCHEMAS** имена файлов конвертируемых наборов схем:

```
$  chmod +x 2.5-schema-ldif.sh
$  SCHEMAS='kerberos.schema sudo.schema' 2.5-schema-ldif.sh
config file testing succeeded
```

На выходе должны получить два набора схем данных в формате LDIF:

```
$  ls ~/ldap | egrep '(kerberos.ldif|sudo.ldif)'
kerberos.ldif
sudo.ldif
```

Переместим их в каталог к остальным наборам схем и поправим права доступа:

```
#  mv ~/ldap/{kerberos.ldif,sudo.ldif} /etc/ldap/schema
#  chown root:root /etc/ldap/schema/{kerberos.ldif,sudo.ldif}
```

```
#  chmod 0644 /etc/ldap/schema/{kerberos.ldif,sudo.ldif}
```

Удалим не нужные больше наборы схем данных:

```
$  rm kerberos.schema sudo.schema
```

Создадим LDIF-файл с необходимыми нам наборами схем данных. Порядок их следования в файле очень важен! Атрибуты наборов схем иерархически связаны и требуют их объявления с соблюдением иерархии. Назовём файл **`2.5-add-schemas.ldif`** и запишем в него:

```
include: file:///etc/ldap/schema/core.ldif
include: file:///etc/ldap/schema/cosine.ldif
include: file:///etc/ldap/schema/inetorgperson.ldif
include: file:///etc/ldap/schema/collective.ldif
include: file:///etc/ldap/schema/corba.ldif
include: file:///etc/ldap/schema/duaconf.ldif
include: file:///etc/ldap/schema/openldap.ldif
include: file:///etc/ldap/schema/dyngroup.ldif
include: file:///etc/ldap/schema/java.ldif
include: file:///etc/ldap/schema/misc.ldif
include: file:///etc/ldap/schema/nis.ldif
include: file:///etc/ldap/schema/ppolicy.ldif
include: file:///etc/ldap/schema/kerberos.ldif
include: file:///etc/ldap/schema/sudo.ldif
```

Последними строчками мы указали получившиеся в результате конвертации наборы схем **`kerberos.ldif`** и **`sudo.ldif.`**

Добавим наши наборы схем:

```
#  ldapadd -QY EXTERNAL -H ldapi:/// -f 2.5-add-schemas.ldif
adding new entry "cn=core,cn=schema,cn=config"
adding new entry "cn=cosine,cn=schema,cn=config"
adding new entry "cn=inetorgperson,cn=schema,cn=config"
adding new entry "cn=collective,cn=schema,cn=config"
adding new entry "cn=corba,cn=schema,cn=config"
adding new entry "cn=duaconf,cn=schema,cn=config"
adding new entry "cn=openldap,cn=schema,cn=config"
adding new entry "cn=dyngroup,cn=schema,cn=config"
adding new entry "cn=java,cn=schema,cn=config"
adding new entry "cn=misc,cn=schema,cn=config"
adding new entry "cn=nis,cn=schema,cn=config"
adding new entry "cn=ppolicy,cn=schema,cn=config"
adding new entry "cn=kerberos,cn=schema,cn=config"
adding new entry "cn=sudo,cn=schema,cn=config"
adding new entry "cn=autofs,cn=schema,cn=config"
```

2.6 Инициализация базы данных

Где работаем: **ldap-srv**

Вот мы и подошли к созданию базы данных, в которой будем хранить нашу рабочую информацию.

Для начала создадим пароль администратора. Утилита **`slappasswd`** генерирует посоленный хэш вводимого нами пароля:

```
# slappasswd -h '{SSHA}'
New password:
Re-enter new password:
{SSHA}RMjrfsi7NkpBMR+NQHTDk4iddvo/2le+
```

Не забудьте сам пароль! :)

Сформируем конфигурационный LDIF-файл для нашей базы данных (**`2.6-db.ldif`**) и запишем получившийся хэш в атрибут **`olcRootPW`**:

```
dn: olcDatabase=mdb,cn=config
objectClass: olcMdbConfig
olcDatabase: mdb
olcSuffix: dc=example,dc=com
olcDbDirectory: /var/lib/ldap
olcDbMaxsize: 1073741824
olcRootDN: cn=admin,dc=example,dc=com
olcRootPW: {SSHA}RMjrfsi7NkpBMR+NQHTDk4iddvo/2le+
olcAccess: {0}to *
  by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" manage
  by * break
olcAccess: {1}to attrs=userPassword
  by self write
  by anonymous auth
olcAccess: {2}to *
  by self read

dn: olcDatabase=monitor,cn=config
objectClass: olcDatabaseConfig
olcDatabase: monitor
```

И вновь несколько комментариев...

- В качестве механизма манипуляции данными выбираем ранее подключенный модуль **mdb**.
- В качестве суффикса (корневой записи) создаваемого DIT (Directory Information Tree) обычно используется имя домена DNS. Но это не является обязательным. Для примера мы задали суффикс **dc=example,dc=com**.
- В атрибуте **olcDbDirectory** мы указали путь к каталогу бузы данных. Атрибут обязательный и требует существование каталога на момент загрузки в базу данных.
- Механизм **mdb** требует указания максимального размера базы данных. Он задается в байтах и должен быть больше её ожидаемого размера, даже с учётом прироста. В файловой системе должно быть достаточно свободного места для размещения базы данных такого размера.
- В атрибут **olcRootDN** мы записываем DN администратора нашей базы данных.
- Для пользователя, указанного в атрибуте **olcRootDN**, задавать правило в ACL не нужно, он обладает полным доступом к данным в вашей базе. Поэтому постарайтесь сохранить его пароль в надежном месте (например, используя keepass или gpg).
- Добавляем три правила доступа для базы данных **mdb**:
  - Доступ ко всей базе данных:
    - Разрешить доступ пользователю **root** с использованием механизма SASL EXTERNAL.
    - Продолжить анализ ACL, если нет совпадений с субъектами доступа, указанными с помощью директивы « **by** ».
  - Доступ к атрибуту **userPassword**:
    - Разрешить доступ для смены пароля самим пользователем.
    - Разрешить доступ для аутентификации.
  - Доступ к остальной базе данных:
    - Разрешить пользователям просматривать свои записи.
      (важно для аутентификации через **nslcd**, раздел 5)
- С помощью механизма **monitor** включаем мониторинг базы данных (добавляем базу данных **monitor**).

Для нашей базы данных необходимо создать каталог и задать ему права доступа:

```
#  mkdir /var/lib/ldap
#  chown openldap:openldap /var/lib/ldap
#  chmod 0700 /var/lib/ldap
```

Загрузим конфигурацию базы данных:

```
#  ldapadd -QY EXTERNAL -H ldapi:/// -f 2.6-db.ldif
adding new entry "olcDatabase=mdb,cn=config"
adding new entry "olcDatabase=monitor,cn=config"
```

Определим RootDN для доступа к конфигурации службы каталогов. Он будет ссылаться на RootDN, находящийся в нашей БД. Эта запись желательна только при первоначальной настройке или в тестовой конфигурации. Не оставляйте её в боевой системе! Для **{-1}frontend** зададим минимально необходимые права для доступа к RootDSE. Создадим LDIF-файл **2.6-acl-mod.ldif**, модифицирующий права доступа:

```
dn: olcDatabase={0}config,cn=config
changetype: modify
add: olcRootDN
olcRootDN: cn=admin,dc=example,dc=com

dn: olcDatabase={-1}frontend,cn=config
changetype: modify
add: olcAccess
olcAccess: {0}to *
  by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth manage
  by * break
olcAccess: {1}to dn.base=""
  by * read
olcAccess: {2}to dn.base="cn=subschema"
  by * read
```

Загрузим LDIF, изменяющий ACL:

```
#  ldapadd -QY EXTERNAL -H ldapi:/// -f 2.6-acl-mod.ldif
modifying entry "olcDatabase={-1}frontend,cn=config"
modifying entry "olcDatabase={0}config,cn=config"
```

Теперь мы ещё больше повысили значимость учётной записи администратора. С её помощью теперь можно получить полный доступ к службе каталогов. Имейте это ввиду. :)

Проверим корректность всех ACL и, заодно, наличие всех добавленных нами данных (вывод отформатирован для наглядности):

```
#  ldapsearch -QLLL -Y EXTERNAL -H ldapi:/// -b 'cn=config' dn olcAccess
dn: cn=config

dn: cn=module{0},cn=config

dn: cn=schema,cn=config
dn: cn={0}core,cn=schema,cn=config
dn: cn={1}cosine,cn=schema,cn=config
dn: cn={2}inetorgperson,cn=schema,cn=config
dn: cn={3}collective,cn=schema,cn=config
dn: cn={4}corba,cn=schema,cn=config
dn: cn={5}duaconf,cn=schema,cn=config
dn: cn={6}openldap,cn=schema,cn=config
dn: cn={7}dyngroup,cn=schema,cn=config
dn: cn={8}java,cn=schema,cn=config
dn: cn={9}misc,cn=schema,cn=config
dn: cn={10}nis,cn=schema,cn=config
dn: cn={11}ppolicy,cn=schema,cn=config
dn: cn={12}kerberos,cn=schema,cn=config
dn: cn={13}sudo,cn=schema,cn=config

dn: olcDatabase={-1}frontend,cn=config
olcAccess: {0}to *
  by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external ,cn=auth manage
  by * break
olcAccess: {1}to dn.base=""
  by * read
olcAccess: {2}to dn.base="cn=subschema"
  by * read

dn: olcDatabase={0}config,cn=config
olcAccess: {0}to *
  by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external ,cn=auth" manage
```

```
dn: olcDatabase={1}mdb,cn=config
olcAccess: {0}to * by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external ,cn=auth" manage
  by * break
olcAccess: {1}to attrs=userPassword
  by self write
  by anonymous auth
olcAccess: {2}to *
  by self read

dn: olcDatabase={2}monitor,cn=config
```

Убедимся, что учётная запись администратора имеет доступ к нашей службе каталогов:

```
$  ldapwhoami -WD cn=admin,dc=example,dc=com
Enter LDAP Password:
dn:cn=admin,dc=example,dc=com
```

Отлично! Теперь у нас есть работающий сервер OpenLDAP.

Отредактируем файл **/etc/ldap/ldap.conf**. Это нужно только для того, чтобы немного упростить себе жизнь и меньше печатать в дальнейшем. В **BASE** подставьте свой суффикс, а в **URI** — FQDN Вашего сервера OpenLDAP:

```
BASE  dc=example,dc=com
URI  ldap://ldap-srv.example.com
# TLS_CACERT /etc/ssl/certs/rootca.crt
# TLS_REQCERT demand
TIMELIMIT 15
TIMEOUT  20
```

С настройкой TLS мы разберемся в следующем разделе.

И последний штрих. Добавим демон нашей службы каталогов в автозагрузку:

```
#  update-rc.d slapd defaults
```

3. Настройка TLS (Transport Layer Security)

В наши дни использование TLS является практически обязательным. TLS реализует защитные функции конфиденциальности и целостности данных, а так же служит для поддержки аутентификации LDAP с использованием механизма SASL EXTERNAL. TLS определён в RFC4346.

Для настройки TLS на сервере нам понадобятся SSL сертификат и ключ. Сертификат должен быть подписан доверенным удостоверяющим центром (УЦ, Certificate Authority, CA) или собственным удостоверяющим центром (на основе «самоподписанного», self-signed, сертификата), если используется в тестовых целях.

С помощью OpenSSL мы создадим импровизированный удостоверяющий центр и затем настроим TLS.

Прямая цитата из Википедии:

> Сертификаты, как правило, используются для обмена зашифрованными данными в больших сетях. Криптосистема с открытым ключом решает проблему обмена секретными ключами между участниками безопасного обмена, однако не решает проблему доверия к открытым ключам. Предположим, что Алиса, желая получать зашифрованные сообщения, генерирует пару ключей, один из которых (открытый) она публикует каким-либо образом. Любой, кто желает отправить ей конфиденциальное сообщение, имеет возможность зашифровать его этим ключом, и быть уверенным, что только она (так как только она обладает соответствующим секретным ключом) сможет это сообщение прочесть. Однако описанная схема ничем не может помешать злоумышленнику Давиду создать пару ключей, и опубликовать свой открытый ключ, выдав его за ключ Алисы. В таком случае Давид сможет расшифровывать и читать, по крайней мере, ту часть сообщений, предназначенных Алисе, которые были по ошибке зашифрованы его открытым ключом.
>
> Идея сертификата — это наличие третьей стороны, которой доверяют две другие стороны информационного обмена. Предполагается, что таких третьих сторон немного, и их открытые ключи всем известны каким-либо

способом, например, хранятся в операционной системе или публикуются в журналах. Таким образом, подлог открытого ключа третьей стороны легко выявляется.

Если Алиса сформирует сертификат со своим публичным ключом, и этот сертификат будет подписан третьей стороной (например, Трентом), любой, доверяющий Тренту, сможет удостовериться в подлинности открытого ключа Алисы. В централизованной инфраструктуре в роли Трента выступает удостоверяющий центр. В сетях доверия Трент может быть любым пользователем, и следует ли доверять этому пользователю, удостоверившему ключ Алисы, решает сам отправитель сообщения.

3.1 Удостоверяющий центр на основе самоподписанного сертификата с помощью OpenSSL

Где работаем: **ldap-srv**

Итак, мы создаём централизованную инфраструктуру открытых ключей (PKI) в миниатюре. Для этого сформируем пару закрытый ключ / корневой сертификат удостоверяющего центра. Причём сертификат будет подписан этим же закрытым ключом (т. е. станет «самоподписанным»).

Затем сгенерируем новый закрытый ключ для нашей службы каталогов и создадим для неё сертификат, подписанный закрытым ключом нашего удостоверяющего центра. При этом клиенские рабочие станции должны знать только корневой сертификат. Используя его они могут установить безопасное соединение с любым сервером, который имеет закрытый ключ и соответствующий ему сертификат, подписанный нашим УЦ.

Такие пары (ключ/сертификат) можно создавать для множества задач. Например, в разделе, посвященном репликации, мы создадим второй сервер каталогов, которому тоже понадобится такая пара.

Удостоверяющий центр желательно разворачивать на отдельной машине, не имеющей подключения к сети, но для простоты примера мы проделаем всю работу на **ldap-srv.example.com**.

Создадим каталог для нашего удостоверяющего центра (CA) и установим безопасные права доступа. Затем зададим значение **umask** таким, чтобы вновь создаваемые файлы имели права доступа чтения и записи только для создавшего их пользователя:

```
#  mkdir /root/CA
#  chmod u=rwx,g=,o= /root/CA
#  cd /root/CA
#  umask 066
```

В конфигурационном файле **/etc/ssl/openssl.cnf** в секции « **[CA_default]** » для удобства поменяем значение директивы « **dir = ./demoCA** » на « **dir = ./** ». В этом же разделе директивой « **default_days** » задаётся срок действия выпускаемых сертификатов. Можете поменять её значение по своему усмотрению.

Файлы **index.txt** и **serial** будут играть роль своеобразной базы данных, чтобы отслеживать статус выпущенных закрытых ключей и сертификатов.

Создадим структуру каталогов и файлов для своего удостоверяющего центра:

```
#  mkdir certs crl newcerts private
#  chmod 700 private
#  touch index.txt
#  echo 1000 > serial
```

Сгенерируем закрытый ключ удостоверяющего центра (длиной 4096 бит). Он должен хранится особенно бережно. Поэтому мы защитим его при помощи шифра AES. Вам будет предложено ввести пароль для доступа к новому закрытому ключу. Запомните его и не потеряйте. Это последний рубеж защиты от злоумышленника. Без пароля выпустить новые сертификаты не получится.

```
#  openssl genrsa -aes256 -out private/rootca.key 4096
```

Изменим права доступа к новому ключу, чтобы ненароком его не стереть:

```
#  chmod 400 private/rootca.key
```

Откройте конфигурационный файл OpenSSL (**openssl.cnf**) и найдите секции **[usr_cert]** и **[v3_ca]**. Убедитесь, что в них присутствуют следующие директивы:

```
[ usr_cert ]
# Эти расширения будут добавлены при подписывании запроса нашим УЦ.
basicConstraints=CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
nsComment = "OpenSSL Generated Certificate"
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer

[ v3_ca ]
# Расширения для типового УЦ
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer
basicConstraints = CA:true
keyUsage = cRLSign, keyCertSign
```

Сейчас мы можем выпустить корневой сертификат удостоверяющего центра, подписав его закрытым ключом **rootca.key**. Так как это сертификат УЦ, используем расширение « **v3_ca** »:

```
#  openssl req -sha256 -new -x509 -days 3650 -extensions v3_ca \
       -key private/rootca.key -out certs/rootca.crt \
       -subj /C=RU/ST=Moscow/L=Moscow/O=ExampleInc/OU=ITdept/CN=ca-server/emailAddress=support@example.com
#  chmod 444 certs/rootca.crt
```

В качестве алгоритма хэширования мы используем SHA-2 (подвид SHA-256). Стоимость атаки на SHA-1 стремительно падает, а о практическом применении новоявленного SHA-3 говорить пока рано. Почему мы выбрали именно SHA-256, а не SHA-512? С сертификатом, использующим SHA-512, Вы сможете запустить демон **slapd**, но не сможете к нему подключиться. Увидите лишь такую многозначную ошибку:

```
can't connect: A TLS packet with unexpected length was received...
```

Мы так же указываем количество дней, в течении которых сертификат будет действителен. Десять лет — достаточно большой срок.

С помощью модификатора « **subj** » заносим в сертификат информацию:

- C - Country Name (страна) - RU
- ST - State or Province Name (штат или провинция) - Moscow
- L - Locality Name (город) - Moscow
- O - Organization Name (наименование организации) - ExampleInc
- OU - Organizational Unit Name (наименование подразделения) - ITdept
- CN - Common Name (имя субъекта или FQDN сервера) - ca-server
- emailAddress - Email Address (адрес электронной почты) - support@example.com

Можем посмотреть содержимое сертификата следующей командой:

```
#  openssl x509 -in certs/rootca.crt -noout -text
```

Удостоверяющий центр готов к выпуску сертификатов.

3.2 Выпуск сертификата для сервера службы каталогов

Где работаем: **ldap-srv**

Как правило, закрытые ключи клиентов УЦ не должны хранится в самом УЦ. Клиент может сам сформировать ключевую пару и прислать в УЦ лишь запрос на подпись (Certificate Signing Request, CSR). Запрос содержит открытый ключ. Однако мы будем создавать все ключи и хранить их в одном месте. В организации, которая серьёзно подходит к проблемам безопасности такой процесс работы может быть неприемлемым.

Продолжаем в том же каталоге **/root/CA**. Создадим закрытый ключ для сервера службы каталогов:

```
#  openssl genrsa -out private/ldap-srv.example.com.key 4096
#  chmod 400 private/ldap-srv.example.com.key
```

Сгенерируем запрос на подпись сертификата. Наименование организации (« **ExampleInc** ») должно совпадать с наименованием в корневом сертификате УЦ. В качестве « **Common Name** » укажем FQDN нашего сервера:

```
#  openssl req -sha256 -new \
       -key private/ldap-srv.example.com.key -out certs/ldap-srv.example.com.csr \
       -subj /C=RU/ST=Moscow/L=Moscow/O=ExampleInc/OU=ITdept/CN=ldap-
srv.example.com/emailAddress=support@example.com
```

Следующим шагом должно быть подписание запроса CSR существующим доверенным удостоверяющим центром (например, VeriSign) в обмен на деньги. Но нам не хочется платить за эту услугу, или у нас нет своего (корпоративного) CA, или это просто тестовая конфигурация, а может нам просто всё равно. Поэтому мы подпишем его с помощью своего собственного CA:

```
#  openssl ca -extensions usr_cert -notext -md sha256 \
       -keyfile private/rootca.key -cert certs/rootca.crt \
       -in certs/ldap-srv.example.com.csr -out certs/ldap-srv.example.com.crt
#  chmod 444 certs/ldap-srv.example.com.crt
```

Создадим каталог для ключевой информации нашего сервера и поместим туда получившиеся у нас файлы:

```
#  mkdir /etc/ldap/ssl
#  chown openldap:openldap /etc/ldap/ssl
#  chmod 0500 /etc/ldap/ssl
#  cp certs/ldap-srv.example.com.crt private/ldap-srv.example.com.key /etc/ldap/ssl
```

Установим права доступа для ключевой информации:

```
#  chown openldap:openldap /etc/ldap/ssl/{ldap-srv.example.com.crt,ldap-srv.example.com.key}
#  chmod 0400 /etc/ldap/ssl/{ldap-srv.example.com.crt,ldap-srv.example.com.key}
```

Поместим корневой сертификат в каталог с сертификатами операционной системы и зададим для него права доступа:

```
#  cp certs/rootca.crt /etc/ssl/certs/
#  chmod 0644 /etc/ssl/certs/rootca.crt
```

В заключение можем удалить запрос CSR, он нам больше не нужен:

```
#  rm -rf certs/ldap-srv.example.com.csr
```

Каталог **/root/CA** теперь выполняет роль удостоверяющего центра. Аналогичным образом его можно создать на другой машине, тогда надо будет копировать секретные ключи и подписанные сертификаты по сети с помощью **scp** или через съёмный носитель. Неплохой идеей будет хранить этот каталог на отдельном носителе и монтировать его только для выпуска новых сертификатов. Сам носитель можно, например, положить в сейф. Процесс создания новых пар ключ-сертификат в будущем будет состоять из трёх этапов:

1. Генерация секретного ключа и запроса на подписание сертификата;
2. Подписание сертификата с помощью закрытого ключа нашего CA (**rootca.key**);
3. Перемещение новых секретного ключа и подписанного сертификата в каталоги, где они будут использоваться.

3.3 Настройка конфигурации TLS

Где работаем: **ldap-srv**

Вернёмся во временный каталог:

```
$ cd ~/ldap
```

Создадим LDIF-файл **3.2-tls-config.ldif** для внесения в каталог конфигурации TLS и запишем в него:

```
dn: cn=config
add: olcTLSVerifyClient
olcTLSVerifyClient: never
-
add: olcTLSCertificateFile
olcTLSCertificateFile: /etc/ldap/ssl/ldap-srv.example.com.crt
-
add: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/ldap/ssl/ldap-srv.example.com.key
-
add: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/ssl/certs/rootca.crt
```

Этими записи говорят демону **slapd**, где лежит его сертификат и ключ, где лежит корневой сертификат УЦ и что от клиентов требовать наличие сертификата не нужно. Чтобы окончательно всё запутать, мы могли бы создать сертификаты для всех клиентов. В реальной жизни такая аутентификация клиента принесёт небольшое усиление защиты за счёт большого увеличения работы по сопровождению всех этих сертификатов. Тем более далее в этом руководстве мы опишем механизмы аутентификации Kerberos.

Загрузим конфигурацию TLS в наш каталог:

```
#  ldapmodify -QY EXTERNAL -H ldapi:/// -f 3.2-tls-config.ldif
modifying entry "cn=config"
```

Теперь наш сервер OpenLDAP должен поддерживать расширения TLS. Перепроверим, что всё в порядке:

```
#  ldapsearch -QLLLY EXTERNAL -H ldapi:/// -b cn=config -s base | grep -i tls
olcTLSCertificateFile: /etc/ldap/ssl/ldap-srv.example.com.crt
olcTLSCertificateKeyFile: /etc/ldap/ssl/ldap-srv.example.com.key
olcTLSCACertificateFile: /etc/ssl/certs/rootca.crt
olcTLSVerifyClient: never
```

Вновь отредактируем конфигурационный файл **/etc/ldap/ldap.conf** и включим поддержку TLS:

```
BASE            dc=example,dc=com
URI             ldap://ldap-srv.example.com
TLS_CACERT      /etc/ssl/certs/rootca.crt
TLS_REQCERT     demand
TIMELIMIT       15
TIMEOUT         20
```

Обычно для конфигурации **cn=config** перезагрузка службы не требуется, но чтобы созданная нами ключевая информация «подхватилась», придётся это сделать:

```
#  service slapd restart
```

```
 * Stopping OpenLDAP slapd         [ OK ]
 * Starting OpenLDAP slapd         [ OK ]
```

Проверьте соединение с сервером с использованием TLS (модификатор « **-ZZ** »). На этот раз мы выполним запрос с использованием DN нашего администратора:

```
$  ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b cn=config -s base
Enter LDAP Password:
dn: cn=config
objectClass: olcGlobal
cn: config
olcArgsFile: /var/run/slapd/slapd.args
olcPidFile: /var/run/slapd/slapd.pid
olcTLSCACertificateFile: /etc/ssl/certs/rootca.crt
olcTLSCertificateFile: /etc/ldap/ssl/ldap-srv.example.com.crt
olcTLSCertificateKeyFile: /etc/ldap/ssl/ldap-srv.example.com.key
olcTLSVerifyClient: never
```

Обратите внимание, что в запросе **ldapsearch** нам не пришлось указывать имя хоста (« **-H ldap://ldap-srv.example.com** »). Всё потому что оно указано в файле **/etc/ldap/ldap.conf** . Что касается TLS, то во время инициирования соединения клиент (на данном этапе клиентский запрос выполняет сам сервер) получает от сервера его подписанный сертификат (**ldap-srv.example.com.crt**). Клиент может ничего не знать о сервере, но у него есть сертификат CA (**rootca.crt**), с помощью которого и проверяется сервер.

3.4 Создание CRL и отзыв сертификатов

Где работаем: **ldap-srv**

Сертификаты не вечны. Во-первых при создании задаётся его срок действия. При этом частота обновления сертификатов — баланс между безопасностью и удобством. Во-вторых он может быть скомпрометирован, если скомпрометирован его закрытый ключ. Как во втором случае оповестить субъектов доступа, что сертификат более не действителен? Для этого и служит Certificate Revocation List (CRL). Он представляет собой список серийных номеров отозванных сертификатов, подписанный УЦ, который их выпустил.

Вернёмся в каталог удостоверяющего центра:

```
#  cd /root/CA
```

Прежде чем мы сможем сгенерировать CRL, надо создать файл **crlnumber**. Он нужен **openssl**, чтобы отслеживать номер следующего CRL:

```
#  echo 1000 > crlnumber
```

В стандартной конфигурации **openssl** использует CRL V1. Раскомментируйте строку « **crl_extensions = crl_ext** » в **/etc/ssl/openssl.cnf**, чтобы переключиться на CRL V2. Это хорошая идея, за исключением тех случаев, когда надо использовать именно CRL V1 (например, при использовании сильно устаревшего браузера). Создаём CRL:

```
#  openssl ca -keyfile private/rootca.key -cert certs/rootca.crt  -gencrl -out crl/rootca.crl
```

Посмотреть результат можно так:

```
#  openssl crl -in crl/rootca.crl -text
```

Предположим, что закрытый ключ сервера **ldap-srv** был скомпрометирован. Чтобы оповестить об этом клиентские машины, надо отозвать сертификат этого сервера, создать CRL, а затем — распространить CRL среди клиентов.

Отзываем сертификат:

```
#  openssl ca -keyfile private/rootca.key -cert certs/rootca.crt -revoke certs/ldap-srv.example.com.crt
```

Заглянем в **index.txt**. Начало записи с нашим сертификатом теперь изменилось с « **V** » на « **R** »:

```
# cat index.txt
R 160116072355Z 150119081313Z 1000 unknown /C=RU/ST=Moscow/O=ExampleInc/OU=ITdept/CN=ldap-
srv.example.com/emailAddress=support@example.com
```

Обратите внимание, что копии новых сертификатов так же содержатся в каталоге **newcerts**. При этом имя файла содержит серийный номер сертификата. Когда отзываете сертификаты, вместо файлов в каталоге **certs** Вы можете пользоваться каталогом **newcerts**. Результат будет идентичен. Например:

```
#  openssl ca -keyfile private/rootca.key -cert certs/rootca.crt -revoke newcerts/1000.pem
```

Обновим CRL:

```
#  openssl ca -keyfile private/rootca.key -cert certs/rootca.crt -gencrl -out crl/rootca.crl
```

Взглянём на содержимое CRL:

```
#  openssl crl -in crl/rootca.crl -text
```

Вы должны увидеть нечто подобное:

```
...
Revoked Certificates:
    Serial Number: 1000
        Revocation Date: ...
```

Поместим CRL в каталог, где его увидит клиентское ПО:

```
#  cp crl/rootca.crl /etc/ssl
```

Осталось только распространить этот CRL среди клиентов. Мы делаем это простым путём — копированием файлов по сети вручную.

Где работаем: **ldap-client**

На каждой клиентской машине надо будет запустиь:

```
#  scp user@ldap-srv.example.com:/etc/ssl/certs/rootca.crt /etc/ssl/certs
#  scp user@ldap-srv.example.com:/etc/ssl/rootca.crl /etc/ssl/
```

Настроим клиентскую конфигурацию в **/etc/ldap/ldap.conf** и укажем, где хранится CRL:

```
BASE  dc=example,dc=com
URI  ldap://ldap-srv.example.com
TLS_CACERT /etc/ssl/certs/rootca.crt
TLS_REQCERT demand
TLS_CRLFILE /etc/ssl/rootca.crl
TIMELIMIT 15
TIMEOUT  20
```

И попробуем получить доступ к серверу каталогов:

```
$  ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b cn=config -s base dn
ldap_start_tls: Connect error (-11)
        additional info: (unknown error code)
```

Наш CRL работает. Но к **slapd** теперь не подключиться. Надо выпустить новый сертификат для сервера.

Где работаем: **ldap-srv**

Сделаем это простой последовательностью команд. Мы повторяем пройденное, комментарии излишни:

```
#  cd /root/CA
#  openssl genrsa -out private/ldap-srv.example.com.key 4096
#  chmod 400 private/ldap-srv.example.com.key
#  openssl req -sha256 -new \
       -key private/ldap-srv.example.com.key -out certs/ldap-srv.example.com.csr \
       -subj /C=RU/ST=Moscow/L=Moscow/O=ExampleInc/OU=ITdept/CN=ldap-
srv.example.com/emailAddress=support@example.com
#  openssl ca -extensions usr_cert -notext -md sha256 \
       -keyfile private/rootca.key -cert certs/rootca.crt \
       -in certs/ldap-srv.example.com.csr -out certs/ldap-srv.example.com.crt
#  chmod 444 certs/ldap-srv.example.com.crt
#  cp certs/ldap-srv.example.com.crt private/ldap-srv.example.com.key /etc/ldap/ssl
#  chown openldap:openldap /etc/ldap/ssl/{ldap-srv.example.com.crt,ldap-srv.example.com.key}
#  chmod 0400 /etc/ldap/ssl/{ldap-srv.example.com.crt,ldap-srv.example.com.key}
```

Перезупустим демон **slapd** и убедимся, что к серверу можно подключиться:

```
#  service slapd restart
$  ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b cn=config -s base dn
Enter LDAP Password:
dn: cn=config
```

В целом по отзыву сертификатов... Иногда в реальных условиях лучше применить другой подход. Намного удобней, когда клиентские машины самостоятельно выясняют у сервера, действителен конкретный сертификат или нет. В выпускаемые сертификаты можно вносить запись « **CRL Distribution Points** ». Она заставит клиентов самих периодически скачивать новый CRL. Или можно использовать OCSP. Но эта тема выходит за рамки данного руководства.

4. Управление пользователями и группами в OpenLDAP

Где работаем: **ldap-srv**

На данном этапе у нас есть пустой каталог OpenLDAP. Мы можем проверить это, просто попытавшись извлечь из него информацию:

```
$  ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b dc=example,dc=com
Enter LDAP Password:
No such object (32)
```

Если Вам интересно, коды ошибок LDAP описаны в RFC 4511, в приложении A «Результирующие коды LDAP». В случае ошибки « **32** » сервер информирует нас, что запрашиваемый объект **dc=example,dc=com** не найден. Но такую же ошибку можно получить, если ACL сервера запрещают доступ.

Для работы нашего каталога понадобится создать корневой суффикс базы данных и добавить в него две организационных единицы (Organizational Unit, OU) для хранения пользователей и групп. Создадим LDIF-файл **4-users+groups.ldif**, в котором опишем эту информацию:

```
dn: dc=example,dc=com
dc: example
objectClass: top
objectClass: domain

dn: ou=users,dc=example,dc=com
ou: Users
objectClass: top
```

```
objectClass: organizationalUnit
description: Central location for UNIX users

dn: ou=groups,dc=example,dc=com
ou: Groups
objectClass: top
objectClass: organizationalUnit
description: Central location for UNIX groups
```

Добавим эту информацию в наш каталог:

```
$  ldapmodify -a -xZZWD cn=admin,dc=example,dc=com -f 4-users+groups.ldif
Enter LDAP Password:
adding new entry "dc=example,dc=com"
adding new entry "ou=users,dc=example,dc=com"
adding new entry "ou=groups,dc=example,dc=com"
```

Можем удостовериться в том, что информация добавлена:

```
$  ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com
Enter LDAP Password:
dn: dc=example,dc=com
dc: example
objectClass: top
objectClass: domain

dn: ou=users,dc=example,dc=com
ou: Users
objectClass: top
objectClass: organizationalUnit
description: Central location for UNIX users

dn: ou=groups,dc=example,dc=com
ou: Groups
objectClass: top
objectClass: organizationalUnit
description: Central location for UNIX groups
```

Теперь давайте добавим несколько групп:

- **`sysadmin`**, для объединения системных администраторов Linux;
- **`nssproxy`**, которая будет использоваться для опроса сервера, чтобы не делать анонимный опрос;
- **`test.group`**, для тестирования различных частей архитектуры PAM и LDAP. Тестирование — это всегда хорошо!

Этот список не окончательный и может быть дополнен, чтобы соответствовать нуждам Вашей организации.

Для добавления групп в каталог создадим новый LDIF, **`4-groups.ldif`**:

```
dn: cn=sysadmin,ou=groups,dc=example,dc=com
cn: sysadmin
objectClass: top
objectClass: posixGroup
gidNumber: 1100
description: UNIX systems administrators

dn: cn=nssproxy,ou=groups,dc=example,dc=com
cn: nssproxy
objectClass: top
objectClass: posixGroup
gidNumber: 801
description: Network Service Switch Proxy

dn: cn=test.group,ou=groups,dc=example,dc=com
cn: test.group
objectClass: top
objectClass: posixGroup
gidNumber: 1101
description: Test Group
```

Загрузим LDIF в наш каталог:

```
$  ldapmodify -a -xZZWD cn=admin,dc=example,dc=com -f 4-groups.ldif
Enter LDAP Password:
adding new entry "cn=sysadmin,ou=groups,dc=example,dc=com"
adding new entry "cn=nssproxy,ou=groups,dc=example,dc=com"
adding new entry "cn=test.group,ou=groups,dc=example,dc=com"
```

Настало время создать несколько пользователей. И вновь отмечаем, что список может быть расширен в соответствии с потребностями Вашей организации:

- **pablo**. Это я ;)
- **nssproxy**, который будет использоваться для опроса сервера, чтобы не делать опрос от анонимного пользователя.
- **test.user**. Как и тестовая группа, этот пользователь будет использоваться для проверки наших настроек.

Создадим LDIF-файл **4-users.ldif** для добавления пользователей. Пока не беспокойтесь о паролях, мы сейчас к ним вернёмся:

```
dn: cn=pablo,ou=users,dc=example,dc=com
uid: pablo
gecos: Pablo Sdoba
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
userPassword: {SSHA}RsAMqOI3647qg1gAZF3x2BKBnp0sEVfa
shadowLastChange: 15140
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 1100
gidNumber: 1100
homeDirectory: /home/pablo

dn: cn=nssproxy,ou=users,dc=example,dc=com
uid: nssproxy
gecos: Network Service Switch Proxy User
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
userPassword: {SSHA}RsAMqOI3647qg1gAZF3x2BKBnp0sEVfa
shadowLastChange: 15140
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/false
uidNumber: 801
gidNumber: 801
homeDirectory: /home/nssproxy

dn: cn=test.user,ou=users,dc=example,dc=com
uid: test.user
gecos: Test User
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
userPassword: {SSHA}RsAMqOI3647qg1gAZF3x2BKBnp0sEVfa
shadowLastChange: 15140
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 1101
gidNumber: 1101
homeDirectory: /home/test.user
```

Пользователи связаны с группами через атрибут **gidNumber**. Для того, чтобы добавить в группу других

пользователей, в запись группы необходимо добавить атрибут memberUID, перечислив в нём UID пользователей через запятую. Например, это может выглядеть вот так:

```
dn: cn=sysadmin,ou=groups,dc=example,dc=com
cn: sysadmin
objectClass: top
objectClass: posixGroup
gidNumber: 1100
description: UNIX systems administrators
memberUID: 801,1101
```

Добавьте пользователей в каталог:

```
$  ldapmodify -a -xZZWD cn=admin,dc=example,dc=com -f 4-users.ldif
Enter LDAP Password:
adding new entry "cn=pablo,ou=users,dc=example,dc=com"
adding new entry "cn=nssproxy,ou=users,dc=example,dc=com"
adding new entry "cn=test.user,ou=users,dc=example,dc=com"
```

Теперь установим пароли для пользователей. Повторите нижеследующую команду для всех пользователей. Обратите внимание, что сначала Вам будет предложено дважды ввести пароль изменяемой записи, а затем — пароль записи **cn=admin,dc=example,dc=com**:

```
$  ldappasswd -xZZWD cn=admin,dc=example,dc=com -S cn=nssproxy,ou=users,dc=example,dc=com
New password:
Re-enter new password:
Enter LDAP Password:
```

Если заглянуть в журнал **/var/log/slapd.log**, то можно увидеть строки, говорящие об изменении записи:

```
slapd[5319]: conn=1022 op=1 PASSMOD id="cn=nssproxy,ou=users,dc=example,dc=com" new
slapd[5319]: conn=1022 op=1 RESULT oid= err=0 text=
```

Для того, чтобы пользователь **nssproxy** имел доступ к информации нашего каталога, необходимо поправить ACL базы данных с пользователями и группами. Но какое у этой базы DN? Давайте вспомним:

```
 $  ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b cn=config dn | grep -i database
Enter LDAP Password:
dn: olcDatabase={-1}frontend,cn=config
dn: olcDatabase={0}config,cn=config
dn: olcDatabase={1}mdb,cn=config
dn: olcDatabase={2}monitor,cn=config
```

В разделе 2.6 мы дали нашей учетной записи администратора очень широкие права, поэтому он может заглянуть в конфигурацию **cn=config**.

Теперь мы знаем, что требуется отредактировать ACL записи **olcDatabase={1}mdb,cn=config**.

Давайте проверим, какие ACL настроены для данного DN (вывод отформатирован):

```
$  ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b olcDatabase={1}mdb,cn=config olcAccess
Enter LDAP Password:
dn: olcDatabase={1}mdb,cn=config
olcAccess: {0}to *
  by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external ,cn=auth" manage
  by * break
olcAccess: {1}to attrs=userPassword
  by self write
  by anonymous auth
olcAccess: {2}to *
  by self read
```

Мы должны немного изменить ACL, чтобы предоставить доступ пользователю **nssproxy** на чтение атрибутов учётных записей. Для этого создадим LDIF-файл **4-nssproxy.acl.ldif**:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
replace: olcAccess
olcAccess: {0}to *
```

```
  by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external ,cn=auth" manage
  by * break
-
add: olcAccess
olcAccess: {1}to attrs=userPassword
  by self write
  by anonymous auth
-
add: olcAccess
olcAccess: {2}to *
  by self read
  by dn.base="cn=nssproxy,ou=users,dc=example,dc=com" read
```

Загрузим LDIF с изменениями:

```
$  ldapmodify -xZZWD cn=admin,dc=example,dc=com -f 4-nssproxy.acl.ldif
Enter LDAP Password:
modifying entry "olcDatabase={1}mdb,cn=config"
```

Проверим, можем ли мы просматривать информацию в каталоге от имени пользователя **nssproxy**. Для этого выполним запрос с использованием его учётных данных. Результатом запроса должно быть всё DIT (Directory Information Tree). Опустим его, для краткости:

```
$  ldapsearch -xZZLLLWD cn=nssproxy,ou=users,dc=example,dc=com "(objectClass=*)"
Enter LDAP Password:
dn: dc=example,dc=com
...
```

Убедитесь, что другие учётные записи пользователей не имеют доступа к DIT:

```
$  ldapsearch -xZZLLLWD cn=pablo,ou=users,dc=example,dc=com "(objectClass=*)"
Enter LDAP Password:
No such object (32)
```

LDAP-браузер

База данных нашего сервера каталогов по-немногу заполняется объектами. С этого момента стоит подумать о выборе LDAP-браузера. Как правило, это приложение, которое позволяет наглядно отображать записи в базе данных и работать с ними. Лично у меня самый любимый — ldapvi. Но он консольный и имеет объективные недостатки. Попробуйте Apache Directory Studio. Он имеет нормальный GUI, бесплатен и поддерживает множество удобных функций для управления сервером каталогов. Этот браузер может быть установлен как отдельно, так и в качестве компонента программного средства Eclipse. Должен предупредить, при использовании Apache Directory Studio у меня возникали проблемы с доступом к серверу каталогов. Единственно разумное объяснение, которое я нашёл — ориентированность этого программного средства на использование с Apache Directory Server (ApacheDS). Так же неплохой вариант для работы из Windows — LDAP Admin.

5. Настройка аутентификации пользователей через OpenLDAP на клиенте

Где работаем: **ldap-client**

Перейдём к настройке клиентской рабочей станции. Для начала установим необходимые пакеты:

```
#  apt-get install ldap-utils libnss-ldapd libpam-ldapd
```

При установке нам будут предложены некоторые настройки . Далее мы всё равно внесём в конфигурацию изменения, но это нужно сделать, чтобы все пакеты нормально установились:

- URI сервера LDAP: **ldap://ldap-srv.example.com**;
- База поиска сервера LDAP: **dc=example,dc=com**;
- Имена настраиваемых служб: **group,netgroup,passwd,shadow**.

Прежде чем делать запросы к LDAP-серверу, проверим параметры клиента в **/etc/ldap/ldap.conf**:

```
BASE  dc=example,dc=com
URI  ldap://ldap-srv.example.com
TLS_CACERT /etc/ssl/certs/rootca.crt
TLS_REQCERT demand
TLS_CRLFILE /etc/ssl/rootca.crl
TIMELIMIT 15
TIMEOUT  20
```

Конечно, для того, чтобы всё заработало, сертификат нашего Certificate Authority (**rootca.crt**) и CRL-файл (**rootca.crl**) должны быть на месте.

Проверим работоспособность простым запросом. Результат опустим (Вы должны увидеть всё DIT):

```
$  ldapsearch -xZZLLLWD cn=nssproxy,ou=users,dc=example,dc=com
Enter LDAP Password:
dn: dc=example,dc=com
...
```

Поправим конфигурацию нашей локальной службы имён LDAP в файле **/etc/nslcd.conf**. Измените значение директивы « **bindpw** » на пароль записи **cn=nssproxy,ou=users,dc=example,dc=com**, который мы задавали ранее:

```
uid nslcd
gid nslcd
uri ldap://ldap-srv.example.com
base dc=example,dc=com
binddn cn=nssproxy,ou=users,dc=example,dc=com
bindpw пароль.пользователя.nssproxy
rootpwmoddn cn=admin,dc=example,dc=com
base group ou=groups,dc=example,dc=com
base passwd ou=users,dc=example,dc=com
base shadow ou=users,dc=example,dc=com
bind_timelimit 5
timelimit 10
idle_timelimit 60
ssl start_tls
tls_reqcert allow
tls_cacertfile /etc/ssl/certs/rootca.crt
nss_initgroups_ignoreusers bin,daemon,games,lp,mail,nobody,nslcd,root,sshd,sync,uucp
nss_initgroups_ignoreusers sys,man,news,proxy,www-data,backup,list,irc,gnats,landscape
```

Краткое описание использованных директив:

- С помощью директивы « **base** » мы сообщаем демону **nslcd**, где в DIT искать ту или иную информацию;
- « **bind_timelimit** » ограничивает время на установление соединения с сервером пятью секундами;

- « **timelimit** » устанавливает максимальное время ожидания ответа от сервера в 10 секунд;
- « **idle_timelimit** » заставит **nslcd** разорвать соединение с сервером, если в течении минуты в соединении не было никакой активности;
- « **ssl** » заставляет клиента использовать TLS при подключении к серверу;
- « **tls_reqcert** » и « **tls_cacertfile** » подобны директивам « **TLS_REQCERT** » и « **TLS_CACERT** » в конфигурации **/etc/ldap/ldap.conf** (необходимость проверки сертификата сервера и путь к корневому сертификату для выполнения проверки);
- « **nss_initgroups_ignoreusers** » описывает пользователей системы, поиск которых не нужно производить в DIT (чтобы мы могли работать с машиной при проблемах с доступом к серверу каталогов). В значение этой директивы следует внести имена всех общесистемных пользователей.

Поменяем права доступа к **nslcd.conf**, потому что теперь в нём хранится информация для аутентификации :

```
#  chmod 600 /etc/nslcd.conf
#  chown nslcd:nslcd /etc/nslcd.conf
```

Проверим содержимое **/etc/nsswitch.conf**:

```
passwd:         compat ldap
group:          compat ldap
shadow:         compat ldap

hosts:          files dns
networks:       files

protocols:      db files
services:       db files
ethers:         db files
rpc:            db files

netgroup:       nis ldap
```

Убедимся в том, что демон **nslcd** запускается при старте системы и перезапустим его:

```
#  update-rc.d nslcd defaults
System start/stop links for /etc/init.d/nslcd already exist.
#  service nslcd restart
* Starting LDAP connection daemon nslcd          [ OK ]
```

Убедимся, что в системе **ldap-client НЕТ** учётной записи с именем **nssproxy**. Следующая команда должна выполняться без результата:

```
$  grep nssproxy /etc/passwd
```

Убедимся так же, что кэширующий демон службы имён загружается при старте системы и перезапустим его:

```
#  update-rc.d nscd defaults
 System start/stop links for /etc/init.d/nscd already exist.
#  service nscd restart
 * Restarting Name Service Cache Daemon nscd
   ...done.
```

Сделаем пару запросов к LDAP-серверу, используя настроенную нами систему:

```
$  getent passwd test.user
test.user:x:1101:1101:Test User:/home/test.user:/bin/bash
$  getent group test.group
test.group:*:1101:
```

Для того, чтобы проверить доступность информации о пароле потребуется выполнить **getent** от имени пользователя **root**:

```
#  getent shadow test.user
```

```
test.user:*:15140:0:99999:7:::0
```

Не беспокойтесь, хэш пароля мы видеть не должны.

Отлично, всё работает! Вышеприведённые результаты команд означают, что система **ldap-client** может осуществлять поиск по данным пользователей и групп в нашем каталоге OpenLDAP.

Создадим домашний каталог нашего тестового пользователя и установим для него права доступа:

```
#  mkdir /home/test.user
#  chown test.user:test.group /home/test.user
```

Запустим в отдельном терминале вывод журнала аутентификации:

```
#  tail -f /var/log/auth.log
```

В другом терминале выполним:

```
$  su - test.user
```

Мы должны получить приглашение командной строки пользователя **test.user**.

Теперь попробуем зайти на машину **ldap-client** по сети с использованием **ssh** под учётной записью **test.user**.

Демон **ssh** в конфигурации по-умолчанию должен работать с поддержкой PAM (и, соответственно, поддерживать аутентификацию через LDAP). Но на всякий случай приведём рабочую конфигурацию **/etc/ssh/sshd_config**:

```
Port 22
Protocol 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
UsePrivilegeSeparation yes
KeyRegenerationInterval 3600
ServerKeyBits 1024
LogLevel INFO
LoginGraceTime 120
PermitRootLogin without-password
StrictModes yes
RSAAuthentication yes
PubkeyAuthentication yes
IgnoreRhosts yes
RhostsRSAAuthentication no
HostbasedAuthentication no
PermitEmptyPasswords no
ChallengeResponseAuthentication no
X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
AcceptEnv LANG LC_*
Subsystem sftp /usr/lib/openssh/sftp-server
UsePAM yes

AddressFamily inet
AllowGroups sysadmin test.group
SyslogFacility AUTHPRIV
PasswordAuthentication yes
AllowTcpForwarding no
ClientAliveInterval 120
ClientAliveCountMax 2
```

Часть файла до пустой строки — конфигурация по-умолчанию. Далее — добавленное нами. Обратите внимание на строку с директивой **AllowGroups**. С помощью неё мы ограничиваем список групп пользователей, которые могут быть аутентифицированы через **ssh**. За информацией по остальным директивам обратитесь к документации.

Проверим работу с использованием какой-нибудь третьей машины. Например, выполним на нашем DNS-сервере (**dns-srv**):

```
$  ssh test.user@ldap-client.example.com
Password:
test.user@ldap-client:~$
```

Вывод последней команды сокращён. Если появляется приглашение командной строки, значит всё в порядке!

Где работаем: **ldap-srv**

Давайте заглянем в **/var/log/slapd.log** на нашем сервере. Мы можем обнаружить там следующие строки:

```
ldap-srv slapd[1304]: <= mdb_equality_candidates: (objectClass) not indexed
ldap-srv slapd[1304]: <= mdb_equality_candidates: (uid) not indexed
```

С помощью следующей команды мы можем убедиться, что в нашем каталоге пока нет никаких индексов:

```
$  ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b olcDatabase={1}mdb,cn=config olcDbIndex
Enter LDAP Password:
dn: olcDatabase={1}mdb,cn=config
```

Это значит, что в нашей базе данных надо создать индексы для атрибутов из журнала **/var/log/slapd.log**. Поэтому создадим ещё один LDIF-файл **5-posixAccount.indexes.ldif** и запишем в него:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: default pres,eq
-
add: olcDbIndex
olcDbIndex: uid
-
add: olcDbIndex
olcDbIndex: cn,sn pres,eq,sub
-
add: olcDbIndex
olcDbIndex: objectClass eq
-
add: olcDbIndex
olcDbIndex: memberUid eq
-
add: olcDbIndex
olcDbIndex: uniqueMember eq
-
add: olcDbIndex
olcDbIndex: uidNumber
-
add: olcDbIndex
olcDbIndex: gidNumber eq
```

Зачем мелочиться? Укажем по-больше индексируемых атрибутов. И загрузим конфигурацию в наш каталог:

```
$  ldapadd -xZZWD cn=admin,dc=example,dc=com -f posixAccount.indexes.ldif
Enter LDAP Password:
modifying entry "olcDatabase={1}mdb,cn=config"
```

Проверим результат изменений:

```
$  ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b olcDatabase={1}mdb,cn=config olcDbIndex
Enter LDAP Password:
dn: olcDatabase={1}mdb,cn=config
olcDbIndex: default pres,eq
olcDbIndex: uid
olcDbIndex: cn,sn pres,eq,sub
olcDbIndex: objectClass eq
olcDbIndex: memberUid eq
```

```
olcDbIndex: uniqueMember eq
olcDbIndex: uidNumber
olcDbIndex: gidNumber eq
```

Отлично! Теперь в журнале **/var/log/slapd.log** не должно быть ошибок.

6. Настройка OpenLDAP в качестве хранилища правил sudo

Для начала хорошей идеей будет ознакомиться с этими источниками:

Официальный сайт sudo

Руководство по sudoers и LDAP

Файл README.LDAP проекта sudo

Хорошая статья на русском в вики Archlinux про sudo

6.1 Конвертация файла sudoers в конфигурацию OpenLDAP

Где работаем: **ldap-srv**

Для удобства создадим новый каталог:

```
$  mkdir ~/sudo
$  cd ~/sudo
```

За основу возьмём вот такой вполне обычный файл с правилами **sudo** и сохраним его как **6.1-sudoers.source**:

```
Defaults      env_reset
Defaults      mail_badpass
Defaults      secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

root          ALL=(ALL) ALL

%sysadmin     ALL=(ALL) ALL
```

Для конвертации этого файла в LDIF воспользуемся скриптом **/usr/share/doc/sudo-ldap/sudoers2ldif.gz** из пакета **sudo-ldap**:

```
$  zcat /usr/share/doc/sudo-ldap/sudoers2ldif.gz > 6.1-sudoers2ldif
$  SUDOERS_BASE=ou=sudo,ou=services,dc=example,dc=com perl 6.1-sudoers2ldif 6.1-sudoers.source >
sudoers.ldif
```

Между прочим, можно и не создавать отдельный файл **6.1-sudoers.source**, а просто в последней команде вместо него указать Ваш текущий **/etc/sudoers**. Таким образом Вы преобразуете в LDIF свою любимую конфигурацию. :)

Итак, у нас теперь есть **sudoers** в формате LDIF. Но в нашей службе каталогов нет контейнера для правил **sudo**! У нас, в сущности, даже нет контейнера для служб. Поэтому добавим в начало получившегося у нас **6.1-sudoers.ldif** несколько строк для контейнера служб (первый блок строк) и входящего в него контейнера правил **sudo** (второй блок строк). Итоговый файл **6.1-sudoers.ldif** для **6.1-sudoers.source** будет выглядеть так:

```
dn: ou=services,dc=example,dc=com
ou: Services
objectClass: top
objectClass: organizationalUnit
```

```
description: Group all services under this OU

dn: ou=sudo,ou=services,dc=example,dc=com
objectClass: organizationalUnit
description: sudo
objectClass: top

dn: cn=defaults,ou=sudo,ou=services,dc=example,dc=com
objectClass: top
objectClass: sudoRole
cn: defaults
description: Default sudoOption's go here
sudoOption: env_reset
sudoOption: mail_badpass
sudoOption: secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
sudoOrder: 1

dn: cn=root,ou=sudo,ou=services,dc=example,dc=com
objectClass: top
objectClass: sudoRole
cn: root
sudoUser: root
sudoHost: ALL
sudoRunAsUser: ALL
sudoCommand: ALL
sudoOrder: 2

dn: cn=%sysadmin,ou=sudo,ou=services,dc=example,dc=com
objectClass: top
objectClass: sudoRole
cn: %sysadmin
sudoUser: %sysadmin
sudoHost: ALL
sudoRunAsUser: ALL
sudoCommand: ALL
sudoOrder: 3
```

Заметьте, директива **secure_path** определена без использования кавычек!

Небольшое примечание по поводу скрипта **sudoers2ldif**. Если в Вашем **sudoers** файле встречаются строки вида « **env_keep = COLORS** », то после всех проведённых с файлом **sudoers.ldif** манипуляций надо сделать ещё один фокус — убрать лишние пробелы. Так, чтобы получилось « **env_keep=COLORS** ». Иначе при загрузке LDIF мы получим ошибки. Провернуть это можно с помощью одной команды:

```
$  sed -i.bak -e "s/ = /=/g" -re 's/= {0,2}"/=/g' -e 's/"$//g' -e "s/p \+/p+/g" 6.1-sudoers.ldif
```

Загрузим нашу конфигурацию в каталог:

```
$  ldapmodify -axZZWD cn=admin,dc=example,dc=com -f 6.1-sudoers.ldif
Enter LDAP Password:
adding new entry "ou=services,dc=example,dc=com"
adding new entry "ou=sudo,ou=services,dc=example,dc=com"
adding new entry "cn=defaults,ou=sudo,ou=services,dc=example,dc=com"
adding new entry "cn=root,ou=sudo,ou=services,dc=example,dc=com"
adding new entry "cn=%sysadmin,ou=sudo,ou=services,dc=example,dc=com"
```

Убедимся, что обычные пользователи не могут просмотреть конфигурацию **sudoers**:

```
$  ldapsearch -xZZLLLWD cn=pablo,ou=users,dc=example,dc=com -b ou=sudo,ou=services,dc=example,dc=com
Enter LDAP Password:
No such object (32)
```

А вот наш пользователь **nssproxy**, напротив, должен иметь к ней доступ:

```
$  ldapsearch -xZZLLLWD cn=nssproxy,ou=users,dc=example,dc=com -b ou=sudo,ou=services,dc=example,dc=com
Enter LDAP Password:
dn: ou=sudo,ou=services,dc=example,dc=com
objectClass: organizationalUnit
...
```

Вы должны были увидеть вывод, почти идентичный содержимому файла **6.1-sudoers.ldif**. Если так, то

продолжаем.

6.2 Настройка sudo клиента

Где работаем: **ldap-client**

Теперь, когда у нас есть работающая схема **sudo** (**dn: cn={13}sudo,cn=schema,cn=config**) и LDAP-версия файла **sudoers**, мы можем настроить клиентские машины для запроса данных **sudoers** с сервера OpenLDAP.

Прежде чем установить пакет **sudo-ldap** необходимо задать пароль для учетной записи **root**:

```
#  passwd
Введите новый пароль UNIX:
Повторите ввод нового пароля UNIX:
```

Установим **sudo-ldap**:

```
#  apt-get install sudo-ldap
```

Конфигурация LDAP для **sudo** по-умолчанию считывается из ссылки **/etc/sudo-ldap.conf** на **/etc/ldap/ldap.conf**. Нам это будет не очень удобно, потому что в эту конфигурацию надо снова поместить пароль **nssproxy**. Поэтому удалим эту символическую ссылку и заменим её на файл с таким же именем:

```
# rm /etc/sudo-ldap.conf
# touch /etc/sudo-ldap.conf
```

Запишем в файл **/etc/sudo-ldap.conf**:

```
BASE    dc=example,dc=com
URI     ldap://ldap-srv.example.com
BINDDN  cn=nssproxy,ou=users,dc=example,dc=com
BINDPW  пароль.пользователя.nssproxy
TLS_CACERTFILE  /etc/ssl/certs/rootca.crt
TLS_CRLFILE     /etc/ssl/rootca.crl
TLS_CHECKPEER   no
TIMELIMIT       15
TIMEOUT         20
SUDOERS_BASE    ou=sudo,ou=services,dc=example,dc=com
#SUDOERS_DEBUG 2
```

Настроим права доступа на этот файл (в нём теперь есть пароль!):

```
#  chmod 600 /etc/sudo-ldap.conf
#  chown root:root /etc/sudo-ldap.conf
```

Файл сертификата нашего корневого удостоверяющего центра (**rootca.crt**) мы уже скопировали на **ldap-client** в разделе 5.

Убедимся, что в файле **/etc/nsswitch.conf** директива **sudoers** выглядит так или добавим её:

```
sudoers:        files ldap
```

Проверим работоспособность наших настроек:

```
$ sudo -l -U pablo
Matching Defaults entries for pablo on ldap-client:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
env_reset, mail_badpass, secure_path=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin:/usr/local/sbin

User pablo may run the following commands on ldap-client:
    (ALL) ALL
```

Мы видим как локальные настройки **/etc/sudoers**, так и информацию из нашего каталога OpenLDAP. Значит всё работает!

Где работаем: **ldap-srv**

Если мы вновь заглянем в журнал **/var/log/slapd.log**, то увидим там ошибки такого вида:

```
ldap-srv slapd[868]: <= mdb_equality_candidates: (sudoUser) not indexed
```

Для того, чтобы это исправить сделаем вот такой небольшой LDIF-файл **6.2-sudoers.indexes.ldif** с новыми индексами:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: sudoUser eq,sub
-
add: olcDbIndex
olcDbIndex: sudoHost eq
```

И загрузим его в каталог:

```
$  ldapmodify -axZZWD cn=admin,dc=example,dc=com  -f 6.2-sudoers.indexes.ldif
Enter LDAP Password:
modifying entry "olcDatabase={1}mdb,cn=config"
```

Теперь при выполнении на **ldap-client** команд с **sudo** ошибок в журнале сервера быть не должно. На этом реализация хранилища настроек **sudo** закончена.

7. OpenLDAP как хранилище карт автоматического монтирования для autofs

В этом разделе мы разберёмся, как настроить сервер NFS. Клиент NFS (**ldap-client**) будет монтировать ресурсы сервера с помощью **autofs** версии 5. При этом карты автоматического монтирования будут располагаться на сервере OpenLDAP (**ldap-srv**). Мы так же создадим центральный NFS репозиторий для программного обеспечения.

7.1 Создание сервера NFS

Развернём NFS-сервер по аналогии с первым разделом. Для начала обновим информацию на DNS сервере (в нашем случае — в файле **/etc/hosts**):

Где работаем: **dns-srv**

```
127.0.0.1            localhost
192.168.122.140      dns-srv.example.com

192.168.122.150      ldap-srv.example.com
192.168.122.151      ldap-client.example.com
192.168.122.154      nfs-srv.example.com
```

И перезапустим демон DNS, чтобы изменения вступили в силу:

```
#  service dnsmasq restart
```

Теперь настроим наш NFS-сервер.

Где работаем: **nfs-srv**

Настроим сеть в файле **/etc/network/interfaces**:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.168.122.154
        netmask 255.255.255.0
        gateway 192.168.122.1
        dns-nameservers 192.168.122.140
```

Настроим имя NFS-сервера. В Ubuntu 14.04 останов или перезапуск сервиса **networking** не поддерживается, поэтому придётся сделать так:

```
#  echo 'nfs-srv.example.com' > /etc/hostname
#  reboot
```

Проверим разрешение имени будущего сервера LDAP:

```
$  dig nfs-srv.example.com +short
192.168.122.150
```

Проверим обратное разрешение имени:

```
$  dig -x `dig nfs-srv.example.com +short` +short
nfs-srv.example.com.
```

7.2 Интеграция сервера NFS с сервером каталогов

Где работаем: **nfs-srv**

Установим необходимые пакеты:

```
#  passwd
Введите новый пароль UNIX:
Повторите ввод нового пароля UNIX:
passwd: password updated successfully
#  apt-get install nfs-common nfs-kernel-server nfs4-acl-tools libnss-ldapd libpam-ldapd sudo-ldap
```

Заменим ссылку **/etc/sudo-ldap.conf** на пустой файл:

```
#  rm /etc/sudo-ldap.conf
#  touch /etc/sudo-ldap.conf
```

Так как наши пользовательские учётные записи хранятся на сервере OpenLDAP, NFS-сервер тоже должен быть LDAP-клиентом. Произведём их интеграцию. Для этого надо внести изменения в четыре конфигурационных файла и произвести ряд нехитрых манипуляций.

Конфигурация LDAP-клиента в **/etc/ldap/ldap.conf**:

```
BASE  dc=example,dc=com
URI  ldap://ldap-srv.example.com
```

```
TLS_CACERT /etc/ssl/certs/rootca.crt
TLS_REQCERT demand
TLS_CRLFILE /etc/ssl/rootca.crl
TIMELIMIT 15
TIMEOUT  20
```

Конфигурация LDAP для **sudo** в **/etc/sudo-ldap.conf**:

```
BASE  dc=example,dc=com
URI  ldap://ldap-srv.example.com
TLS_CACERT /etc/ssl/certs/rootca.crt
TLS_REQCERT demand
TLS_CRLFILE /etc/ssl/rootca.crl
BINDDN  cn=nssproxy,ou=users,dc=example,dc=com
BINDPW  пароль.пользователя.nssproxy
SUDOERS_BASE    ou=sudo,ou=services,dc=example,dc=com
TIMELIMIT 15
TIMEOUT  20
```

Конфигурация локальной службы имён LDAP в **/etc/nslcd.conf**:

```
uid nslcd
gid nslcd
uri ldap://ldap-srv.example.com
base dc=example,dc=com
binddn cn=nssproxy,ou=users,dc=example,dc=com
bindpw пароль.пользователя.nssproxy
rootpwmoddn cn=admin,dc=example,dc=com
base group ou=groups,dc=example,dc=com
base passwd ou=users,dc=example,dc=com
base shadow ou=users,dc=example,dc=com
bind_timelimit 5
timelimit 10
idle_timelimit 60
ssl start_tls
tls_reqcert never
tls_cacertfile /etc/ssl/certs/rootca.crt
nss_initgroups_ignoreusers bin,daemon,games,lp,mail,nobody,nslcd,root,sshd,sync,uucp
nss_initgroups_ignoreusers sys,man,news,proxy,www-data,backup,list,irc,gnats,landscape
```

Конфигурация Name Service Switch (NSS) в **/etc/nsswitch.conf**:

```
passwd:         compat ldap
group:          compat ldap
shadow:         compat ldap

hosts:          files dns
networks:       files

protocols:      db files
services:       db files
ethers:         db files
rpc:            db files

netgroup:       nis ldap
sudoers:        files ldap
```

Скопируем сертификат нашего удостоверяющего центра и CRL на NFS-сервер. Подставим вместо **user** имя Вашей пользовательской учётной записи на **ldap-srv**:

```
#  scp user@ldap-srv.example.com:/etc/ssl/certs/rootca.crt /etc/ssl/certs/
#  scp user@ldap-srv.example.com:/etc/ssl/rootca.crl /etc/ssl/
```

Завершая интеграцию с LDAP-сервером, поменяем права доступа к **/etc/nslcd.conf** и **/etc/sudo-ldap.conf**, потому что теперь в них хранится информация для аутентификации. Затем перезапустим сервисы **nslcd** и **nscd**:

```
#  chmod 600 /etc/{nslcd.conf,sudo-ldap.conf}
#  chown root:root /etc/sudo-ldap.conf
#  chown nslcd:nslcd /etc/nslcd.conf
#  service nslcd restart
 * Restarting LDAP connection daemon nslcd                                [ OK ]
```

```
#  service nscd restart
 * Restarting LDAP connection daemon nscd                                [ OK ]
```

Теперь удостоверимся, что пользователя **test.user** нет в локальной конфигурации в **/etc/passwd**:

```
$  grep ^test.user /etc/passwd || echo No such user
No such user
```

А теперь проверим, что мы можем получить информацию об этом пользователе от нашего сервера каталогов. Напоминаю, что данные **shadow** можно получить только выполнив **getent** от имени суперпользователя (UID=0):

```
$  getent passwd test.user
test.user:x:1101:1101:Test User:/home/test.user:/bin/bash
$  getent group test.group
test.group:*:1101:
#  getent shadow test.user
test.user:*:15140:0:99999:7:::0
```

Можем переходить к настройке сетевых ресурсов.

7.3 Настройка сетевых ресурсов NFS-сервера

Где работаем: **nfs-srv**

Создайте каталоги, в которых будут находиться пользовательские домашние каталоги (**/export/home**) и центральный репозиторий программного обеспечения (**/export/install**):

```
#  mkdir -p /export/home /export/install
```

Держите в уме, что нам необходим домашний каталог для каждого пользователя.

Теперь можем создать тестовый пример. Для этих целей у нас и есть учётная запись **test.user**:

```
#  mkdir /export/home/test.user
#  cp /etc/skel/.* /export/home/test.user
#  chown -R test.user:test.group /export/home/test.user
```

Если забыли пароль учётной записи **test.user**, теперь можем его поменять:

```
$  ldappasswd -xZZWD cn=admin,dc=example,dc=com -S cn=test.user,ou=users,dc=example,dc=com
```

Укажем экспортируемые каталоги в **/etc/exports**:

```
/export/home    *.example.com(rw,no_subtree_check)
/export/install *.example.com(rw,no_subtree_check)
```

Перезапустим демон NFS:

```
#  service nfs-kernel-server restart
```

И проверим работоспособность:

```
$  showmount -e
Export list for nfs-srv.example.com:
/export/install *.example.com
/export/home    *.example.com
```

Каталоги успешно экспортируются. С помощью сервера OpenLDAP мы будем централизованно управлять тем, что и куда монтировать на клиентских машинах.

7.4 Создание карт автоматического монтирования на сервере LDAP

Где работаем: **ldap-srv**

Теперь, когда сервер NFS настроен, мы должны добавить новую схему данных для автоматического монтирования в базу данных нашего сервера каталогов. Самый простой способ её получить — установить пакет **autofs-ldap**:

```
#  apt-get install autofs-ldap
```

Искомая схема данных - **/etc/ldap/schema/autofs.schema**. Как Вы уже возможно догадались, прежде всего её надо перевести в формат LDIF. В разделе 2.5 для этого мы использовали скрипт **2.5-schema-ldif.sh**. Схема **autofs.schema** зависит от схемы **core.schema**. Поэтому перед конвертацией надо добавить в итоговый **autofs.schema** директиву **include**. Выполним в консоли:

```
$  echo "include /etc/ldap/schema/core.schema" > ~/ldap/autofs.schema
$  cat /etc/ldap/schema/autofs.schema >> ~/ldap/autofs.schema
$  SCHEMAS='autofs.schema' ~/ldap/2.5-schema-ldif.sh
config file testing succeeded
```

Посмотрим содержимое получившегося LDIF. Он довольно короткий:

```
$  cat ~/ldap/autofs.ldif
dn: cn=autofs,cn=schema,cn=config
objectClass: olcSchemaConfig
cn: autofs
olcAttributeTypes: {0}( 1.3.6.1.1.1.1.25 NAME 'automountInformation' DESC 'Inf
 ormation used by the autofs automounter' EQUALITY caseExactIA5Match SYNTAX 1.
 3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
olcObjectClasses: {0}( 1.3.6.1.1.1.1.13 NAME 'automount' DESC 'An entry in an
 automounter map' SUP top STRUCTURAL MUST ( cn $ automountInformation $ object
 class ) MAY description )
olcObjectClasses: {1}( 1.3.6.1.4.1.2312.4.2.2 NAME 'automountMap' DESC 'An gro
 up of related automount objects' SUP top STRUCTURAL MUST ou )
```

Переместим его к остальным наборам схем:

```
#  mv autofs.ldif /etc/ldap/schema
```

Загрузим его в конфигурацию сервера каталогов:

```
$  ldapadd -xZZWD cn=admin,dc=example,dc=com  -f /etc/ldap/schema/autofs.ldif
Enter LDAP Password:
adding new entry "cn=autofs,cn=schema,cn=config"
```

Можем удостовериться в том, что схема данных загружена:

```
$  ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b cn=schema,cn=config dn | grep auto
Enter LDAP Password:
dn: cn={14}autofs,cn=schema,cn=config
```

Создадим файл конфигурации с картами автоматического монтирования (**7.4-autofs-map.ldif**):

```
# Контейнер для конфигураций autofs
dn: ou=autofs,ou=services,dc=example,dc=com
ou: AutoFS
objectClass: top
objectClass: organizationalUnit
description: Automount maps

# Контейнер для карты auto.master
dn: ou=auto.master,ou=autofs,ou=services,dc=example,dc=com
```

```
ou: auto.master
objectClass: top
objectClass: automountMap

# Точка монтирования /nfs
dn: cn=/nfs,ou=auto.master,ou=autofs,ou=services,dc=example,dc=com
cn: /nfs
objectClass: top
objectClass: automount
automountInformation: ldap:ou=auto.nfs,ou=autofs,ou=services,dc=example,dc=com rsize=8192,wsize=8192

# Контейнер для карты auto.nfs
dn: ou=auto.nfs,ou=autofs,ou=services,dc=example,dc=com
ou: auto.nfs
objectClass: top
objectClass: automountMap

# Конфигурация дочерней точки монтирования home карты auto.nfs
dn: cn=home,ou=auto.nfs,ou=autofs,ou=services,dc=example,dc=com
objectClass: top
objectClass: automount
cn: home
automountInformation: nfs-srv.example.com:/export/home

# Конфигурация дочерней точки монтирования install карты auto.nfs
dn: cn=install,ou=auto.nfs,ou=autofs,ou=services,dc=example,dc=com
objectClass: top
objectClass: automount
cn: install
automountInformation: nfs-srv.example.com:/export/install
```

И добавим новую информацию в службу каталогов:

```
$  ldapadd -xZZWD cn=admin,dc=example,dc=com  -f 7.4-autofs-map.ldif
```

Проверить сделанные изменения можно так:

```
$  ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b ou=autofs,ou=services,dc=example,dc=com dn
Enter LDAP Password:
dn: ou=autofs,ou=services,dc=example,dc=com
dn: ou=auto.master,ou=autofs,ou=services,dc=example,dc=com
dn: cn=/nfs,ou=auto.master,ou=autofs,ou=services,dc=example,dc=com
dn: ou=auto.nfs,ou=autofs,ou=services,dc=example,dc=com
dn: cn=home,ou=auto.nfs,ou=autofs,ou=services,dc=example,dc=com
dn: cn=install,ou=auto.nfs,ou=autofs,ou=services,dc=example,dc=com
```

Отлично! Теперь мы готовы настроить клиент для сервера NFS.

7.5 Настройка клиента для сервера NFS

Где работаем: **ldap-client**

В предыдущих разделах мы уже произвели интеграцию клиента с сервером каталогов. В этом разделе настроим на нём демон автоматического монтирования так, чтобы демон брал карты монтирования с сервера каталогов.

Как всегда, для начала установим несколько пакетов:

```
#  apt-get install nfs-common nfs4-acl-tools autofs5 autofs5-ldap
```

Настроим демон, транслирующий UID/GID в имена и наоборот (**rpc.idmapd**) в файле **/etc/idmapd.conf**:

```
[General]

Verbosity = 0
```

```
Pipefs-Directory = /run/rpc_pipefs
Domain = example.com

[Mapping]

Nobody-User = nobody
Nobody-Group = nogroup

[Translation]

Method = nsswitch
GSS-Methods = nsswitch
```

Добавим в **/etc/nsswitch.conf** строку:

```
automount: ldap
```

Настроим демон автоматического монтирования в файле **/etc/default/autofs**:

```
TIMEOUT=300
BROWSE_MODE="no"
MOUNT_NFS_DEFAULT_PROTOCOL=4
# Измените LOGGING на "none", если собираетесь использовать конфигурацию в продакшене
LOGGING="debug"
LDAP_URI="ldap://ldap-srv.example.com"
SEARCH_BASE="ou=autofs,ou=services,dc=example,dc=com"
MAP_OBJECT_CLASS="automountMap"
ENTRY_OBJECT_CLASS="automount"
MAP_ATTRIBUTE="ou"
ENTRY_ATTRIBUTE="cn"
VALUE_ATTRIBUTE="automountInformation"
USE_MISC_DEVICE="yes"
```

Настроим аутентификацию демона автоматического монтирования в файле **/etc/autofs_ldap_auth.conf**:

```
<?xml version="1.0" ?>
 <autofs_ldap_sasl_conf
  usetls="yes"
  tlsrequired="no"
  authrequired="simple"
  user="cn=nssproxy,ou=users,dc=example,dc=com"
  secret="пароль.пользователя.nssproxy"
/>
<!-- EOF -->
```

Снова прячем файл с аутентификационной информацией от посторонних глаз, иначе демон автоматического монтирования его не примет:

```
#  chmod 0600 /etc/autofs_ldap_auth.conf
#  chown root:root /etc/autofs_ldap_auth.conf
```

Убедимся, что **rpcbind** стартует при загрузке системы и перезапустим его. То же самое сделаем с демоном **autofs**. Для корректной работы **rpcbind** лично мне пришлось перезагрузить клиентскую рабочую станцию. Вывод убран для краткости:

```
#  update-rc.d rpcbind defaults
#  service rpcbind restart
#  update-rc.d autofs defaults
#  service autofs restart
```

Проверьте файлы журналов сервера (**nfs-srv**) и клиента (**ldap-client**). Если всё в порядке, можно проверять созданную нами конфигурацию на клиентской машине:

```
$  ls -a /nfs/
.  ..
$  cd /nfs/home
$  pwd
/nfs/home
$  cd /nfs/install
$  pwd
```

```
/nfs/install
$  ls -a /nfs
.  ..  home  install
$ df -h /nfs/home
Файл.система                     Размер Использовано  Дост Использовано% Смонтировано в
nfs-srv.example.com:/export/home   3,6G         1,6G  1,8G           47% /nfs/home
$ df -h /nfs/install
Файл.система                       Размер Использовано  Дост Использовано% Смонтировано в
nfs-srv.example.com:/export/install   3,6G         1,6G  1,8G           47% /nfs/install
```

Если Вы видите похожий вывод, то цель достигнута!

8. Репозиторий NFS netgroup на основе OpenLDAP для autofs

В этом разделе мы настроим сервер каталогов для хранения конфигурации NFS **`netgroup`**. Это механизм разграничения доступа к сетевым ресурсам. Затем мы настроим клиента NFS, чтобы проверить, что наша конфигурация **`netgroup`** действительно работает.

ВНИМАНИЕ! Прежде чем переходить к следующему разделу и добавлять группы **`oracle`** и **`itdept`**, ознакомьтесь с проблемой, описанной в разделе 8.4. Для того, чтобы она у Вас не появилась, можете прямо сейчас выполнить инструкции раздела 8.4.2 и потом вернуться сюда. А можете и не выполнять. Зависит от того, является ли это проблемой в Вашем конкретном случае.

8.1 Настройка сервера OpenLDAP

Где работаем: **ldap-srv**

Нам понадобится схема наборов данных Network Information Service (NIS). В разделе 2.5 мы уже добавили её в конфигурацию сервера каталогов. Проверим, что она на месте:

```
$  ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b cn=schema,cn=config dn | grep nis
Enter LDAP Password:
dn: cn={10}nis,cn=schema,cn=config
```

Проверим, содержит ли эта схема атрибуты для **`netgroup`**:

```
$ sudo ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b cn={10}nis,cn=schema,cn=config | grep NAME | cut
-d' ' -f5 | grep -i netgroup
Enter LDAP Password:
'memberNisNetgroup'
'nisNetgroupTriple'
'nisNetgroup'
```

Хорошо. Но у нас пока нет никаких записей, которые используют эти атрибуты. Создадим LDIF-файл **`8.1-netgroup.ldif`**, который добавит в службу каталогов контейнер для конфигураций **`netgroup`** и несколько примеров групп **`netgroup`**:

```
# Контейнер для конфигураций netgroup
dn: ou=netgroup,ou=services,dc=example,dc=com
ou: netgroup
objectClass: top
objectClass: organizationalUnit
description: NFS Netgroups

# Первый пример группы netgroup
dn: cn=oracle,ou=netgroup,ou=services,dc=example,dc=com
objectClass: top
objectClass: nisNetgroup
```

```
cn: oracle
nisNetgroupTriple: (oracle.example.com,,)
description: All Oracle machines

# Второй пример группы netgroup
dn: cn=itdept,ou=netgroup,ou=services,dc=example,dc=com
objectClass: top
objectClass: nisNetgroup
cn: itdept
nisNetgroupTriple: (ldap-client.example.com,,)
description: IT department machines
```

Будьте осторожны с использованием FQDN в наборах netgroup!

Если Вы сделаете это неверно, ничего не заработает. Например, такая запись ошибочна:

```
nisNetgroupTriple: (ldap-client.example.com,,example.com)
```

А такая запись — верная:

```
nisNetgroupTriple: (ldap-client.example.com,,)
```

Такой вариант записи тоже допускается:

```
nisNetgroupTriple: (ldap-client,,example.com)
```

Выберите синтаксис, который больше по душе, и придерживайтесь его.

Добавим новые записи в службу каталогов:

```
$  ldapadd -xZZWD cn=admin,dc=example,dc=com -f 8.1-netgroup.ldif
Enter LDAP Password:
adding new entry "ou=netgroup,ou=services,dc=example,dc=com"
adding new entry "cn=oracle,ou=netgroup,ou=services,dc=example,dc=com"
adding new entry "cn=itdept,ou=netgroup,ou=services,dc=example,dc=com"
```

8.2 Настройка сервера NFS

Где работаем: **nfs-srv**

Наш сервер NFS уже интегрирован с сервером каталогов в соответствии с разделом 7. Нижеследующие манипуляции без этого не заработают.

Измените значение конфигурации **`netgroup`** в файле **`/etc/nsswitch.conf`**, чтобы информация о группах **`netgroup`** запрашивалась только у сервера каталогов:

```
netgroup:     ldap
```

Проверим, что возвращает запрос конфигурации **`netgroup`** к нашему серверу каталогов:

```
$ getent netgroup
Перечисление не поддерживается для netgroup
```

Хм, интересно. Так как **`netgroup`**, в сущности, — механизм безопасности, у нас не получится просмотреть весь их список сразу. Значит, надо проверить их по-отдельности:

```
$ getent netgroup oracle
oracle               (oracle.example.com,,)
$ getent netgroup itdept
itdept                  (ldap-client.example.com,,)
```

Вот так выглядит запись о событии запроса « **getent netgroup itdept** » в журнале **slapd.log** на сервере каталогов:

```
slapd[870]: conn=1448 op=2 SRCH base="dc=example,dc=com" scope=2 deref=0
filter="(&(objectClass=nisNetgroup)(cn=itdept))"
slapd[870]: conn=1448 op=2 SRCH attr=cn nisNetgroupTriple memberNisNetgroup
slapd[870]: conn=1448 op=2 SEARCH RESULT tag=101 err=0 nentries=1 text=
```

Теперь изменим файл с записями экспортируемых каталогов **/etc/exports**, чтобы он выглядел так:

```
/export/home    @itdept(rw,no_subtree_check)
/export/install @itdept(rw,no_subtree_check)
```

Обратите внимание, что мы определили нашу новую группу с помощью знака « **@** ». Клиентская машина **ldap-client** — часть этой группы **netgroup**. Поэтому она должна иметь возможность монтировать указанные каталоги. Но прежде чем мы это проверим, нужно обновить конфигурацию демона NFS:

```
#  service nfs-kernel-server reload
$  showmount -e
Export list for nfs-srv.example.com:
/export/install @itdept
/export/home    @itdept
```

Теперь можем протестировать наши новые группы **netgroup** на клиентской машине **ldap-client**.

8.3 Настройка клиента NFS

Где работаем: **ldap-client**

Подключитесь к клиентской машине. **Используйте пользователя, чей домашний каталог не находится в /nfs/home!**

```
$  ssh pablo@ldap-client.example.com
```

В разделе 7 мы уже настроили эту машину в качестве клиента NFS с использованием **autofs**. Теперь посмотрим, можем ли мы автоматически монтировать каталог **/nfs/home** с новыми настройками сервера NFS. Первым делом изменим конфигурацию **netgroup** в **/etc/nsswitch.conf**, чтобы она выглядела так:

```
netgroup:     ldap
```

Проверим, есть ли у нас доступ к записям **netgroup** на сервере каталогов:

```
$  getent netgroup itdept
itdept                  (ldap-client.example.com,,)
```

Хорошо. Теперь удостоверимся, что **/nfs/home** не примонтирован. **ВНИМАНИЕ! Не выполняйте нижеследующую команду от имени пользователя, домашний каталог которого находится в /nfs/home!**

```
#  umount /nfs/home
```

Проверим, анонсирует ли сервер NFS ресурсы, доступные для группы **itdept**:

```
$  showmount -e nfs-srv.example.com
Export list for nfs-srv.example.com:
/export/install @itdept
/export/home    @itdept
```

Анонсирует. Следовательно, мы должны иметь возможность примонтировать их:

```
$ cd /nfs/home
$ df -h .
Файл.система                    Размер Использовано  Дост Использовано% Смонтировано в
nfs-srv.example.com:/export/home   3,6G         1,6G  1,8G           47% /nfs/home
```

Отлично! Работает. Но сработает ли автоматическое монтирование, если у нас нет доступа? Для этого понадобится отмонтировать каталоги **/nfs/home** и **/nfs/install** на клиенте:

```
#  umount /nfs/home
#  umount /nfs/install
```

Перейдём на сервер NFS и изменим настройки доступа к сетевым ресурсам.

Где работаем: **nfs-srv**

Отредактируем файл **/etc/exports**, изменив группу **netgroup**:

```
/export/home    @oracle(rw,no_subtree_check)
/export/install @oracle(rw,no_subtree_check)
```

Мы заменили группу **itdept**, в которую входит машина **ldap-client.example.com** на группу **oracle**, в которую эта машина не входит. Теперь машина **ldap-client.example.com** не сможет получить доступ к сетевым ресурсам. Но чтобы изменения вступили в силу, нужно перезагрузить сервис NFS:

```
#  service nfs-kernel-server reload
$  showmount -e
Export list for nfs-srv.example.com:
/export/home         @oracle
```

Вернёмся на клиентскую машину и попробуем получить доступ к ресурсам, анонсируемым **nfs-srv**. **Напоминаю, надо использовать учётную запись пользователя, домашний каталог которого не лежит в /nfs/home!**

Где работаем: **ldap-client**

Попробуем открыть каталог **/nfs/home**:

```
$ cd /nfs/home
-bash: cd: /nfs/home: Нет такого файла или каталога
```

Отлично! Это значит, что наши группы **netgroup** работают.

8.4 Добавление новых значений атрибута nisNetgroupTriple

Этот раздел справочный. Для дальнейших экспериментов он играет роль постольку-поскольку.

8.4.1 Проблема и два варианта решения

Где работаем: **ldap-srv**

Я столкнулся со следующей проблемой. При попытке добавить новое значение атрибута **nisNetgroupTriple** в

запись **cn=itdept,ou=netgroup,ou=services,dc=example,dc=com** появляется ошибка. Возьмём в качестве примера вот такой LDIF файл **8.4.1-ldap-client2-netgroup.ldif**:

```
dn: cn=itdept,ou=netgroup,ou=services,dc=example,dc=com
changetype: modify
add: nisNetgroupTriple
nisNetgroupTriple: (ldap-client2.example.com,,)
```

При попытке его загрузить в сервер каталогов мы увидим ошибку:

```
$  ldapadd -xZZWD cn=admin,dc=example,dc=com -f 8.4.1-ldap-client2-netgroup.ldif
Enter LDAP Password:
modifying entry "cn=itdept,ou=netgroup,ou=services,dc=example,dc=com"
ldap_modify: Inappropriate matching (18)
        additional info: modify/add: nisNetgroupTriple: no equality matching rule
```

После небольших поисков, стало понятно, что это не ошибка, а ограничение схемы данных **nis**. Два возможных пути в такой ситуации:

1. Длинный путь. Удалить ветку с определеним группы в контейнере **ou=netgroup,ou=services,dc=example,dc=com** целиком и создать её заново с новыми наборами **nisNetgroupTriple**.
2. Короткий путь. Создать и загрузить в сервер каталогов файл LDIF, переопределяющий все наборы **nisNetgroupTriple**, вроде такого:

```
dn: cn=itdept,ou=netgroup,ou=services,dc=example,dc=com
changetype: modify
replace: nisNetgroupTriple
nisNetgroupTriple: (ldap-client.example.com,,)
nisNetgroupTriple: (ldap-client2.example.com,,)
```

Несмотря на то, что данные способы работают, использовать их повседневно может быть накладно. Вариант — написать какой-нибудь скрипт, автоматизирующий этот процесс, но лучше прибегнуть к ещё одному способу.

8.4.2 Третий вариант решения

Где работаем: **ldap-srv**

Есть третий путь, который позволит нам добавлять/удалять наборы **nisNetgroupTriple**. Он требует модификации схемы данных **nis**. Однако тут мы вновь сталкиваемся с неудобством. Перед изменением схемы придётся удалить все записи, в которых есть атрибут **nisNetgroupTriple**, потому что мы собираемся изменить определение этого атрибута.

Вот **diff** оригинальной и модифицированной схемы с парой строк контекста (**-C 1**), чтобы было видно, где были произведены изменения:

```
#  diff -C 1 cn\=\{10\}nis.ldif.original cn\=\{10\}nis.ldif.modified
*** cn={10}nis.ldif.original 2012-05-17 17:26:18.479629651 -0400
--- cn={10}nis.ldif.modified 2012-05-17 17:28:46.289627851 -0400
***************
*** 33,35 ****
  olcAttributeTypes: {12}( 1.3.6.1.1.1.1.14 NAME 'nisNetgroupTriple' DESC 'Netgr
!  oup triple' SYNTAX 1.3.6.1.1.1.0.0 )
  olcAttributeTypes: {13}( 1.3.6.1.1.1.1.15 NAME 'ipServicePort' EQUALITY intege
--- 33,35 ----
  olcAttributeTypes: {12}( 1.3.6.1.1.1.1.14 NAME 'nisNetgroupTriple' DESC 'Netgr
!  oup triple' EQUALITY caseExactIA5Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
  olcAttributeTypes: {13}( 1.3.6.1.1.1.1.15 NAME 'ipServicePort' EQUALITY intege
```

Как мы можем видеть, изменение объекта **nisNetgroupTriple** довольно тривиальное.

Но произвести это изменение — задача не очень тривиальная.

Для начала удалим все дочерние записи у **ou=netgroup,ou=services,dc=example,dc=com**:

```
$  ldapdelete -xZZWD cn=admin,dc=example,dc=com  cn=oracle,ou=netgroup,ou=services,dc=example,dc=com
$  ldapdelete -xZZWD cn=admin,dc=example,dc=com  cn=itdept,ou=netgroup,ou=services,dc=example,dc=com
```

Посмотрим, какой порядковый номер у схемы данных **nis** в нашем каталоге:

```
$  ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b cn=schema,cn=config dn |grep nis
Enter LDAP Password:
dn: cn={10}nis,cn=schema,cn=config
```

Так, номер 10. Теперь посмотрим, какой порядковый номер у атрибута **nisNetgroupTriple** в схеме данных **nis**:

```
$  ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b cn=schema,cn=config | grep nisNetgroupTriple
Enter LDAP Password:
olcAttributeTypes: {12}( 1.3.6.1.1.1.1.14 NAME 'nisNetgroupTriple' DESC 'Netgr
  a netgroup' SUP top STRUCTURAL MUST cn MAY ( nisNetgroupTriple $ memberNisNe
```

Номер 12. Замечательно. Сформируем новый LDIF под названием **8.4.2-nis-schema-change.ldif** для изменения схемы. Если два последних номера у Вас отличаются — подставьте свои значения:

```
dn: cn={10}nis,cn=schema,cn=config
changetype: modify
delete: olcAttributeTypes
olcAttributeTypes: {12}( 1.3.6.1.1.1.1.14 NAME 'nisNetgroupTriple' DESC 'Netgr
 oup triple' SYNTAX 1.3.6.1.1.1.0.0 )
-
add: olcAttributeTypes
olcAttributeTypes: {12}( 1.3.6.1.1.1.1.14 NAME 'nisNetgroupTriple' DESC 'Netgr
 oup triple' EQUALITY caseExactIA5Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

Загрузим его в наш сервер каталогов:

```
$  ldapadd -xZZWD cn=admin,dc=example,dc=com  -f nis-schema-change.ldif
```

И проверим результат:

```
$ ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b cn={10}nis,cn=schema,cn=config olcAttributeTypes |grep
-A 1 nisNetgroupTriple
Enter LDAP Password:
olcAttributeTypes: {12}( 1.3.6.1.1.1.1.14 NAME 'nisNetgroupTriple' DESC 'Netgr
 oup triple' EQUALITY caseExactIA5Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

Отлично! Схема данных обновлена. Теперь можем добавлять новые записи в группы **netgroup** по одной без каких-либо проблем. Это особенно удобно при работе с каталогом из GUI какого-нибудь LDAP-браузера (пример — Apache Directory Studio). Если Вы пришли в этот раздел прямиком из начала раздела 8, то можете продолжить в 8.1. Если Вы уже проделывали все предыдущие пункты, то остаётся только вернуть на место группы **netgroup**. Следуя нашим примерам, вернуть их можно с помощью такого нехитрого файла LDIF (**8.4.2-netgroup-addentryonly.ldif**):

```
dn: cn=oracle,ou=netgroup,ou=services,dc=example,dc=com
objectClass: top
objectClass: nisNetgroup
cn: oracle
nisNetgroupTriple: (oracle.example.com,,)
description: All Oracle machines

dn: cn=itdept,ou=netgroup,ou=services,dc=example,dc=com
objectClass: top
objectClass: nisNetgroup
cn: itdept
nisNetgroupTriple: (ldap-client.example.com,,)
description: IT department machines
```

Загрузим его в наш сервер каталогов:

```
$  ldapadd -xZZWD cn=admin,dc=example,dc=com  -f netgroup-addentryonly.ldif
```

Следующий раздел опишет, как развернуть область (realm) Kerberos с использованием OpenLDAP в качестве хранилища принципалов (principal). Это будет очень весело! Потому что как только у нас есть область Kerberos, мы можем использовать её для безопасной аутентификации в ssh, для защиты соединений с OpenLDAP с помощью SASL GSSAPI, для защиты запросов NFS на автоматическое монтирование и самого доступа к сетевым ресурсам. И много для чего ещё!

9. Kerberos KDC с использованием OpenLDAP в качестве бэкэнда и аутентификацией SASL GSSAPI

В этом разделе мы научимся использовать OpenLDAP 2.4 в качестве хранилища принципалов (principals) Kerberos и разберем, как настраивать клиентские рабочие станции. Kerberos — это сетевой протокол, который работает на основе билетов (tickets) и позволяет передавать данные через незащищённые сети для безопасной идентификации и аутентификации. Его дизайн преимущественно опирается на клиент-серверную модель и позволяет произвести взаимную аутентификацию субъекта и объекта доступа. Сообщения протокола устойчивы к прослушиванию и атакам повтора. Kerberos работает на основе криптографии с симметричным ключом и требует наличия доверенной третьей стороны, а так же может применяться с использованием криптографии с открытым ключом на некоторых этапах процесса аутентификации.

В этом руководстве в роли сервера Kerberos будем использовать **`ldap-srv`**. Но ничто не мешает Вам использовать для этого отдельную машину.

9.1 Настройка сервера

Где работаем: **ldap-srv**

Включите NTP и убедитесь, что сервер и клиенты синхронизированы! Мы не будем описывать, как настраивать NTP. Вы можете найти множество примеров в сети, если потребуется.

Теперь, когда наш сервер OpenLDAP настроен, мы можем приступить к конфигурированию сервера Kerberos. В этом разделе мы опишем, как развернуть центр распределения ключей Kerberos (KDC, Key Distribution Center) для области (realm) EXAMPLE.COM. Начнём с установки необходимых пакетов. В процессе будет так же установлено несколько пакетов-зависимостей. Пакет **`wamerican`** создаст файл **`/usr/share/dict/words`**, используемый **`kadmind`**. Вместо него можно использовать любой другой словарь или несколько словарей. Чтобы посмотреть их список, посмотрите содержимое метапакета **`wordlist`**.

```
#  apt-get install krb5-kdc krb5-pkinit krb5-admin-server wamerican libsasl2-modules-gssapi-mit
```

В разделе 2.5 мы уже установили схему Kerberos для OpenLDAP сервера, но давайте лишний раз убедимся, что она на месте:

```
#  ldapsearch -QLLLY EXTERNAL -H ldapi:/// -b cn=schema,cn=config dn | grep -i kerberos
dn: cn={12}kerberos,cn=schema,cn=config
```

Схема на месте. В ней содержится достаточно много объектов. Чтобы посмотреть их все, используйте следующий запрос:

```
#  ldapsearch -QLLLY EXTERNAL -H ldapi:/// -b cn={12}kerberos,cn=schema,cn=config | grep NAME | cut -d' '
-f5 | sort
```

Эта команда должна вернуть список объектов. Это важно, потому что если новых атрибутов Kerberos LDAP нет, то **kdb5_ldap_util**(8) выдаст следующую ошибку:

```
kdb5_ldap_util: Kerberos Container create FAILED: No such object while creating realm 'EXAMPLE.COM'
```

Итак, у нас есть набор схемы данных и объекты. Но есть ли у нас контейнер для принципалов Kerberos?

```
#  ldapsearch -QLLLY EXTERNAL -H ldapi:/// -b ou=services,dc=example,dc=com dn | grep -i kerberos
```

Нет, нету. Создадим его, а заодно — пользователя и группу, которые будут использоваться Kerberos для взаимодействия с сервером OpenLDAP. Используем для этой цели вот такой LDIF-файл **9.1-kerberos.ldif**:

```
# Создадим контейнер для Kerberos
dn: cn=kerberos,ou=services,dc=example,dc=com
cn: kerberos
objectClass: top
objectClass: krbContainer

# Добавим группу krbadmin
dn: cn=krbadmin,ou=groups,dc=example,dc=com
objectClass: top
objectClass: posixGroup
cn: krbadmin
gidNumber: 800
description: Kerberos administrator's group.

# Добавим пользователя krbadmin
dn: cn=krbadmin,ou=users,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: posixAccount
objectClass: top
cn: krbadmin
givenName: Kerberos Administrator
mail: kerberos.admin@example.com
sn: krbadmin
uid: krbadmin
uidNumber: 800
gidNumber: 800
homeDirectory: /home/krbadmin
loginShell: /bin/false
displayname: Kerberos Administrator
```

Загрузим этот файл в базу данных нашего сервера каталогов:

```
$  ldapadd -xZZWD cn=admin,dc=example,dc=com -f ~/ldap/9.1-kerberos.ldif
Enter LDAP Password:
adding new entry "cn=kerberos,ou=services,dc=example,dc=com"
adding new entry "cn=krbadmin,ou=groups,dc=example,dc=com"
adding new entry "cn=krbadmin,ou=users,dc=example,dc=com"
```

Зададим пароль для пользователя **krbadmin**. Сохраните пароль в надежном месте (например, используя keepass или gpg).

```
$  ldappasswd -xZZWSD "cn=admin,dc=example,dc=com" "cn=krbadmin,ou=users,dc=example,dc=com"
```

Создадим конфигурационный файл Kerberos **/etc/krb5.conf**:

```
[logging]
 default = SYSLOG:INFO:LOCAL1
 kdc = SYSLOG:NOTICE:LOCAL1
 admin_server = SYSLOG:WARNING:LOCAL1

[libdefaults]
 default_realm = EXAMPLE.COM
 dns_lookup_realm = false
 dns_lookup_kdc = false
 ticket_lifetime = 24h
 renew_lifetime = 7d
 forwardable = true
```

```
[realms]
 EXAMPLE.COM = {
  kdc = ldap-srv.example.com
  admin_server = ldap-srv.example.com
  default_domain = example.com
  database_module = openldap_ldapconf
 }

[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM

[appdefaults]
 pam = {
  debug = false
  ticket_lifetime = 36000
  renew_lifetime = 36000
  forwardable = true
  krb4_convert = false
 }

[dbmodules]
 openldap_ldapconf = {
  db_library = kldap
  ldap_kerberos_container_dn = cn=kerberos,ou=services,dc=example,dc=com
  ldap_kdc_dn = cn=krbadmin,ou=users,dc=example,dc=com
   # этот объект должен иметь права на чтение контейнера области,
   # контейнера принципала и поддеревьев области
  ldap_kadmind_dn = cn=krbadmin,ou=users,dc=example,dc=com
   # этот объект должен иметь права на чтение контейнера области,
   # контейнера принципала и поддеревьев области
  ldap_service_password_file = /etc/krb5.d/stash.keyfile
  ldap_servers = ldapi:///
  ldap_conns_per_server = 5
 }
```

Теперь создадим список контроля доступа (ACL) администратора Kerberos. Не путайте этот ACL с ACL OpenLDAP. Это не одно и то же. Чуть ниже мы поработаем с ACL для OpenLDAP. Создадим файл **/etc/krb5kdc/kadm5.acl** с таким содержимым:

```
*/admin@EXAMPLE.COM *
```

Отредактируем **/etc/krb5kdc/kdc.conf**, конфигурационный файл службы аутентификации (AS, Authentication Service) и центра распределения ключей (KDC):

```
[kdcdefaults]
 kdc_ports = 88
 kdc_tcp_ports = 88

[realms]
 EXAMPLE.COM = {
  #master_key_type = aes256-cts
  acl_file = /etc/krb5kdc/kadm5.acl
  dict_file = /usr/share/dict/words
  admin_keytab = /etc/krb5kdc/kadm5.keytab
  supported_enctypes = aes256-cts:normal aes128-cts:normal des3-hmac-sha1:normal arcfour-hmac:normal des-
hmac-sha1:normal des-cbc-md5:normal des-cbc-crc:normal
 }
```

В отдельном терминале включите просмотр журнала сервера OpenLDAP. Так мы сможем увидеть сообщения, генерируемые командой **kdb5_ldap_util**(8):

```
#  tail -F /var/log/slapd.log
```

Создайте каталог-тайник для пароля. В файле **/etc/krb5.conf** он указан в переменной **ldap_service_password_file**:

```
#  mkdir /etc/krb5.d
#  chmod u=rwx,g=,o= /etc/krb5.d
```

Извлеките пароль пользователя **cn=krbadmin,ou=users,dc=example,dc=com**, используя **kdb5_ldap_util**(8):

```
#  kdb5_ldap_util -D "cn=admin,dc=example,dc=com" stashsrvpw -f /etc/krb5.d/stash.keyfile
cn=krbadmin,ou=users,dc=example,dc=com
```

Вам будет предложено ввести мастер-пароль базы данных. Очень важно, чтобы Вы НЕ ЗАБЫЛИ этот пароль!

В основном терминале выполните следующую команду для добавления записей Kerberos в базу данных OpenLDAP:

```
#  kdb5_ldap_util -D "cn=admin,dc=example,dc=com" create -subtrees
"cn=kerberos,ou=services,dc=example,dc=com" -r EXAMPLE.COM -s
```

Вышеуказанная команда создаст несколько записей в **cn=kerberos,ou=services,dc=example,dc=com**. Чтобы увидеть их, выполните запрос:

```
#  ldapsearch -QLLLY EXTERNAL -H ldapi:/// -b cn=kerberos,ou=services,dc=example,dc=com dn
```

Но может ли кто-нибудь кроме пользователя **cn=admin** увидеть нашу информацию Kerberos? Это было-бы не очень мудро. Поэтому давайте взглянём на ACL нашего сервера OpenLDAP:

```
#  ldapsearch -QLLLY EXTERNAL -H ldapi:/// -b olcDatabase={1}mdb,cn=config olcAccess
```

Что нам надо сделать, так это дать права доступа на чтение/запись администратору Kerberos (**cn=krbadmin,ou=users,dc=example,dc=com**) к поддереву **cn=kerberos,ou=services,dc=example,dc=com**. Никакой другой пользователь не должен иметь доступа к этой информации (за исключением администратора каталога).

Для этого сформируем LDIF-файл **9.1-kerberos.acl.ldif**:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
replace: olcAccess
olcAccess: {0}to *
  by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" manage
  by * break
olcAccess: {1}to attrs=userPassword,userPKCS12
  by self write
  by anonymous auth
olcAccess: {2}to attrs=shadowLastChange
  by self write
olcAccess: {3}to dn.subtree="cn=kerberos,ou=services,dc=example,dc=com"
  by dn.exact="cn=krbadmin,ou=users,dc=example,dc=com" manage
olcAccess: {4}to *
  by dn.base="cn=nssproxy,ou=users,dc=example,dc=com" read
  by self read
```

Загрузим данные в сервер каталогов:

```
#  ldapmodify -QY EXTERNAL -H ldapi:/// -f 9.1-kerberos.acl.ldif
```

Убедитесь, что новые ACL работают. Суперпользователь и пользователь **cn=admin,dc=example,dc=com** должны иметь доступ к поддереву **cn=kerberos,ou=services,dc=example,dc=com**. Пользователь **cn=nssproxy,ou=users,dc=example,dc=com** не сможет обнаружить само существование контейнера Kerberos, а **cn=krbadmin,ou=users,dc=example,dc=com** должен не только видеть контейнер, но и иметь для него права на чтение/запись. Мы так же должны убедиться, что обычные пользователи всё ещё могут использовать сервер OpenLDAP для аутентификации и что они могут менять себе пароли.

Этот запрос возвращает поддерево **cn=kerberos,ou=services,dc=example,dc=com**:

```
$  ldapsearch -xZZLLLWD cn=krbadmin,ou=users,dc=example,dc=com -b cn=kerberos,ou=services,dc=example,dc=com
dn
```

Следующий запрос должен завершиться с ошибкой « **No such object (32)** »:

```
$  ldapsearch -xZZLLLWD cn=nssproxy,ou=users,dc=example,dc=com -b cn=kerberos,ou=services,dc=example,dc=com
dn
```

Теперь с клиентской машины убедимся, что пользователь, учётная запись которого хранится на сервере каталогов, может менять свой пароль:

Где работаем: **ldap-client**

```
$  su - test.user
$  passwd
(current) LDAP Password:
Новый пароль :
Повторите ввод нового пароля :
passwd: password updated successfully
```

Где работаем: **ldap-srv**

Настало время настроить журналирование. Для начала добавим в конфигурацию **rsyslog** следующие строки (**/etc/rsyslog.conf**):

```
...

# Пишем журнал демона krb5-kdc в /var/log/krb5kdc.log:
if $programname == 'krb5kdc' then /var/log/krb5kdc.log
& ~

# Пишем журнал демона krb5-admin-server в /var/log/kadmind.log:
if $programname == 'kadmind' then /var/log/kadmind.log
& ~
```

Создадим файлы журналов и зададим для них права доступа:

```
#  touch /var/log/{krb5kdc,kadmind}.log
#  chmod 0640 /var/log/{krb5kdc,kadmind}.log
#  chown syslog:adm /var/log/{krb5kdc,kadmind}.log
```

Настроим **logrotate**. Создадим два конфигурационных файла.

Файл **/etc/logrotate.d/krb5kdc** для демона **krb5-kdc**:

```
/var/log/krb5kdc.log {
 daily
 missingok
 rotate 7
 compress
 delaycompress
 notifempty
}
```

Файл **/etc/logrotate.d/kadmind** для демона **krb5-admin-server**:

```
/var/log/kadmind.log {
 daily
 missingok
 rotate 7
 compress
 delaycompress
 notifempty
}
```

Перезапустим демон **rsyslog**, чтобы конфигурация вступила в силу. Добавим демоны **krb5-kdc** и **krb5-admin-server** в автоматический запуск, затем запустим их:

```
#  service rsyslog restart
#  update-rc.d krb5-kdc defaults
```

```
#  update-rc.d krb5-admin-server defaults
#  service krb5-kdc start
#  service krb5-admin-server start
```

Загляните в журнал **/var/log/krb5kdc.log**. Вы должны увидеть там подобную строку:

```
...
Dec  1 15:49:54 ldap-srv krb5kdc[1992]: commencing operation
```

Проверим, что наши демоны слушают необходимые порты. Порт 88 (**krb5kdc**), порты 464 и 749 (**kadmind**):

```
$  ss -tan | egrep ':88|:464|:749'
LISTEN     0      2                          *:749                      *:*
LISTEN     0      5                          *:464                      *:*
LISTEN     0      5                          *:88                       *:*
```

Поздравляю, теперь у Вас есть рабочий сервер аутентификации (AS) и сервер распределения ключей (KDC) MIT Kerberos 5!

Что дальше? В первую очередь нам необходимо создать принципал для локального сервера. Затем — принципал для пользователя **test.user**. Сделаем это так (разобъём вывод на части, чтобы добавить комментарии):

```
#  kadmin.local
Authenticating as principal root/admin@EXAMPLE.COM with password.
```

Здесь мы создаём принципал машины для текущего сервера (на котором залогинены). Далее жирный шрифт — вводимый нами текст:

```
kadmin.local:  addprinc -randkey host/ldap-srv.example.com@EXAMPLE.COM
WARNING: no policy specified for host/ldap-srv.example.com@EXAMPLE.COM; defaulting to no policy
Principal "host/ldap-srv.example.com@EXAMPLE.COM" created.
```

После создания принципала сервера мы можем добавить его в набор ключей Kerberos (**/etc/krb5.keytab**):

```
kadmin.local:  ktadd host/ldap-srv.example.com@EXAMPLE.COM
Entry for principal host/ldap-srv.example.com@EXAMPLE.COM with kvno 2, encryption type aes256-cts-hmac-
sha1-96 added to keytab FILE:/etc/krb5.keytab.
Entry for principal host/ldap-srv.example.com@EXAMPLE.COM with kvno 2, encryption type aes128-cts-hmac-
sha1-96 added to keytab FILE:/etc/krb5.keytab.
Entry for principal host/ldap-srv.example.com@EXAMPLE.COM with kvno 2, encryption type des3-cbc-sha1 added
to keytab FILE:/etc/krb5.keytab.
Entry for principal host/ldap-srv.example.com@EXAMPLE.COM with kvno 2, encryption type arcfour-hmac added
to keytab FILE:/etc/krb5.keytab.
Entry for principal host/ldap-srv.example.com@EXAMPLE.COM with kvno 2, encryption type des-hmac-sha1 added
to keytab FILE:/etc/krb5.keytab.
Entry for principal host/ldap-srv.example.com@EXAMPLE.COM with kvno 2, encryption type des-cbc-md5 added to
keytab FILE:/etc/krb5.keytab.
```

Следующим шагом создадим принципал для моего пользователя **pablo** и зададим для него пароль:

```
kadmin.local:  addprinc pablo@EXAMPLE.COM
WARNING: no policy specified for pablo@EXAMPLE.COM; defaulting to no policy
Enter password for principal "pablo@EXAMPLE.COM":
Re-enter password for principal "pablo@EXAMPLE.COM":
Principal "pablo@EXAMPLE.COM" created.
```

Теперь создадим принципал пользователя с правами администратора. Помните файл **/etc/krb5kdc/kadm5.acl**? Вот где он вступает в игру. С помощью этого файла пользователи с префиксом **/admin** имеют права администратора на доступ к области Kerberos. Это значит, что они могут создавать и удалять записи пользователей и политики доступа. Поэтому убедитесь, что Вы знаете этих пользователей и доверяете им!

```
kadmin.local:  addprinc pablo/admin@EXAMPLE.COM
WARNING: no policy specified for pablo/admin@EXAMPLE.COM; defaulting to no policy
Enter password for principal "pablo/admin@EXAMPLE.COM":
Re-enter password for principal "pablo/admin@EXAMPLE.COM":
Principal "pablo/admin@EXAMPLE.COM" created.
```

Посмотрим список текущих принципалов в области:

```
kadmin.local:  get_principals
K/M@EXAMPLE.COM
krbtgt/EXAMPLE.COM@EXAMPLE.COM
kadmin/admin@EXAMPLE.COM
kadmin/changepw@EXAMPLE.COM
kadmin/history@EXAMPLE.COM
kadmin/ldap-srv.example.com@EXAMPLE.COM
host/ldap-srv.example.com@EXAMPLE.COM
pablo@EXAMPLE.COM
pablo/admin@EXAMPLE.COM
```

А эта команда выведет подробную информацию о принципале **host/ldap-srv.example.com@EXAMPLE.COM**:

```
kadmin.local:  getprinc host/ldap-srv.example.com@EXAMPLE.COM
Principal: host/ldap-srv.example.com@EXAMPLE.COM
Expiration date: [never]
Last password change: Сб. дек. 06 09:21:17 MSK 2014
Password expiration date: [none]
Maximum ticket life: 1 day 00:00:00
Maximum renewable life: 0 days 00:00:00
Last modified: Сб. дек. 06 09:21:17 MSK 2014 (root/admin@EXAMPLE.COM)
Last successful authentication: [never]
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 6
Key: vno 2, aes256-cts-hmac-sha1-96, no salt
Key: vno 2, aes128-cts-hmac-sha1-96, no salt
Key: vno 2, des3-cbc-sha1, no salt
Key: vno 2, arcfour-hmac, no salt
Key: vno 2, des-hmac-sha1, no salt
Key: vno 2, des-cbc-md5, no salt
MKey: vno 1
Attributes:
Policy: [none]
kadmin.local:  exit
```

Обратите внимание, что был создан новый файл **/etc/krb5.keytab**. Вот почему мы запустили бинарник **kadmin.local** от имени суперпользователя. Иначе мы получили бы следующую ошибку:

```
$  kadmin.local
Authenticating as principal pablo/admin@EXAMPLE.COM with password.
kadmin.local: Error reading password from stash: Permission denied while initializing kadmin.local
interface
```

Только суперпользователь имеет доступ на чтение к файлу-тайнику (**/etc/krb5.d/stash.keyfile**).

Давайте посмотрим, что записано в **/etc/krb5.keytab**:

```
#  klist -ek /etc/krb5.keytab
Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
---- --------------------------------------------------------------------
   2 host/ldap-srv.example.com@EXAMPLE.COM (aes256-cts-hmac-sha1-96)
   2 host/ldap-srv.example.com@EXAMPLE.COM (aes128-cts-hmac-sha1-96)
   2 host/ldap-srv.example.com@EXAMPLE.COM (des3-cbc-sha1)
   2 host/ldap-srv.example.com@EXAMPLE.COM (arcfour-hmac)
   2 host/ldap-srv.example.com@EXAMPLE.COM (des-hmac-sha1)
   2 host/ldap-srv.example.com@EXAMPLE.COM (des-cbc-md5)
```

Как мы можем видеть, это список ключей шифрования машины **ldap-srv**. Неудивительно, что доступ предоставлен только суперпользователю!

У нас осталась ещё одна вещь на сервере, которую надо поправить. Это ошибка «**mdb_equality_candidates: (krbPrincipalName) not indexed**» в журнале **/var/log/slapd.log**. Создадим LDIF-файл **9.1-kerberos.indexes.ldif**, который внесёт поправки в индексы:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
```

```
add: olcDbIndex
olcDbIndex: krbPrincipalName eq
-
add: olcDbIndex
olcDbIndex: ou eq
```

Добавим эти индексы в сервер каталогов:

```
#  ldapadd -QY EXTERNAL -H ldapi:/// -f 9.1-kerberos.indexes.ldif
```

Отлично! Теперь у нас есть настроенный KDC!

9.2 Настройка клиента

С действующим KDC мы можем заставить клиентские машины и сервисы использовать его. Обозначим основные цели по настройке клиентов, чтобы интегрировать их в инфраструктуру Kerberos:

1. Аутентификация Kerberos для **sshd**.
2. Аутентификация OpenLDAP с помощью SASL GSSAPI.
3. Применение аутентификации SASL GSSAPI для AutoFS.

9.2.1 Аутентификация Kerberos для sshd

Где работаем: **ldap-client**

ВНИМАНИЕ! Клиенты Kerberos должны иметь возможность подключения к TCP портам KDC с номерами 88 и 749!

Установим необходимые пакеты:

```
#  apt-get install krb5-user libpam-krb5 libsasl2-modules-gssapi-mit
```

Во время установки в качестве области Kerberos можете задать EXAMPLE.COM, а в качестве серверов, обслуживающих область, **ldap-srv.example.com**.

Отредактируйте конфигурацию клиента в файле **/etc/krb5.conf** следующим образом:

```
[logging]
 default = SYSLOG:INFO:LOCAL1
 kdc = SYSLOG:NOTICE:LOCAL1
 admin_server = SYSLOG:WARNING:LOCAL1

[libdefaults]
 default_realm = EXAMPLE.COM
 dns_lookup_realm = false
 dns_lookup_kdc = false
 ticket_lifetime = 24h
 renew_lifetime = 7d
 forwardable = true

[realms]
 EXAMPLE.COM = {
  kdc = ldap-srv.example.com
  admin_server = ldap-srv.example.com
  default_domain = example.com
 }

[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

```
[appdefaults]
 pam = {
  debug = false
  ticket_lifetime = 36000
  renew_lifetime = 36000
  forwardable = true
  krb4_convert = false
 }
```

Создайте новый принципал машины для этого хоста. Мы запускаем команду **kadmin** от имени суперпользователя, чтобы можно было записать результирующий файл **/etc/krb5.keytab**. Иначе мы получим не очень понятную ошибку «**No such file or directory while adding key to keytab**». Мы так же должны использовать модификатор « **-p** », чтобы дать понять команде **kadmin**, от имени какого принципала мы хотим подключиться. Если мы его не зададим, то получим ошибку «**Client not found in Kerberos database while initializing kadmin interface**», потому что не создавали принципал **root/admin@EXAMPLE.COM**. Не создавайте этот принципал! Мы хотим знать, кто подключается с правами администратора (пользователи с префиксом **/admin**). Если создадим — не сможем различать администраторов между собой.

```
#  kadmin -p pablo/admin@EXAMPLE.COM
Authenticating as principal pablo/admin@EXAMPLE.COM with password.
Password for pablo/admin@EXAMPLE.COM:

kadmin:  addprinc -randkey host/ldap-client.example.com@EXAMPLE.COM
kadmin:  ktadd host/ldap-client.example.com@EXAMPLE.COM
kadmin:  exit
```

Эта команда создала файл **/etc/krb5.keytab**.

Теперь мы можем отредактировать **/etc/ssh/sshd_config** и включить аутентификацию Kerberos. Не забудьте добавить **test.group** в директиву **AllowGroups**. Иначе мы не сможем протестировать конфигурацию и увидим ошибку « **User test.user from ldap-srv.example.com not allowed because none of user's groups are listed in AllowGroups** » в файле **/var/log/auth.log**.

```
AddressFamily inet
Port 22
Protocol 2
AllowGroups sysadmin test.group
SyslogFacility AUTHPRIV
LoginGraceTime 30s
PermitRootLogin no
StrictModes yes
IgnoreRhosts yes
PermitEmptyPasswords no
PasswordAuthentication yes
AllowTcpForwarding no
X11Forwarding yes
PrintLastLog yes
ClientAliveInterval 120
ClientAliveCountMax 2
Banner /etc/issue

UsePAM yes
ChallengeResponseAuthentication yes
KerberosAuthentication yes
KerberosOrLocalPasswd yes
KerberosTicketCleanup yes
GSSAPIAuthentication yes
GSSAPICleanupCredentials yes
```

Перезапустим демон:

```
#  service ssh restart
```

Запустите на клиентской рабочей станции отображение файла журнала:

```
#  tail -F /var/log/auth.log
```

А пока вернёмся на сервер.

Где работаем: **ldap-srv**

Создадим принципал пользователя **test.user**:

```
$  kadmin -p pablo/admin@EXAMPLE.COM
Authenticating as principal pablo/admin@EXAMPLE.COM with password.
Password for pablo/admin@EXAMPLE.COM:
kadmin:  addprinc test.user@EXAMPLE.COM
WARNING: no policy specified for test.user@EXAMPLE.COM; defaulting to no policy
Enter password for principal "test.user@EXAMPLE.COM":
Re-enter password for principal "test.user@EXAMPLE.COM":
Principal "test.user@EXAMPLE.COM" created.
kadmin:  exit
```

Возьмём билет Kerberos у **test.user** и прикинемся этим пользователем. Сначалы мы уничтожим свои собственные билеты (если они есть), используя **kdestroy**, затем возьмём билет **test.user** с помощью **kinit**, а в итоге проверим, что он у нас есть с помощью **klist**:

```
$  kdestroy
$  kinit -p test.user@EXAMPLE.COM
Password for test.user@EXAMPLE.COM:
$  klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: test.user@EXAMPLE.COM

Valid starting       Expires              Service principal
06.12.2014 14:58:06  07.12.2014 14:58:06  krbtgt/EXAMPLE.COM@EXAMPLE.COM
```

Теперь попробуем авторизоваться на клиенте без пароля, используя этот билет:

```
$  ssh test.user@ldap-client.example.com
```

Где работаем: **ldap-client**

В открытом нами журнале на клиенте **/var/log/auth.log** мы должны увидеть что-то подобное:

```
Dec  6 15:00:55 ldap-client sshd[2170]: Authorized to test.user, krb5 principal test.user@EXAMPLE.COM
(krb5_kuserok)
Dec  6 15:00:55 ldap-client sshd[2170]: Accepted gssapi-with-mic for test.user from 192.168.122.150 port
45995 ssh2
Dec  6 15:00:55 ldap-client sshd[2170]: pam_unix(sshd:session): session opened for user test.user by
(uid=0)
```

Успех! Переходим к следующей цели.

9.2.2 Аутентификация OpenLDAP с помощью SASL GSSAPI

Где работаем: **ldap-srv**

Для настройки аутентификации SASL GSSAPI мы должны изменить конфигурацию сервера OpenLDAP таким образом, чтобы он знал о существовании нашей области Kerberos. После этого мы можем настроить клиентов.

Подключимся к KDC и создадим новый принципал. Мы по-прежнему запускаем **kadmin** от имени суперпользователя, потому что хотим создать новый набор ключей в файле **/etc/ldap/krb5.keytab**, чтобы наш демон **slapd** имел свой набор.

```
#  kadmin -p pablo/admin@EXAMPLE.COM
kadmin:  addprinc -randkey ldap/ldap-srv.example.com@EXAMPLE.COM
```

```
kadmin:  ktadd -k /etc/ldap/krb5.keytab ldap/ldap-srv.example.com@EXAMPLE.COM
```

Поменяем права доступа на этот файл , чтобы демон **slapd** смог его читать:

```
#  chown root:openldap /etc/ldap/krb5.keytab
#  chmod 640 /etc/ldap/krb5.keytab
```

Создадим LDIF-файл **9.2.2-sasl.ldif** с директивами SASL:

```
dn: cn=config
changetype: modify
add: olcSaslSecProps
olcSaslSecProps: noanonymous,noplain
-
add: olcSaslHost
olcSaslHost: ldap-srv.example.com
-
add: olcSaslRealm
olcSaslRealm: EXAMPLE.COM
```

И загрузим его в базу данных службы каталогов:

```
#  ldapadd -QY EXTERNAL -H ldapi:/// -f 9.2.2-sasl.ldif
```

На самом деле нам не нужно играться с **olcSaslSecProps** , но я оставил его, потому что пытался добавить ключевое слово **noactive**. Если при этом попытаться выполнить поиск, то получим следующую ошибку:

```
#  ldapsearch -LLLY EXTERNAL -H ldapi:/// -b cn=config -s base | grep -i sasl
SASL/EXTERNAL authentication started
ldap_sasl_interactive_bind_s: Authentication method not supported (7)
additional info: SASL(-4): no mechanism available: security flags do not match required
```

В данный момент такое поведение — не совсем то, что нам нужно. Может быть позже. Сейчас проверим, загрузил ли демон **slapd** нашу конфигурацию SASL:

```
#  ldapsearch -QLLLY EXTERNAL -H ldapi:/// -b cn=config -s base | grep -i sasl
olcSaslSecProps: noanonymous,noplain
olcSaslHost: ldap-srv.example.com
olcSaslRealm: EXAMPLE.COM
```

Хорошо! Теперь нам нужно добавить параметр **KRB5_KTNAME** в файл **/etc/default/slapd**:

```
SLAPD_CONF=
SLAPD_USER="openldap"
SLAPD_GROUP="openldap"
SLAPD_PIDFILE=
SLAPD_SERVICES="ldap:/// ldapi:///"
SLAPD_SENTINEL_FILE=/etc/ldap/noslapd
SLAPD_OPTIONS="-4"
export KRB5_KTNAME=/etc/ldap/krb5.keytab
```

Для того, чтобы изменения вступили в силу, нам нужно перезапустить демон **slapd**:

```
#  service slapd restart
```

Демон **slapd** снова запущен — мы можем переходить к настройке клиента.

Где работаем: **ldap-client**

Зайдём на клиентскую рабочую станцию по **ssh** от имени пользователя **pablo**:

```
$  ssh pablo/admin@ldap-client.example.com
```

Или даже так:

```
$  ssh pablo/admin@EXAMPLE.COM@ldap-client.example.com
```

Получим билет от KDC:

```
$  kdestroy
$  kinit -p pablo/admin
$  klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: pablo@EXAMPLE.COM

Valid starting       Expires              Service principal
07.12.2014 14:16:24  08.12.2014 14:16:24  krbtgt/EXAMPLE.COM@EXAMPLE.COM
```

Все настройки для доступа на сервер у нас уже внесены в файл **/etc/ldap/ldap.conf**. Проверим, можем ли мы сделать запрос к LDAP-серверу:

```
$  ldapwhoami
ldap_sasl_interactive_bind_s: No such object (32)
```

Такая ошибка может показаться странной. Нужно указать механизм GSSAPI для доступа с помощью SASL. Такой вариант должен сработать:

```
$  ldapwhoami -Y GSSAPI
SASL/GSSAPI authentication started
SASL username: pablo@EXAMPLE.COM
SASL SSF: 56
SASL data security layer installed.
dn:uid=pablo,cn=example.com,cn=gssapi,cn=auth
```

Супер! Добавленный модификатор мы тоже можем внести в **/etc/ldap/ldap.conf**, чтобы его не приходилось вбивать вручную каждый раз:

```
BASE  dc=example,dc=com
URI  ldap://ldap-srv.example.com
TLS_CACERT /etc/ssl/certs/rootca.crt
TLS_CRLFILE /etc/ssl/rootca.crl
TLS_REQCERT allow
TIMELIMIT 15
TIMEOUT  20
SASL_MECH GSSAPI
```

Вы заметили, что OpenLDAP преобразовал имя принципала Kerberos в формат DN (Distinguished Name)? У нас был принципал с таким именем:

pablo@EXAMPLE.COM

… который был преобразован **slapd** в это:

dn:uid=pablo,cn=example.com,cn=gssapi,cn=auth

Это значит нам нужно вернуться на сервер и задать новые правила доступа (ACL), если мы хотим, чтобы наш Kerberos-принципал AutoFS мог читать информацию для автоматического монтирования.

9.2.3 Применение аутентификации SASL GSSAPI для autofs

Вздохните поглубже, далее от Вас потребуется внимательность и терпение. ;)

Где работаем: **ldap-srv**

Вернёмся на наш сервер OpenLDAP и отредактируем правила доступа ACL. Но прежде чем мы продолжим, всегда неплохо проверить текущую конфигурацию. Теперь мы можем формировать запросы с использованием GSSAPI (без модификатора « **-x** »):

```
$  ldapsearch -xZZLLLWD cn=admin,dc=example,dc=com -b cn=config olcAccess
```

Запрос вернёт нам четыре DN с атрибутами **olcAccess**:

```
dn: olcDatabase={-1}frontend,cn=config
dn: olcDatabase={0}config,cn=config
dn: olcDatabase={1}mdb,cn=config
dn: olcDatabase={2}monitor,cn=config
```

Мы должны изменить ACL для **olcDatabase={0}config** и **olcDatabase={1}mdb**. Приготовьтесь, ACL станут немного сложней. Создадим LDIF-файл **9.2.3-gssapi.acl.ldif** и запишем в него:

```
dn: olcDatabase={0}config,cn=config
changetype: modify
delete: olcAccess
-
add: olcAccess
olcAccess: {0}to *
  by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth manage
  by dn.regex="uid=.*/admin,cn=example.com,cn=gssapi,cn=auth" manage

dn: olcDatabase={1}mdb,cn=config
changetype: modify
delete: olcAccess
-
add: olcAccess
olcAccess: {0}to *
  by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" manage
  by * break
-
add: olcAccess
# Позволять принципалам Kerberos с префиксом /admin
# менять любые пароли.
olcAccess: {1}to attrs=userPassword,userPKCS12
  by dn.regex="uid=.*/admin,cn=example.com,cn=gssapi,cn=auth" manage
  by self write
  by anonymous auth
-
add: olcAccess
# Позволять принципалам Kerberos с префиксом /admin
# записывать метку времени об изменении пароля.
olcAccess: {2}to attrs=shadowLastChange
  by dn.regex="uid=.*/admin,cn=example.com,cn=gssapi,cn=auth" manage
  by self write
-
add: olcAccess
# Позволять принципалам Kerberos с префиксом /admin просматривать
# контейнер с настройками Kerberos, но не менять его содержимое.
# Это правило заставит пользоваться инструментами Kerberos при администрировании области.
olcAccess: {3}to dn.subtree="cn=kerberos,ou=services,dc=example,dc=com"
  by dn.exact="cn=krbadmin,ou=users,dc=example,dc=com" write
  by dn.regex="uid=.*/admin,cn=example.com,cn=gssapi,cn=auth" read
-
# Позволять принципалам Kerberos с именем autofsclient/*
# видеть содержимое карт автоматического монтирования.
add: olcAccess
olcAccess: {4}to dn.subtree="ou=autofs,ou=services,dc=example,dc=com"
  by dn.regex="uid=.*/admin,cn=example.com,cn=gssapi,cn=auth" manage
  by dn.regex="uid=autofsclient/.*,cn=example.com,cn=gssapi,cn=auth" read
-
add: olcAccess
# Позволять принципалам Kerberos с префиксом /admin доступ на запись к остальному каталогу.
olcAccess: {5}to *
  by dn.regex="uid=.*/admin,cn=example.com,cn=gssapi,cn=auth" manage
  by dn.exact="cn=krbadmin,ou=users,dc=example,dc=com" write
  by self read
  by dn.base="cn=nssproxy,ou=users,dc=example,dc=com" read
```

Ух! Вы поняли все производимые изменения? Напоминаю, что **cn=admin,dc=example,dc=com** (**RootDN**) всегда имеет

полный доступ к **olcDatabase={1}mdb,cn=config**. То же касается и **olcDatabase={0}config**.Поэтому **cn=admin** явно не упомянут в ACL. Не волнуйтесь и внимательно прочитайте правила, одно за другим. У Вас всё получится ;)

Изменения масштабные, поэтому сделаем резервную копию на случай, если всё пойдёт наперекосяк:

```
#  tar zcvf ~/ldap/slapd.d.backup.tar.gz /etc/ldap/slapd.d
```

Загрузим новые ACL в конфигурацию OpenLDAP:

```
#  ldapmodify -QY EXTERNAL -H ldapi:/// -f 9.2.3-gssapi.acl.ldif
modifying entry "olcDatabase={-1}frontend,cn=config"
modifying entry "olcDatabase={0}config,cn=config"
modifying entry "olcDatabase={1}mdb,cn=config"
modifying entry "olcDatabase={2}monitor,cn=config"
```

Проверим, что **cn=admin,dc=example,dc=com** имеет права доступа к базе данных сервера OpenLDAP. Следующий запрос должен вернуть « **dn: cn=config** »:

```
$ ldapsearch -xLLLZWD cn=admin,dc=example,dc=com -b cn=config -s base dn
```

Суперпользователь с UID и GID равными нулю имеет аналогичный уровень доступа:

```
#  ldapsearch -LLLQY EXTERNAL -H ldapi:/// -b cn=config -s base dn
```

Такой же уровень доступа имеют принципалы с префиксом **/admin**. Получим билет принципала и выполним запрос от его имени с помощью SASL:

```
$  kdestroy
$  kinit -p pablo/admin@EXAMPLE.COM
$  ldapsearch -ZZLLLQb cn=config -s base dn
dn: cn=config
```

Проверьте, работают ли права для пользователя **cn=nssproxy**. Следующая команда должна вернуть все DN из DIT кроме записей, в которых присутствует **cn=kerberos**:

```
$  ldapsearch -xZZLLLWD cn=nssproxy,ou=users,dc=example,dc=com -b dc=example,dc=com dn
```

А могут ли обычные пользователи менять свой пароль? Проверим это с клиентской рабочей станции.

Где работаем: **ldap-client**

Процесс обновления пароля изменился. Если мы попытаемся запустить команду **passwd** от имени пользователя, которого нет в локальном файле **/etc/passwd**, пароль будет изменен с использованием механизмов Kerberos:

```
$ su - test.user
$ passwd
Current Kerberos password:
Новый пароль :
Повторите ввод нового пароля :
passwd: password updated successfully
```

Для спокойствия убедимся, что пароль поменялся в нужном месте. Сверим текущее время и время изменения пароля в принципале **test.user@EXAMPLE.COM**. Они должны различаться незначительно.

Время изменения пароля записывается в принципалах в атрибуте **krbLastPwdChange**. Запись содержит время по Гринвичу (GMT). Поэтому для начала посмотрим текущее время GMT на клиентской машине:

```
$ date -u
Сб. дек. 20 17:19:24 UTC 2014
```

Теперь получим билет администратора, чтобы заглянуть в атрибуты принципала **test.user@EXAMPLE.COM**.

```
$  kdestroy
$  kinit -p pablo/admin@EXAMPLE.COM
```

И наконец, отправим вот такой длиннющий запрос (ответ сервера — только последняя строчка) к серверу каталогов по атрибуту **krbLastPwdChange**:

```
$  ldapsearch -ZZLLLQb
krbPrincipalName=test.user@EXAMPLE.COM,cn=EXAMPLE.COM,cn=kerberos,ou=services,dc=example,dc=com -s base
krbLastPwdChange

dn: krbPrincipalName=test.user@EXAMPLE.COM,cn=EXAMPLE.COM,cn=kerberos,ou=servi ces,dc=example,dc=com
krbLastPwdChange: 20141220171526Z
```

Как мы можем видеть, пароль принципала **test.user@EXAMPLE.COM** был изменён четыре минуты назад (буква Z указывает на время по Гринвичу). Замечательно!

Теперь мы можем настроить клиентскую машину для использования механизма GSSAPI в работе с картами автомонтирования AutoFS. На этом этапе Вы должны быть авторизованы на клиентской машине от имени пользователя, **домашний каталог которого не лежит на сервере NFS!** Если это не так, авторизуйтесь повторно.

Перед тем как продолжить, проверьте, отмонтированы ли все каталоги NFS:

```
$  df -h
```

Проверьте общесистемную конфигурацию AutoFS в файле **/etc/defaults/autofs**:

```
TIMEOUT=300
BROWSE_MODE="no"
MOUNT_NFS_DEFAULT_PROTOCOL=4
LOGGING="debug"
OPTIONS="-d -v"
LDAP_URI="ldap://ldap-srv.example.com"
SEARCH_BASE="ou=autofs,ou=services,dc=example,dc=com"
MAP_OBJECT_CLASS="automountMap"
ENTRY_OBJECT_CLASS="automount"
MAP_ATTRIBUTE="ou"
ENTRY_ATTRIBUTE="cn"
VALUE_ATTRIBUTE="automountInformation"
USE_MISC_DEVICE="yes"
```

Заметьте, что атрибут **LOGGING** установлен в значение **debug**, а **OPTIONS** в « **-d -v** » Это для того, чтобы помочь нам с поиском проблем, если они возникнут. В рабочей конфигурации этот параметр должен быть изменён. Мы вернёмся к этому через пару минут.

Далее нам нужно создать принципал Kerberos для демона **autofs**. Мы так же должны добавить ключи этого нового принципала в набор ключей нашего клиента.

```
#  kadmin -p pablo/admin@EXAMPLE.COM
kadmin:  addprinc -randkey autofsclient/ldap-client.example.com@EXAMPLE.COM
kadmin:  ktadd autofsclient/ldap-client.example.com@EXAMPLE.COM
```

Как мы можем видеть, ключи **autofsclient** теперь являются частью набора ключей клиента:

```
#  klist -ek /etc/krb5.keytab
Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
---- --------------------------------------------------------------------------
   2 host/ldap-client.example.com@EXAMPLE.COM (aes256-cts-hmac-sha1-96)
   2 host/ldap-client.example.com@EXAMPLE.COM (aes128-cts-hmac-sha1-96)
   2 host/ldap-client.example.com@EXAMPLE.COM (des3-cbc-sha1)
   2 host/ldap-client.example.com@EXAMPLE.COM (arcfour-hmac)
   2 host/ldap-client.example.com@EXAMPLE.COM (des-hmac-sha1)
   2 host/ldap-client.example.com@EXAMPLE.COM (des-cbc-md5)
   2 autofsclient/ldap-client.example.com@EXAMPLE.COM (aes256-cts-hmac-sha1-96)
   2 autofsclient/ldap-client.example.com@EXAMPLE.COM (aes128-cts-hmac-sha1-96)
   2 autofsclient/ldap-client.example.com@EXAMPLE.COM (des3-cbc-sha1)
   2 autofsclient/ldap-client.example.com@EXAMPLE.COM (arcfour-hmac)
   2 autofsclient/ldap-client.example.com@EXAMPLE.COM (des-hmac-sha1)
```

```
   2 autofsclient/ldap-client.example.com@EXAMPLE.COM (des-cbc-md5)
```

Теперь мы можем изменить порядок аутентификации **autofs** в файле **/etc/autofs_ldap_auth.conf**, используя механизм GSSAPI и новый принципал:

```
<?xml version="1.0" ?>
 <autofs_ldap_sasl_conf
  usetls="yes"
  tlsrequired="yes"
  authrequired="yes"
  authtype="GSSAPI"
  clientprinc="autofsclient/ldap-client.example.com@EXAMPLE.COM"
/>
<!-- EOF -->
```

Перезапустим демон autofs:

```
#  service autofs restart
```

Проверим журнал **/var/log/syslog**, всё ли работает?

```
...
Dec 20 22:12:27 ldap-client automount[943]: Starting automounter version 5.0.7, master map auto.master
Dec 20 22:12:27 ldap-client automount[943]: using kernel protocol version 5.02
Dec 20 22:12:27 ldap-client automount[943]: lookup_nss_read_master: reading master ldap auto.master
Dec 20 22:12:28 ldap-client kernel: [    4.116950] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow
Control: RX
Dec 20 22:12:28 ldap-client kernel: [    4.118253] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
Dec 20 22:12:28 ldap-client nslcd[972]: version 0.8.13 starting
Dec 20 22:12:28 ldap-client nslcd[972]: accepting connections
Dec 20 22:12:28 ldap-client automount[943]: master_do_mount: mounting /nfs
Dec 20 22:12:28 ldap-client automount[943]: automount_path_to_fifo: fifo name /var/run/autofs.fifo-nfs
Dec 20 22:12:28 ldap-client automount[943]: lookup_nss_read_map: reading map ldap
ldap:ou=auto.nfs,ou=autofs,ou=services,dc=example,dc=com
Dec 20 22:12:28 ldap-client automount[943]: mounted indirect on /nfs with timeout 300, freq 75 seconds
Dec 20 22:12:28 ldap-client automount[943]: st_ready: st_ready(): state = 0 path /nfs
...
```

Отлично! Попробуем перейти в каталог **/nfs/home**:

```
$  cd /nfs/home
$  pwd
/nfs/home
```

Супер! Изменим атрибуты **LOGGING** и **OPTIONS** в файле **/etc/defaults/autofs** на повседневные:

```
...
LOGGING="none"
# OPTIONS="-d -v"
...
```

И перезапустим демон **autofs**:

```
#  service autofs restart
```

Вот и всё! На этом фокусы с Kerberos закончились. :)

В следующем разделе мы опишем, как создавать резервную копию сервера OpenLDAP и восстанавливаться из неё. Затем мы настроим репликацию. Принимая во внимание зависимость наших клиентов от сервера OpenLDAP, нам надо обеспечить некоторый уровень отказоустойчивости.

10. Резервное копирование и восстановление сервера OpenLDAP

Глава по обслуживанию в руководстве администратора OpenLDAP не слишком подробно описывает решаемые проблемы. Будем надеятся, что этот раздел поможет Вам.

10.1 Резервное копирование

Где работаем: **ldap-srv**

Что мы будем резервировать?

- Содержимое базы данных OpenLDAP;
- Конфигурацию сервера OpenLDAP;
- Общесистемную конфигурацию сервера.

База данных и конфигурация сервера OpenLDAP могут быть выгружены в LDIF-файлы с помощью **slapcat**. Запакуем их вместе в один tar-файл. Общесистемная конфигурация — это всё остальное. Например, набор ключей Kerberos (keytab) и файл **/etc/defaults/slapd**.

Создадим файл для скрипта резервного копирования, и каталоги, куда мы будем складировать данные:

```
#  mkdir -p /root/scripts /root/backup /var/log/backup
#  touch /root/scripts/backup.slapd.sh
#  chmod +x /root/scripts/backup.slapd.sh
```

И запишем в файл **backup.slapd.sh** несложный скрипт:

```
#!/bin/sh

# Копируем данные и конфигурацию OpenLDAP в сжатые LDIF-файлы.
# А так же резервируем весь каталог OpenLDAP и конфигурацию демона slapd.

umask 022

PATH="/bin:/usr/bin:/usr/local/bin:/sbin:/usr/sbin:/usr/local/sbin"
export PATH

DATE=`date +%Y%m%d`

BACKUP_DIR="/root/backup/slapd"
BACKUP_DATA_FILENAME="slapd.data.${DATE}.ldif"
BACKUP_CONFIG_FILENAME="slapd.config.${DATE}.ldif"
BACKUP_TAR_FILENAME="slapd.directory.${DATE}.tar.gz"

CA_TLS_CERT="/etc/ssl/certs/rootca.crt"

DIT_CONFIG="cn=config"
DIT_SUFFIX="dc=example,dc=com"

SLAPD_CONFIG_FILENAME="/etc/default/slapd"
SLAPD_DIR="/etc/ldap"
SLAPD_LOG_ROTATION="/etc/logrotate.d/slapd"
SLAPD_TLS_CERT="/etc/ldap/ssl/ldap-srv.example.com.crt"
SLAPD_TLS_KEY="/etc/ldap/ssl/ldap-srv.example.com.key"
SLAPCAT_OPTIONS="-F /etc/ldap/slapd.d"

LOGFILE="/var/log/backup/slapd.log"
KEEP="30"

# Убедимся, что скрипт запущен от имени суперпользователя
if [ `id -u` -ne "0" ]; then
        echo "ERROR: only root can run this script." | tee -a ${LOGFILE}
```

```
	exit 1
fi

# Проверим, есть ли у нас файл для журнала
if [ ! -f ${LOGFILE} ]; then
	touch ${LOGFILE}

	if [ "$?" -ne "0" ]; then
		echo "ERROR: could not create the log file."
		exit 1
	fi
fi

# Убедимся, что у нас есть каталог для резервных копий
if [ ! -d ${BACKUP_DIR} ]; then
	mkdir -p ${BACKUP_DIR}

	if [ "$?" -ne "0" ]; then
		echo "ERROR: could not create the backup directory." | tee -a ${LOGFILE}
		exit 1
	fi
fi

# Убедимся, что в нашем каталоге с резервными копиями не свалено слишком много файлов
# и удалим все резервные копии кроме последних ${KEEP} копий.
FILES=`find ${BACKUP_DIR} -type f -name "slapd.*" -print | wc -l`

if [ "${FILES}" -gt "${KEEP}" ]; then
	OVER=`echo ${FILES}-${KEEP} | bc`
	RMFILES=`find ${BACKUP_DIR} -type f -name "slapd.*" -print | sort -r | tail -${OVER}`
	echo "NOTE: removing ${RMFILES} from the backup directory." >> ${LOGFILE}
	rm ${RMFILES}
fi

# Создаём резервную копию данных из DIT
slapcat ${SLAPCAT_OPTIONS} -b ${DIT_SUFFIX} -l ${BACKUP_DIR}/${BACKUP_DATA_FILENAME} >/dev/null 2>&1

if [ "$?" -eq "0" ]; then
	gzip -f ${BACKUP_DIR}/${BACKUP_DATA_FILENAME} 2>&1 >> ${LOGFILE}

	if [ "$?" -ne "0" ] ; then
		echo "ERROR: dump file compression problem." | tee -a ${LOGFILE}
		exit 1
	fi
else
	echo "ERROR: problem running slapcat(8C) for the DIT data backup." | tee -a ${LOGFILE}
	rm ${BACKUP_DIR}/${BACKUP_DATA_FILENAME}
	exit 1
fi

# Сохраняем конфигурацию DIT в виде LDIF-файла
slapcat ${SLAPCAT_OPTIONS} -b ${DIT_CONFIG} -l ${BACKUP_DIR}/${BACKUP_CONFIG_FILENAME} >/dev/null 2>&1

if [ "$?" -eq "0" ]; then
	gzip -f ${BACKUP_DIR}/${BACKUP_CONFIG_FILENAME} 2>&1 >> ${LOGFILE}

	if [ "$?" -ne "0" ] ; then
		echo "ERROR: dump file compression problem." | tee -a ${LOGFILE}
		exit 1
	fi
else
	echo "ERROR: problem running slapcat(8C) for the DIT config backup." | tee -a ${LOGFILE}
	rm ${BACKUP_DIR}/${BACKUP_CONFIG_FILENAME}
	exit 1
fi

# Делаем резервную копию файлов каталога и конфигурации
BACKUP_FILES_LIST="${CA_TLS_CERT} ${SLAPD_CONFIG_FILENAME} ${SLAPD_DIR} ${SLAPD_LOG_ROTATION} $
{SLAPD_TLS_CERT} ${SLAPD_TLS_KEY}"

tar zcf ${BACKUP_DIR}/${BACKUP_TAR_FILENAME} ${BACKUP_FILES_LIST} >/dev/null 2>&1

if [ "$?" -ne "0" ]; then
	echo "ERROR: problem running config directory tar." | tee -a ${LOGFILE}
	rm ${BACKUP_DIR}/${BACKUP_TAR_FILENAME}
	exit 1
fi
```

Создадим задание планировщика **cron** для нашего скрипта:

```
#  crontab -e
```

Откроется конфигурация планировщика для суперпользователя. Допишем в неё:

```
# Резервное копирование для OpenLDAP
00     22     *      *      *      /root/scripts/backup.slapd.sh
```

По этому правилу резервное копирование запускается каждый день в 22:00. Запустите скрипт вручную или дождитесь и проверьте, что всё работает.

ВНИМАНИЕ! Копируйте файлы на другой сервер! Лучше всего монтировать сетевые ресурсы в каталоги резервного копирования (в нашем случае — в **/root/backup**). Тогда все данные сразу будут заливаться на удалённый сервер.

10.2 Восстановление

В случае выхода из строя сервера OpenLDAP у нас есть всё, чтобы его воскресить. Сценарий восстановления зависит от типа решаемой проблемы:

1. Полная потеря сервера;
2. Проблемы с ACL;
3. Повреждение данных (или человеческий фактор);
4. Исчерпано свободное место в файловой системе.

10.2.1 Полная потеря сервера

Где работаем: **ldap-srv**

Если сервер который крутит OpenLDAP сломается, то первым делом нам нужно будет найти другой или починить этот. Затем потребуется установить на него Ubuntu 14.04. Потом скопируйте на него файлы из резервной копии и выполните следующие команды:

```
#  apt-get install -y slapd ldap-utils krb5-kdc-ldap krb5-pkinit krb5-admin-server libnss-ldapd libpam-
ldapd wamerican sudo-ldap
#  mv /etc/ldap /etc/ldap.install
#  cd / && tar zxvf /tmp/slapd.directory.20141215.tar.gz
#  update-rc.d slapd defaults
#  service slapd start
```

Это поможет Вам начать работать. Конечно, затем нужно будет перенастроить **rsyslog**. Смотрите раздел 2.3.

10.2.2 Проблемы с ACL

Где работаем: **ldap-srv**

Чаще всего такое случается при настройке сервера. При изменении ACL Вы можете закрыть самим себе любой доступ к нему. Поправить ситуацию не сложно:

```
#  service nslcd stop
#  service nscd stop
#  service slapd stop
#  mv /etc/ldap /etc/ldap.broken
#  cd / && tar zxvf /root/backup/slapd/slapd.directory.20141215.tar.gz
#  service slapd start
#  service nscd start
#  service nslcd start
```

10.2.3 Повреждение данных (или человеческий фактор)

Где работаем: **ldap-srv**

Чтож, иногда и администраторы совершают ошибки. В обычной ситуации достаточно просто запустить LDAP-браузер и вернуть всё как было. Но допустим один администратор в пятницу сделал изменение и уехал заниматься альпинизмом. Другой администратор не имел ни малейшего понятия об изменениях и в субботу после обеда мы имеем сломанную конфигурацию. В таких ситуациях у среднестатистического администратора нет терпения для отладки. Поэтому самый простой (и самый быстрый!) способ исправления ошибки — **остановить демон `slapd`** и вернуть данные из резервной копии, которая была сделана в четверг, с помощью **`slapadd`**.

```
#  service nslcd stop
#  service nscd stop
#  service slapd stop
#  mv /var/lib/ldap /var/lib/ldap.broken
#  mkdir /var/lib/ldap
#  zcat /root/backup/slapd/slapd.data.20141212.ldif.gz > /tmp/slapd.data.ldif
#  slapadd -v < /tmp/slapd.data.ldif
#  chown -R openldap:openldap /var/lib/ldap
#  service slapd start
#  service nscd start
#  service nslcd start
```

Это вернёт данные из любого файла, в который мы сохраняли наши резервные копии.

10.2.4 Исчерпано свободное место в файловой системе

Где работаем: **ldap-srv**

Вы ведь храните каталог **`/var`** в отдельной файловой системе (CCE-14777-7)? Иногда в нём кончается свободное место. Это может случиться как от разрастания базы данных в **`/var/lib/ldap`**, так и от слишком активного наполнения каталога **`/var/log`** журналами событий (подсказка: храните базу LDAP в отдельной ФС). Отсутствие свободного места будет генерировать ошибки в журналах. Используемый нами механизм манипуляции данными MDB не требует каких-либо особых процедур для восстановления из такого состояния. После вполне очевидного решения проблемы увеличения свободного места (**`lvresize`** / монтирование новой ФС / **`btrfs filesystem resize`** …) нужно просто перезапустить демоны:

```
#  service slapd restart
#  service nscd restart
#  service nslcd restart
```

Вот и всё! Если пропали какие-то данные, просто восстановите их из резервной копии как описано в предыдущем разделе.

11. Репликация сервера OpenLDAP

Для организации репликации нашего DIT понадобится развернуть второй сервер OpenLDAP. Новую виртуальную машину назовём **ldap-srv-repl**. В соответствии с терминологией Syncrepl, мы будем работать со следующими сущностями:

- поставщик (provider): **ldap-srv.example.com** (мастер-сервер);
- потребитель (consumer): **ldap-srv-repl.example.com** (сервер-реплика).

Первая задача — прочитать раздел о репликации в руководстве администратора OpenLDAP. Отдельный интерес представляет следующий отрывок:

> Syncrepl поддерживает синхронизацию как на основе запросов, так и на основе посылок. В его базовом режиме синхронизации **refreshOnly** (только обновление), поставщик использует синхронизацию на основе запросов, не требующей отслеживания серверов-потребителей и хранения истории операций. Информация, которая нужна поставщику для обработки периодических запросов на проверку содержимого каталога, находится в синхронизационных куки самих запросов. Для оптимизации синхронизации на основе запросов, Syncrepl использует фазу наличия (present phase) и фазу удаления (delete phase) протокола LDAP Sync, вместо того, чтобы возвращаться к частым полным перезагрузкам содержимого каталога. Для дальнейшей оптимизации синхронизации на основе запросов поставщик может вести журнал в рамках сессии в качестве хранилища истории операций. В режиме синхронизации **refreshAndPersist** (обновление и непрерывность) поставщик использует синхронизацию на основе посылок. Поставщик отслеживает серверы-потребители, запросившие непрерывно-действующий поиск, и посылает им необходимые обновления по мере изменения реплицируемого содержимого каталога поставщика.

Репликация реализуется через наложения OpenLDAP. Ознакомьтесь с инструкцией **slapo-syncprov** в документации **man**. В ней представлена информация о наложении поставщика репликации. С её помощью мы настроим репликацию в режиме **refreshAndPersist** с использованием схемы дельта-syncrepl. Обратите внимание, что говорит официальная документация:

> Как видите, в использовании Syncrepl и **slapd-ldap**(8) Вы можете позволить разгуляться своему воображению, адаптируя технологии репликации к своей конкретной сетевой топологии.

11.1 Настройка поставщика

Настройка дельта-syncrepl предполагает работу и с поставщиком (мастер-сервер), и с потребителем (сервер-реплика). Для начала займёмся поставщиком. У нас это **ldap-srv.example.com**.

План действий по настройке поставщика:

1. Настройка **cn=module**;
2. Настройка базы данных **accesslog** для хранения журнала доступа (**cn=accesslog**);
3. Определение наложения **syncprov** поверх базы данных **accesslog**;
4. Определение наложений поставщика **syncprov** и **accesslog** поверх основной базы данных (**dc=example,dc=com**);
5. Новый пользователь для репликации (аутентификация и выгрузка данных);
6. Ограничения и ACL пользователя для репликации.

11.1.1 Настройка cn=module

Где работаем: **ldap-srv**

Чтобы подключить динамические модули для наложений **accesslog** и **syncprov** не помешает проверить, как подключены уже имеющиеся модули. Мы делали это в разделе 2.4. Воспользуемся настроенным Kerberos и механизмом аутентификации SASL GSSAPI, чтобы упростить наши запросы. Заметьте, что Вы всегда можете использовать учётную запись RootDN (**cn=admin,dc=example,dc=com**), но это будет длиннее.

```
$  cd ~/ldap
$  kdestroy
$  kinit -p pablo/admin
$  ldapsearch -ZZQLLLb cn=config dn |grep module
dn: cn=module{0},cn=config
```

Как мы видим, информация о подключенных модулях содержится в записи **cn=module{0},cn=config**. Посмотрим, какие атрибуты в ней есть:

```
$  ldapsearch -ZZQLLLb cn=module{0},cn=config
dn: cn=module{0},cn=config
objectClass: olcModuleList
cn: module{0}
olcModulePath: /usr/lib/ldap
olcModuleLoad: {0}back_mdb.la
olcModuleLoad: {1}back_monitor.la
```

Два динамических модуля у нас уже подключены. Лежат ли в одном каталоге с ними искомые модули наложений?

```
$  ls /usr/lib/ldap/ |egrep '(accesslog.la|syncprov.la)'
accesslog.la
syncprov.la
```

Вот этот короткий LDIF-файл **11.1.1-new-modules.ldif** поможет нам их подключить:

```
dn: cn=module{0},cn=config
changetype: modify
add: olcModuleLoad
olcModuleLoad: {2}accesslog.la
-
add: olcModuleLoad
olcModuleLoad: {3}syncprov.la
```

Загружаем LDIF:

```
$  ldapmodify -ZZQf 11.1.1-new-modules.ldif
```

Проверяем результат:

```
$  ldapsearch -ZZQLLLb cn=module{0},cn=config
dn: cn=module{0},cn=config
objectClass: olcModuleList
cn: module{0}
olcModulePath: /usr/lib/ldap
olcModuleLoad: {0}back_mdb.la
olcModuleLoad: {1}back_monitor.la
olcModuleLoad: {2}accesslog.la
olcModuleLoad: {3}syncprov.la
```

11.1.2 Настройка базы данных accesslog для хранения журнала доступа

Где работаем: **ldap-srv**

После подключения динамического модуля наложения **accesslog** мы должны создать базу данных для хранения журнала доступа. Очевидно, мы можем сделать это с помощью ещё одного LDIF-файла (**11.1.2-accesslog.ldif**):

```
dn: olcDatabase=mdb,cn=config
changetype: add
objectClass: olcDatabaseConfig
objectClass: olcMdbConfig
olcDatabase: mdb
olcDbDirectory: /var/lib/ldap/accesslog
olcSuffix: cn=accesslog
olcRootDN: cn=admin,dc=example,dc=com
olcDbIndex: default eq
olcDbIndex: entryCSN,objectClass,reqEnd,reqResult,reqStart
```

Создадим каталог для этой базы данных:

```
#  mkdir -p /var/lib/ldap/accesslog
#  chown -R openldap:openldap /var/lib/ldap
```

И загрузим конфигурацию LDIF в наш сервер каталогов:

```
$  ldapadd -ZZQf 11.1.2-accesslog.ldif
adding new entry "olcDatabase=mdb,cn=config"
```

Посмотрим, какой порядковый номер у новой базы данных:

```
$  ldapsearch -ZZQLLLb cn=config dn |grep mdb
dn: olcDatabase={1}mdb,cn=config
dn: olcDatabase={3}mdb,cn=config
```

Первая строка вывода — наше основное DIT. Значит вторая — только что добавленная запись. Проверим только что загруженую конфигурацию:

```
$  ldapsearch -ZZQLLLb olcDatabase={3}mdb,cn=config
```

11.1.3 Определение наложения syncprov поверх базы данных accesslog

Где работаем: **ldap-srv**

Определим наложение **syncprov** для новой базы данных с помощью LDIF-файла **11.1.3-overlay.accesslog.ldif**:

```
dn: olcOverlay=syncprov,olcDatabase={3}mdb,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov
olcSpNoPresent: TRUE
olcSpReloadHint: TRUE
```

Так как мы будем использовать репликацию на основе журнала доступа, отключаем использование фазы наличия с помощью атрибута **olcSpNoPresent**. Атрибут **olcSpReloadHint** указывает, что наложение должно соблюдать флаг reloadHint в Sync Control. Этот флаг используется потребителем, запрашивающим операцию репликации, для указания того, что он хочет принудительно выполнить загрузку полного DIT независимо от всех остальных параметров и значений, таких как Sync Cookie.

Загрузим конфигурацию:

```
$  ldapadd -ZZQf 11.1.3-overlay.accesslog.ldif
adding new entry "olcOverlay=syncprov,olcDatabase={3}mdb,cn=config"
```

В базе данных **olcDatabase={3}mdb,cn=config** должна появиться новая DN-запись:

```
$  ldapsearch -ZZQLLLb olcDatabase={3}mdb,cn=config
dn: olcDatabase={3}mdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcMdbConfig
olcDatabase: {3}mdb
olcDbDirectory: /var/lib/ldap/accesslog
olcSuffix: cn=accesslog
olcRootDN: cn=admin,dc=example,dc=com
olcDbIndex: default eq
olcDbIndex: entryCSN,objectClass,reqEnd,reqResult,reqStart

dn: olcOverlay={0}syncprov,olcDatabase={3}mdb,cn=config
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: {0}syncprov
olcSpNoPresent: TRUE
olcSpReloadHint: TRUE
```

11.1.4 Определение наложений поставщика syncprov и accesslog поверх основной базы данных

Где работаем: **ldap-srv**

Новый LDIF-файл **11.1.4-overlay.primary.ldif** преследует три цели:

1. Установка новых индексов в основной базе данных;
2. Определение наложения syncprov поверх основной базы данных;
3. Определение наложения accesslog поверх основной базы данных.

Запишем в **11.1.4-overlay.primary.ldif**:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: entryCSN eq
-
add: olcDbIndex
olcDbIndex: entryUUID eq

# Добавить наложение syncprov поверх базы данных dc=example,dc=com.
dn: olcOverlay=syncprov,olcDatabase={1}mdb,cn=config
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov
olcSpCheckPoint: 500 15

# Добавить наложение accesslog поверх базы данных dc=example,dc=com.
# Ежедневно сканировать журнал доступа к базе данных и очищать записи старше 7 дней.
dn: olcOverlay=accesslog,olcDatabase={1}mdb,cn=config
objectClass: olcOverlayConfig
objectClass: olcAccessLogConfig
olcOverlay: accesslog
olcAccessLogDB: cn=accesslog
olcAccessLogOps: writes
olcAccessLogPurge: 7+00:00 1+00:00
olcAccessLogSuccess: TRUE
```

Загрузим конфигурацию:

```
$  ldapadd -ZZQf 11.1.4-overlay.primary.ldif
modifying entry "olcDatabase={1}mdb,cn=config"
adding new entry "olcOverlay=syncprov,olcDatabase={1}mdb,cn=config"
adding new entry "olcOverlay=accesslog,olcDatabase={1}mdb,cn=config"
```

Убедимся, что новые наложения на месте:

```
$  ldapsearch -ZZQLLLb olcDatabase={1}mdb,cn=config dn
dn: olcDatabase={1}mdb,cn=config
dn: olcOverlay={0}syncprov,olcDatabase={1}mdb,cn=config
dn: olcOverlay={1}accesslog,olcDatabase={1}mdb,cn=config
```

11.1.5 Новый пользователь для репликации

Где работаем: **ldap-srv**

Для функций репликации нам нужна новая учётная запись пользователя. Она будет использоваться для аутентификации и выгрузки данных сервером-репликой. Ни больше, ни меньше. Очередной LDIF-файл **`11.1.5-replication.ldif`** поможет нам добавить эту запись:

```
dn: cn=replication,dc=example,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: replication
description: OpenLDAP Replication User
userPassword: {SSHA}6yudY3jO+yyTyvov9LhIl1j9S9RZt7Dx
```

Напоминаю, следующая команда генерирует посоленый хэш вводимого пароля. Замените в вышеприведённом файле значение атрибута **`userPassword`** на результат её работы:

```
$  slappasswd -h '{SSHA}'
```

Загрузим конфигурацию:

```
$  ldapadd -ZZQf 11.1.5-replication.ldif
adding new entry "cn=replication,dc=example,dc=com"
```

Обратите внимание, в нашем примере пользователь для репликации называется просто « **`replication`** ». Если Вы планируете иметь несколько серверов для репликации, то в идеальном случае нужно создать отдельного пользователя с уникальным именем для каждого из них. Благодаря этому Вы сможете отслеживать, какой сервер и когда производил репликацию. Вы так же можете задать, какая часть DIT будет реплицироваться каждой машиной, но это уже другая история.

11.1.6 Ограничения и ACL пользователя для репликации

Где работаем: **ldap-srv**

Осталось только задать новому пользователю ограничения и прописать для него ACL. Проделаем это в два шага, начиная с ограничений. Создадим LDIF-файл **`11.1.6-limits.ldif`** и запишем в него:

```
# Добавить ограничения для базы данных cn=accesslog
dn: olcDatabase={3}mdb,cn=config
changetype: modify
add: olcLimits
olcLimits: dn.exact="cn=replication,dc=example,dc=com" time.soft=unlimited time.hard=unlimited
size.soft=unlimited size.hard=unlimited
-
add: olcAccess
olcAccess: to *
  by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth read
  by dn.regex="uid=.*/admin,cn=example.com,cn=gssapi,cn=auth" read
  by dn.exact="cn=admin,dc=example,dc=com" read
  by dn.exact="cn=replication,dc=example,dc=com" read
```

```
# Добавить ограничения для базы данных dc=example,dc=com
dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcLimits
olcLimits: dn.exact="cn=replication,dc=example,dc=com" time.soft=unlimited time.hard=unlimited
size.soft=unlimited size.hard=unlimited
```

Загрузим конфигурацию:

```
$  ldapadd -ZZQf 11.1.6-limits.ldif
modifying entry "olcDatabase={3}mdb,cn=config"
modifying entry "olcDatabase={1}mdb,cn=config"
```

Следующий шаг — предоставить новому пользователю доступ на чтение в основной базе данных. Очень хорошая идея — дважды проверить текущие ACL перед подобной операцией:

```
$  ldapsearch -ZZQLLLb cn=config olcAccess
```

Теперь можем создать ещё один LDIF-файл, который обновит наши ACL. Здесь идея в том, чтобы иметь одинаковый набор ACL как у поставщика, так и у потребителя. За исключением того, что у потребителя нам не нужны ACL для пользователя « **replication** » (который будет описан далее в атрибуте **olcSyncRepl**). Сейчас же мы должны просто добавить его в ACL с номером 0. Запишем в LDIF-файл **11.1.6-provider.acl.ldif**:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
delete: olcAccess
olcAccess: {0}
-
add: olcAccess
olcAccess: {0}to *
  by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth manage
  by dn.exact="cn=replication,dc=example,dc=com" read
  by * break
```

Загрузим изменение ACL:

```
$  ldapadd -ZZQf 11.1.6-provider.acl.ldif
modifying entry "olcDatabase={1}mdb,cn=config"
```

Так мы можем проверить результат:

```
$  ldapsearch -ZZQLLLb olcDatabase={1}mdb,cn=config olcAccess
```

Нам надо убедиться, что пользователь **cn=replication,dc=example,dc=com** имеет доступ ко всему DIT:

```
$  ldapsearch -xZZLLLWD cn=replication,dc=example,dc=com -H ldap://ldap-srv.example.com
```

Наличие доступа ко всему DIT — обязательно. Не идите дальше, пока этого не добъётесь. Когда всё получилось, мы можем перейти к настройке сервера-реплики (потребителя).

11.2 Настройка потребителя

11.2.1 Создание сервера-реплики

Развернём новый сервер по аналогии с разделом 1.

Где работаем: **dns-srv**

Для начала обновим информацию на DNS сервере (файл **/etc/hosts**):

```
127.0.0.1            localhost
192.168.122.140      dns-srv.example.com

192.168.122.150      ldap-srv.example.com
192.168.122.151      ldap-client.example.com
192.168.122.154      nfs-srv.example.com
192.168.122.160      ldap-srv-repl.example.com
```

Теперь установим Ubuntu 14.04 server на наш сервер-реплику и перейдём к его настройке.

Где работаем: **ldap-srv-repl**

Настроим сеть в файле **/etc/network/interfaces**:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.168.122.160
        netmask 255.255.255.0
        gateway 192.168.122.1
        dns-nameservers 192.168.122.140
```

Настроим имя сервера-реплики. В Ubuntu 14.04 останов или перезапуск сервиса **networking** не поддерживается, поэтому придётся сделать так:

```
#  echo 'ldap-srv-repl.example.com' > /etc/hostname
#  reboot
```

Проверим разрешение имени будущего сервера-реплики LDAP:

```
$  dig ldap-srv-repl.example.com +short
192.168.122.160
```

Проверим обратное разрешение имени:

```
$  dig -x `dig ldap-srv-repl.example.com +short` +short
ldap-srv-repl.example.com.
```

Убедитесь, что время на **ldap-srv** и **ldap-srv-repl** совпадает. Часы всех наших серверов должны быть синхронизированы с использованием NTP, атомных часов или любого другого источника.

11.2.2 Развёртывание OpenLDAP на сервере-реплике

Где работаем: **ldap-srv-repl**

Зададим пароль для суперпользователя и установим необходимые пакеты:

```
#  passwd
#  apt-get install -y slapd ldap-utils krb5-kdc-ldap krb5-pkinit wamerican libsasl2-modules-gssapi-mit
```

Теперь мы должны настроить сервер OpenLDAP с пустой базой данных. Подробно этот процесс описан в разделе 2. Но некоторые детали будут различаться, поэтому нельзя дословно повторять его здесь. Тезисно

пробежим по основным моментам.

Создадим временный каталог для работы со службой каталогов:

```
$  mkdir ~/ldap
$  cd ~/ldap
```

Подготовим каталоги для конфигурации и базы данных демона **slapd**:

```
#  service slapd stop
#  rm -rf /etc/ldap/slapd.d
#  rm -rf /var/lib/ldap
#  mkdir /etc/ldap/slapd.d
#  mkdir /var/lib/ldap
#  chown openldap:openldap /var/lib/ldap
#  chmod 0700 /var/lib/ldap
```

Утянем с сервера **ldap-srv** схемы наборов данных в формате LDIF (мы их готовили самостоятельно):

```
#  scp user@ldap-srv.example.com:/etc/ldap/schema/{autofs,sudo,kerberos}.ldif /etc/ldap/schema
```

Подготовим посоленый хэш пароля для администратора OpenLDAP на текущем сервере:

```
$  slappasswd -h '{SSHA}'
New password:
Re-enter new password:
{SSHA}AUEv75PLvTDOyrO5yYvQXmyRurYwTdmn
```

Теперь у нас есть всё, чтобы написать конфигурацию сервера в LDIF-формате. Для понимания синтаксиса Syncrepl ознакомьтесь с документацией. Запишем в файл **11.2.2-consumer.slapd.ldif**:

```
dn: cn=config
objectClass: olcGlobal
cn: config
olcPidFile: /var/run/slapd/slapd.pid
olcArgsFile: /var/run/slapd/slapd.args

dn: olcDatabase={0}config,cn=config
objectClass: olcDatabaseConfig
olcDatabase: {0}config
olcRootDN: cn=config
olcAccess: to *
  by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" manage
  by * none

dn: cn=schema,cn=config
objectClass: olcSchemaConfig
cn: schema

dn: cn=module,cn=config
objectClass: olcModuleList
cn: module
olcModulePath: /usr/lib/ldap
olcModuleLoad: back_mdb.la
olcModuleLoad: back_monitor.la

include: file:///etc/ldap/schema/core.ldif
include: file:///etc/ldap/schema/cosine.ldif
include: file:///etc/ldap/schema/inetorgperson.ldif
include: file:///etc/ldap/schema/collective.ldif
include: file:///etc/ldap/schema/corba.ldif
include: file:///etc/ldap/schema/duaconf.ldif
include: file:///etc/ldap/schema/openldap.ldif
include: file:///etc/ldap/schema/dyngroup.ldif
include: file:///etc/ldap/schema/java.ldif
include: file:///etc/ldap/schema/misc.ldif
include: file:///etc/ldap/schema/nis.ldif
include: file:///etc/ldap/schema/ppolicy.ldif
include: file:///etc/ldap/schema/kerberos.ldif
include: file:///etc/ldap/schema/sudo.ldif
include: file:///etc/ldap/schema/autofs.ldif

# Основная база данных
```

```
dn: olcDatabase=mdb,cn=config
objectClass: olcMdbConfig
olcDatabase: mdb
olcSuffix: dc=example,dc=com
olcDbDirectory: /var/lib/ldap
olcDbMaxsize: 1073741824
olcRootDN: cn=admin,dc=example,dc=com
olcRootPW: {SSHA}AUEv75PLvTDOyrO5yYvQXmyRurYwTdmn
# Специальные индексы для Syncrepl
olcDbIndex: entryUUID eq
# Директивы Syncrepl
olcSyncRepl: rid=0
  provider=ldap://ldap-srv.example.com:389
  bindmethod=simple
  starttls=yes
  binddn="cn=replication,dc=example,dc=com"
  credentials="пароль.пользователя.replication"
  searchbase="dc=example,dc=com"
  logbase="cn=accesslog"
  logfilter="(&(objectClass=auditWriteObject)(reqResult=0))"
  schemachecking=on
  type=refreshAndPersist
  retry="60 +"
  syncdata=accesslog
# Ссылка на мастер-сервер (поставщик)
olcUpdateRef: ldap://ldap-srv.example.com
# Правила доступа практически идентичны
# (нет записи для пользователя replication)
olcAccess: {0}to *
  by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth manage
  by * break
olcAccess: {1}to attrs=userPassword,userPKCS12
  by dn.regex="uid=.*/admin,cn=example.com,cn=gssapi,cn=auth" manage
  by self write
  by anonymous auth
olcAccess: {2}to attrs=shadowLastChange
  by dn.regex="uid=.*/admin,cn=example.com,cn=gssapi,cn=auth" manage
  by self write
olcAccess: {3}to dn.subtree="cn=kerberos,ou=services,dc=example,dc=com"
  by dn.exact="cn=krbadmin,ou=users,dc=example,dc=com" write
  by dn.regex="uid=.*/admin,cn=example.com,cn=gssapi,cn=auth" read
olcAccess: {4}to dn.subtree="ou=autofs,ou=services,dc=example,dc=com"
  by dn.regex="uid=.*/admin,cn=example.com,cn=gssapi,cn=auth" manage
  by dn.regex="uid=autofsclient/.*,cn=example.com,cn=gssapi,cn=auth" read
olcAccess: {5}to *
  by dn.regex="uid=.*/admin,cn=example.com,cn=gssapi,cn=auth" manage
  by dn.exact="cn=krbadmin,ou=users,dc=example,dc=com" write
  by self read
  by dn.base="cn=nssproxy,ou=users,dc=example,dc=com" read

dn: olcDatabase=monitor,cn=config
objectClass: olcDatabaseConfig
olcDatabase: monitor
```

Атрибут **`olcUpdateRef`** играет очень важную роль. Вы можете настроить клиентские машины для использования сервера-реплики в качестве поставщика информации из каталога. Но эта информация будет доступна только для чтения. Если клиентская машина (или пользователь) выполнит запрос, требующий изменения записи, его запрос будет перенаправлен на основной сервер (**`ldap-srv.example.com`**).

Сгенерируем первичную конфигурацию:

```
#  slapadd -n 0 -F /etc/ldap/slapd.d -l 11.2.2-consumer.slapd.ldif
_#################### 100.00% eta   none elapsed            none fast!
Closing DB...
```

Проверим корректность конфигурации:

```
#  slaptest -uF /etc/ldap/slapd.d
config file testing succeeded
```

Поправим права доступа к файлам конфигурации:

```
#  chown -R openldap:openldap /etc/ldap/slapd.d
```

Подготовим систему журналирования. В конфигурации **/etc/rsyslog.conf** добавим несколько строк:

```
$ModLoad imuxsock # provides support for local system logging
$ModLoad imklog   # provides kernel logging support

$KLogPermitNonKernelFacility on

$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

$RepeatedMsgReduction on

$FileOwner syslog
$FileGroup adm
$FileCreateMode 0640
$DirCreateMode 0755
$Umask 0022
$PrivDropToUser syslog
$PrivDropToGroup syslog

$WorkDirectory /var/spool/rsyslog

$IncludeConfig /etc/rsyslog.d/*.conf

# Send slapd logs to /var/log/slapd.log
if $programname == 'slapd' then /var/log/slapd.log
& ~

if $programname == 'krb5kdc' then /var/log/krb5kdc.log
& ~
```

Создадим файлы для журналов наших демонов и зададим для них права доступа. Затем перезапустим **rsyslog**, чтобы изменения вступили в силу:

```
#  touch /var/log/{slapd,krb5kdc}.log
#  chmod 0640 /var/log/{slapd,krb5kdc}.log
#  chown syslog:adm /var/log/{slapd,krb5kdc}.log
#  service rsyslog restart
```

Настроим **logrotate** для управления этими журналами.

Создадим файл конфигурации **/etc/logrotate.d/slapd** и запишем в него:

```
/var/log/slapd.log {
 daily
 missingok
 rotate 7
 compress
 delaycompress
 notifempty
}
```

Файл **/etc/logrotate.d/krb5kdc** для демона **krb5-kdc**:

```
/var/log/krb5kdc.log {
 daily
 missingok
 rotate 7
 compress
 delaycompress
 notifempty
}
```

Проверим настройку **logrotate**:

```
# logrotate -df /etc/logrotate.conf
```

Отредактируем общесистемную конфигурацию демона **slapd** в файле **/etc/default/slapd**:

```
SLAPD_CONF=
SLAPD_USER="openldap"
```

```
SLAPD_GROUP="openldap"
SLAPD_PIDFILE=
SLAPD_SERVICES="ldap:/// ldapi:///"
SLAPD_SENTINEL_FILE=/etc/ldap/noslapd
SLAPD_OPTIONS="-4"
```

11.2.3 Первичная загрузка DIT на сервер-реплику

Первичная загрузка позволяет загрузить DIT на сервер-реплику через файл, не прибегая к репликации и без формирования большого потока траффика в сети. Это не обязательно, но если DIT достаточно большое, мы можем сэкономить много времени.

Но у Вас ничего не получится, если версия используемого сервера OpenLDAP — 2.4.31 и более ранняя. Связано такое поведение с ошибкой ITS#7255, исправленной в версии 2.4.32.

Проверить версию можно так:

```
$  dpkg --status slapd|grep Version
Version: 2.4.31-1+nmu2ubuntu8
```

Самый благоразумный вариант при развёртывании OpenLDAP в такой ситуации — не предпринимать ничего и просто пропустить этот раздел. Пусть пакет **slapd** спокойно обновится и данный функционал у Вас заработает. А первичную загрузку DIT можно произвести просто запустив механизм репликации по сети. Но в любом случае с информацией ознакомиться стоит. Она описывает более подробно процесс, тезисно зафиксированный в документации.

Где работаем: **ldap-srv**

Последний этап пред тем, как мы запустим репликацию — выгрузить всё DIT от поставщика потребителю.

Выгрузим всё DIT в LDIF-файл. В целях безопасности сначала зададим **umask**:

```
#  umask 066
#  slapcat | tee -a /tmp/provider.slapcat.ldif
```

Где работаем: **ldap-srv-repl**

Создадим каталог-приёмник для этого файла:

```
$  mkdir ~/dit
$  chmod u=rwx,g=,o= ~/dit
```

Где работаем: **ldap-srv**

Отправим файл на сервер-реплику (если уж DIT совсем большое, вместо **scp** можно использовать съёмный носитель). Затем — удалим исходный файл:

```
#  scp /tmp/provider.slapcat.ldif user@ldap-srv-repl.example.com:~/dit
#  shred /tmp/provider.slapcat.ldif
```

Где работаем: **ldap-srv-repl**

Уничтожим каталог с базой данных и воссоздадим его пустым:

```
#  service slapd stop
#  rm -rf /var/lib/ldap
```

```
#  mkdir /var/lib/ldap
#  chmod u=rwx,g=,o= /var/lib/ldap
#  chown openldap:openldap /var/lib/ldap
```

Загрузим DIT из скопированного нами файла в базу данных поставщика. Обратите внимание на модификатор « **-w** » у команды **slapadd**. Он позволит записать в базу данных информацию о контексте Syncrepl. Как только все записи будут загружены, значение **contextCSN** обновится в соответствии с самым большим **entryCSN**. В нашем случае это очень кстати :)

```
#  slapadd -l provider.slapcat.ldif -w
```
(именно здесь возникает проблема при использовании **slapd** версии 2.4.31, процесс загрузки просто зависает)

И снова уничтожим временный файл, теперь уже на клиентской машине. В конце концов, в этом файле содежится всё DIT открытым текстом:

```
$  shred ~/dit/provider.slapcat.ldif
```

Убедимся, что всё DIT находится на сервере-реплике:

```
$  ldapsearch -xWD cn=admin,dc=example,dc=com -b dc=example,dc=com -H ldap://ldap-srv-repl.example.com
```

11.2.4 Запуск слубы каталогов на сервере-реплике

Где работаем: **ldap-srv-repl**

Запускаем демон **slapd** и добавляем его в автозагрузку:

```
#  service slapd start
#  update-rc.d slapd defaults
```

Слегка изменим права доступа с помощью LDIF-файла **11.2.4-consumer.acl.ldif**:

```
dn: olcDatabase={0}config,cn=config
changetype: modify
replace: olcRootDN
olcRootDN: cn=admin,dc=example,dc=com
-
replace: olcAccess
olcAccess: to *
  by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" manage
  by dn.regex="uid=.*/admin,cn=example.com,cn=gssapi,cn=auth" manage
  by * none
```

Загрузим изменение ACL:

```
#  ldapmodify -QY EXTERNAL -H ldapi:/// -f 11.2.4-consumer.acl.ldif
modifying entry "olcDatabase={0}config,cn=config"
```

Настроим общесистемную конфигурацию клиентских запросов в файле **/etc/ldap/ldap.conf**. Это немного сократит количество печатаемого нами текста. Пока не обращайте внимание на параметры TLS, мы разберёмся с ними в следующем разделе. Не забудьте, что в этом разделе мы работаем с **ldap-srv-repl.example.com** и запросы мы отправляем именно ему. Это важно, потому что мы хотим опрашивать и модифицировать только потребителя, не поставщика (**ldap-srv.example.com**). Итак, запишем в **/etc/ldap/ldap.conf**:

```
BASE  dc=example,dc=com
URI  ldap://ldap-srv-repl.example.com
# TLS_CACERT /etc/ssl/certs/rootca.crt
# TLS_CRLFILE /etc/ssl/rootca.crl
# TLS_REQCERT demand
TIMELIMIT 15
TIMEOUT  20
```

Проверим, может ли наш администратор выполнить запрос:

```
$  ldapwhoami -WD cn=admin,dc=example,dc=com
Enter LDAP Password:
dn:cn=admin,dc=example,dc=com
```

Если в соответствии с разделом 8.4.2 мы меняли схему данных nis, то нам придётся это повторить и здесь. Три простых шага и всё готово:

Посмотрим, какой порядковый номер у схемы данных **nis** в нашем каталоге и у атрибута **nisNetgroupTriple** в схеме данных **nis**:

```
$  ldapsearch -LLLxWD cn=admin,dc=example,dc=com -b cn=schema,cn=config dn |grep nis
Enter LDAP Password:
dn: cn={10}nis,cn=schema,cn=config
$  ldapsearch -LLLxWD cn=admin,dc=example,dc=com -b cn=schema,cn=config | grep nisNetgroupTriple
Enter LDAP Password:
olcAttributeTypes: {12}( 1.3.6.1.1.1.1.14 NAME 'nisNetgroupTriple' DESC 'Netgr
  a netgroup' SUP top STRUCTURAL MUST cn MAY ( nisNetgroupTriple $ memberNisNe
```

Номера 10 и 12 соответственно. Сформируем новый LDIF под названием **11.2.4-nis-schema-change.ldif** для изменения схемы. Если два последних номера у Вас отличаются — подставьте свои значения:

```
dn: cn={10}nis,cn=schema,cn=config
changetype: modify
delete: olcAttributeTypes
olcAttributeTypes: {12}( 1.3.6.1.1.1.1.14 NAME 'nisNetgroupTriple' DESC 'Netgr
 oup triple' SYNTAX 1.3.6.1.1.1.0.0 )
-
add: olcAttributeTypes
olcAttributeTypes: {12}( 1.3.6.1.1.1.1.14 NAME 'nisNetgroupTriple' DESC 'Netgr
 oup triple' EQUALITY caseExactIA5Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

Загрузим его в наш сервер каталогов:

```
$  ldapadd -xZWD cn=admin,dc=example,dc=com  -f 11.2.4-consumer.nis-change.ldif
Enter LDAP Password:
modifying entry "cn={10}nis,cn=schema,cn=config"
```

Дважды проверьте по журналам, что Вы подключаетесь именно к серверу-реплике!

11.2.5 Настройка TLS для сервера-реплики

Где работаем: **ldap-srv**

Создадим закрытый ключ для сервера-реплики:

```
#  cd /root/CA
#  openssl genrsa -out private/ldap-srv-repl.example.com.key 4096
#  chmod 400 private/ldap-srv-repl.example.com.key
```

Сгенерируем запрос на подпись сертификата:

```
#  openssl req -sha256 -new \
       -key private/ldap-srv-repl.example.com.key -out certs/ldap-srv-repl.example.com.csr \
       -subj /C=RU/ST=Moscow/L=Moscow/O=ExampleInc/OU=ITdept/CN=ldap-srv-
repl.example.com/emailAddress=support@example.com
```

Подпишем его с помощью своего CA:

```
#  openssl ca -extensions usr_cert -notext -md sha256 \
```

```
       -keyfile private/rootca.key -cert certs/rootca.crt \
       -in certs/ldap-srv-repl.example.com.csr -out certs/ldap-srv-repl.example.com.crt
#  chmod 444 certs/ldap-srv-repl.example.com.crt
```

Теперь нужно безопасным образом переместить на сервер **ldap-srv-repl** четыре файла:

- Корневой сертификат, **rootca.crt**;
- Список отозванных сертификатов, **rootca.crl**;
- Закрытый ключ нового сервера, **ldap-srv-repl.example.com.key**;
- Сертификат нового сервера, **ldap-srv-repl.example.com.crt**.

Где работаем: **ldap-srv-repl**

Подготовим каталог для временного хранения этой информации на сервере-реплике в домашнем каталоге пользователя, которого Вы создали при установке операционной системы (допустим, **user**):

```
$  whoami
user
$  mkdir ~/certs
$  chmod u=rwx,g=,o= ~/certs
```

Где работаем: **ldap-srv**

Скопируем файлы в этот каталог с сервера, где находится наш CA, используя учётную запись **user**:

```
#  pwd
/root/CA
#  scp private/ldap-srv-repl.example.com.key certs/ldap-srv-repl.example.com.crt certs/rootca.crt
crl/rootca.crl user@ldap-srv-repl.example.com:~/certs
```

Где работаем: **ldap-srv-repl**

Создадим каталог для ключевой информации нашего сервера и поместим туда получившиеся у нас файлы:

```
#  mkdir /etc/ldap/ssl
#  chown openldap:openldap /etc/ldap/ssl
#  chmod 0500 /etc/ldap/ssl
#  cd /home/user/certs
#  mv ldap-srv-repl.example.com.crt ldap-srv-repl.example.com.key /etc/ldap/ssl
```

Установим права доступа для ключевой информации:

```
#  chown openldap:openldap /etc/ldap/ssl/{ldap-srv-repl.example.com.crt,ldap-srv-repl.example.com.key}
#  chmod 0400 /etc/ldap/ssl/{ldap-srv-repl.example.com.crt,ldap-srv-repl.example.com.key}
```

Поместим корневой сертификат и список отозванных сертификатов в каталог с сертификатами операционной системы и зададим для них права доступа:

```
#  mv rootca.crt /etc/ssl/certs/
#  chmod 0644 /etc/ssl/certs/rootca.crt
#  mv rootca.crl /etc/ssl/
#  chmod 0644 /etc/ssl/rootca.crl
```

Опишем конфигурацию TLS для **ldap-srv-repl** в LDIF-файле **11.2.5-tls.consumer.ldif**:

```
dn: cn=config
add: olcTLSCertificateFile
olcTLSCertificateFile: /etc/ldap/ssl/ldap-srv-repl.example.com.crt
```

```
-
add: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/ldap/ssl/ldap-srv-repl.example.com.key
-
add: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/ssl/certs/rootca.crt
-
add: olcTLSVerifyClient
olcTLSVerifyClient: never
```

Загрузим конфигурацию:

```
$  ldapmodify -xWD cn=admin,dc=example,dc=com -f 11.2.5-tls.consumer.ldif
```

И проверим результат:

```
#  ldapsearch -QLLLY EXTERNAL -H ldapi:/// -b cn=config -s base | grep olcTLS
olcTLSCertificateFile: /etc/ldap/ssl/ldap-srv-repl.example.com.crt
olcTLSCertificateKeyFile: /etc/ldap/ssl/ldap-srv-repl.example.com.key
olcTLSCACertificateFile: /etc/ssl/certs/rootca.crt
olcTLSVerifyClient: never
```

Раскомментируем строчки в **`/etc/ldap/ldap.conf`**:

```
BASE  dc=example,dc=com
URI  ldap://ldap-srv-repl.example.com
TLS_CACERT /etc/ssl/certs/rootca.crt
TLS_CRLFILE /etc/ssl/rootca.crl
TLS_REQCERT demand
TIMELIMIT 15
TIMEOUT  20
```

И попробуем выполнить запрос с использованием TLS (параметр **`Z`**):

```
$  ldapwhoami -xZZWD cn=admin,dc=example,dc=com
Enter LDAP Password:
dn:cn=admin,dc=example,dc=com
```

С помощью LDIF-файла **`11.2.5-consumer.dbindexes.ldif`** изменим настройки журналирования сервера каталогов и добавим индексы для основной базы данных:

```
dn: cn=config
changetype: modify
replace: olcLogLevel
olcLogLevel: stats

dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: entryCSN eq
-
add: olcDbIndex
olcDbIndex: uid eq
-
add: olcDbIndex
olcDbIndex: cn eq
-
add: olcDbIndex
olcDbIndex: ou eq
-
add: olcDbIndex
olcDbIndex: dc eq
```

Теперь журнал сервера будет достаточно подробным. Это поможет нам в отладке. Загрузим конфигурацию:

```
$  ldapadd -xZZWD cn=admin,dc=example,dc=com -f 11.2.5-consumer.dbindexes.ldif
```

Проверим, что у нас получилось в итоге следующей командой:

```
#  ldapsearch -QLLLY EXTERNAL -H ldapi:/// -b cn=config "(|(cn=config)(olcDatabase={1}mdb))"
```

11.2.6 Настройка SASL GSSAPI для сервера-реплики

Где работаем: **ldap-srv-repl**

Включим аутентификацию SASL GSSAPI на новом сервере каталогов с помощью LDIF-файла **11.2.6-consumer.gssapi.ldif**:

```
dn: cn=config
changetype: modify
add: olcSaslHost
olcSaslHost: ldap-srv-repl.example.com
-
add: olcSaslRealm
olcSaslRealm: EXAMPLE.COM
-
add: olcSaslSecProps
olcSaslSecProps: noanonymous,noplain
```

Загрузим конфигурацию:

```
$  ldapadd -xZZWD cn=admin,dc=example,dc=com -f 11.2.6-consumer.gssapi.ldif
```

Теперь нам нужно добавить параметр **KRB5_KTNAME** в файл **/etc/default/slapd**:

```
SLAPD_CONF=
SLAPD_USER="openldap"
SLAPD_GROUP="openldap"
SLAPD_PIDFILE=
SLAPD_SERVICES="ldap:/// ldapi:///"
SLAPD_SENTINEL_FILE=/etc/ldap/noslapd
SLAPD_OPTIONS="-4"
export KRB5_KTNAME=/etc/ldap/krb5.keytab
```

Скопируем конфигурацию-описание области EXAMPLE.COM с основного сервера, чтобы не было проблем при подключении с помощью устилиты **kadmin**:

```
#  scp user@ldap-srv.example.com:/etc/krb5.conf /etc/
```

Создадим набор ключей Kerberos для сервера OpenLDAP, ключ принципала сервера **ldap-srv-repl** и ключ **autofsclient** для автоматического монтирования на этом сервере в диалоге команды **kadmin**. **ВНИМАНИЕ:** заметьте, как мы размещаем принципалы в наборах ключей. Принципалы **host/** и **autofsclient/** размещаются в **/etc/krb5.keytab**, а принципал **ldap/** — в **/etc/ldap/krb5.keytab**:

```
#  kadmin -p pablo/admin@EXAMPLE.COM
kadmin:  addprinc -randkey ldap/ldap-srv-repl.example.com@EXAMPLE.COM
kadmin:  ktadd -k /etc/ldap/krb5.keytab ldap/ldap-srv-repl.example.com@EXAMPLE.COM
kadmin:  addprinc -randkey host/ldap-srv-repl.example.com@EXAMPLE.COM
kadmin:  ktadd -k /etc/krb5.keytab host/ldap-srv-repl.example.com@EXAMPLE.COM
kadmin:  addprinc -randkey autofsclient/ldap-srv-repl.example.com@EXAMPLE.COM
kadmin:  ktadd -k /etc/krb5.keytab autofsclient/ldap-srv-repl.example.com@EXAMPLE.COM
kadmin:  exit
```

Поправим права доступа на новый набор ключей:

```
#  chown root:openldap /etc/ldap/krb5.keytab
#  chmod 640 /etc/ldap/krb5.keytab
```

Перезапустим демон службы каталогов:

```
#  service slapd restart
```

Проверим аутентификацию SASL GSSAPI:

```
$  kdestroy
$  kinit -p pablo/admin@EXAMPLE.COM
$  ldapwhoami
SASL/GSSAPI authentication started
SASL username: pablo/admin@EXAMPLE.COM
SASL SSF: 56
SASL data security layer installed.
dn:uid=pablo/admin,cn=example.com,cn=gssapi,cn=auth
```

11.2.7 Проверка репликации

Сервер-реплика **ldap-srv-repl.example.com** (потребитель OpenLDAP), настроен с использованием TLS и SASL GSSAPI. Сервер получает данные от поставщика **ldap-srv.example.com**. Проверим, работает ли процедура репликации. В качестве примера будем менять рабочую оболочку (**loginShell**) пользователя **test.user**.

Где работаем: **ldap-srv**

Посмотрим текущее значение атрибута **loginShell** пользователя **test.user**:

```
$  ldapsearch -QLLLZZb cn=test.user,ou=users,dc=example,dc=com loginShell
dn: cn=test.user,ou=users,dc=example,dc=com
loginShell: /bin/bash
```

Где работаем: **ldap-srv-repl**

Проверим значение того же поля на сервере-реплике:

```
$  ldapsearch -QLLLZb cn=test.user,ou=users,dc=example,dc=com loginShell
dn: cn=test.user,ou=users,dc=example,dc=com
loginShell: /bin/bash
```

Где работаем: **ldap-srv**

Поменяем оболочку пользователя на **/bin/sh**:

```
$  ldapmodify <<-EOF
dn: cn=test.user,ou=users,dc=example,dc=com
changetype: modify
replace: loginShell
loginShell: /bin/sh
EOF
```

Где работаем: **ldap-srv-repl**

Проверим, применились ли изменения на сервере-реплике:

```
$  ldapsearch -QLLLZb cn=test.user,ou=users,dc=example,dc=com loginShell
dn: cn=test.user,ou=users,dc=example,dc=com
loginShell: /bin/sh
```

Репликация работает! Ура!

11.2.8 Настройка подчиненного сервера Kerberos

Смысл репликации DIT — иметь запасную копию на случай выхода из строя основного сервера (поставщика). Но наше DIT помимо прочего поддерживает инфраструктуру Kerberos. Поэтому мы должны настроить сервер-реплику так, чтобы он выполнял роль подчинённого сервера Kerberos на случай выхода из строя основного.

Для развёртывания подчинённого сервера Kerberos мы уже установили необходимые пакеты и произвели некоторые настройки (в предыдущих разделах). Сейчас нам нужно переместить на подчинённый сервер три файла: закрытый ключ Kerberos, конфигурацию области и файл-тайник.

Где работаем: **ldap-srv-repl**

Создадим на сервере-реплике каталог для их временного хранения:

```
$  mkdir ~/kerberos
$  chmod u=rwx,g=,o= ~/kerberos/
$  cd ~/kerberos/
```

Где работаем: **ldap-srv**

Скопируем файлы во временный каталог:

```
#  scp /etc/krb5kdc/{.k5.EXAMPLE.COM,kdc.conf,kadm5.acl} /etc/krb5.d/stash.keyfile user@ldap-srv-
repl.example.com:~/kerberos
```

Где работаем: **ldap-srv-repl**

Поместим эти файлы на свои места и поправим права доступа:

```
#  mkdir /etc/krb5.d/
#  chmod u=rwx,g=,o= /etc/krb5.d/
#  mv {.k5.EXAMPLE.COM,kdc.conf,kadm5.acl} /etc/krb5kdc/
#  mv stash.keyfile /etc/krb5.d/
#  chown -R root:root /etc/krb5kdc /etc/krb5.d
```

Добавим в базу данных некоторые индексы. Запишем в файл **`11.2.8-kerberos.indexes.ldif`**:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: objectClass eq
-
add: olcDbIndex
olcDbIndex: krbPrincipalName eq
-
add: olcDbIndex
olcDbIndex: sudoUser eq,sub
-
add: olcDbIndex
olcDbIndex: uidNumber,gidNumber,memberUid,uniqueMember eq
```

Загрузим конфигурацию:

```
$  ldapadd -QZZf 11.2.6-kerberos.indexes.ldif
```

Проверим, на месте ли новые индексы:

```
$  ldapsearch -QZZLLLb olcDatabase={1}mdb,cn=config olcDbIndex
dn: olcDatabase={1}mdb,cn=config
olcDbIndex: entryUUID eq
```

```
olcDbIndex: entryCSN eq
olcDbIndex: uid eq
olcDbIndex: cn eq
olcDbIndex: ou eq
olcDbIndex: dc eq
olcDbIndex: objectClass eq
olcDbIndex: krbPrincipalName eq
olcDbIndex: sudoUser eq,sub
olcDbIndex: uidNumber,gidNumber,memberUid,uniqueMember eq
```

Убедитесь, что подчинённый KDC стартует автоматически и запустите его. Обратите внимание, что демон управления доменом Kerberos (**kadmind**) на подчинённом сервере мы не запускаем!

```
#  update-rc.d  krb5-kdc defaults
#  service krb5-kdc start
```

11.2.9 Резервное копирование и восстановление сервера-реплики

Настройте резервное копирование и восстановление на сервере-реплике в соответствии с методикой, приведённой в разделе 10.

11.3 Проверка статуса репликации

Порой файлы журналов не дают чёткого представления, закончен процесс репликации или нет. Чтобы в этом удостовериться, просто запросите у поставщика и потребителя значение contextCSN. Если ответы от обоих серверов совпадают, репликация закончена.

Где работаем: **ldap-srv**

```
$  kdestroy
$  kinit -p pablo/admin
$  ldapsearch  -ZZQLLL -s base contextCSN
dn: dc=example,dc=com
contextCSN: 20150112110636.224261Z#000000#000#000000
```

Где работаем: **ldap-srv-repl**

```
$  kdestroy
$  kinit -p pablo/admin
$  ldapsearch  -ZZQLLL -s base contextCSN
dn: dc=example,dc=com
contextCSN: 20150112110636.224261Z#000000#000#000000
```

Всё отлично.

11.4 Настройка клиента

Где работаем: **ldap-client**

Когда второй сервер OpenLDAP заработал, нам остаётся настроить клиентов, чтобы они использовали оба. Для

этого отредактируем несколько файлов.

Конфигурация **sudoers** через LDAP (**/etc/sudo-ldap.conf**):

```
BASE    dc=example,dc=com
URI     ldap://ldap-srv.example.com ldap://ldap-srv-repl.example.com
BINDDN  cn=nssproxy,ou=users,dc=example,dc=com
BINDPW  пароль.пользователя.nssproxy
TLS_CACERTFILE  /etc/ssl/certs/rootca.crt
TLS_CRLFILE     /etc/ssl/rootca.crl
TLS_CHECKPEER   no
TIMELIMIT       15
TIMEOUT         20
SUDOERS_BASE    ou=sudo,ou=services,dc=example,dc=com
```

Клиентская конфигурация LDAP (**/etc/ldap/ldap.conf**):

```
BASE  dc=example,dc=com
URI  ldap://ldap-srv.example.com ldap://ldap-srv-repl.example.com
TLS_CACERT /etc/ssl/certs/rootca.crt
TLS_CRLFILE /etc/ssl/rootca.crl
TLS_REQCERT demand
TIMELIMIT 15
TIMEOUT  20
SASL_MECH GSSAPI
```

Конфигурация локальной службы имён (**/etc/nslcd.conf**):

```
uid nslcd
gid nslcd
uri ldap://ldap-srv.example.com
uri ldap://ldap-srv-repl.example.com
base dc=example,dc=com
binddn cn=nssproxy,ou=users,dc=example,dc=com
bindpw nssproxyuserpass
rootpwmoddn cn=admin,dc=example,dc=com
base group ou=groups,dc=example,dc=com
base passwd ou=users,dc=example,dc=com
base shadow ou=users,dc=example,dc=com
bind_timelimit 5
timelimit 10
idle_timelimit 60
ssl start_tls
tls_reqcert demand
tls_cacertfile /etc/ssl/certs/rootca.crt
nss_initgroups_ignoreusers bin,daemon,games,lp,mail,nobody,nslcd,root,sshd,sync,uucp
nss_initgroups_ignoreusers sys,man,news,proxy,www-data,backup,list,irc,gnats,landscape
```

Конфигурация Kerberos (**/etc/krb5.conf**):

```
[logging]
 default = SYSLOG:INFO:LOCAL1
 kdc = SYSLOG:NOTICE:LOCAL1
 admin_server = SYSLOG:WARNING:LOCAL1

[libdefaults]
 default_realm = EXAMPLE.COM
 dns_lookup_realm = false
 dns_lookup_kdc = false
 ticket_lifetime = 24h
 renew_lifetime = 7d
 forwardable = true

[realms]
 EXAMPLE.COM = {
  kdc = ldap-srv.example.com
  kdc = ldap-srv-repl.example.com
  admin_server = ldap-srv.example.com
  master_kdc = ldap-srv.example.com
  default_domain = example.com
  database_module = openldap_ldapconf
 }

[domain_realm]
```

```
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM

[appdefaults]
 pam = {
  debug = false
  ticket_lifetime = 36000
  renew_lifetime = 36000
  forwardable = true
  krb4_convert = false
 }
```

Как можно видеть, изменения требуются совсем небольшие. Поэтому, возможно, Вы захотите такой процесс автоматизировать. Сделать это можно разными путями. От простой раздачи файлов по сети с помощью ftp/http/nfs до развесистой системы управления версиями вроде Puppet.

После внесения всех изменений на клиентской машине, попробуйте выключить **ldap-srv.example.com** и проверить, будут ли доступны сервисы, которые он поддерживал, с нового сервера, **ldap-srv-repl.example.com**.

11.5 Устранение проблем

11.5.1 Настройка журналирования Syncrepl

Если Вы испытываете трудности с Syncrepl, попробуйте настроить журналирование:

```
ldapmodify -Z <<-EOF
dn: cn=config
changetype: modify
replace: olcLogLevel
olcLogLevel: stats sync
EOF
```

Вы можете сделать это как на поставщике, так и на потребителе.

11.5.2 syncrepl_message_to_entry

Если в журнале **slapd.log** Вы видите подобную запись:

```
syncrepl_message_to_entry: rid=000 mods check (objectClass: value #1 invalid per syntax)
```

… то это значит, что на сервере-реплике нет необходимой схемы набора данных. Конечно, значение **rid=000** может быть другим на Вашем сервере. Это идентификатор репликации, задаваемый в атрибуте **olcSyncrepl** на севере-реплике. Сравните набор загруженных схем с основным сервером (поставщиком). Они должны совпадать. Выявите, каких схем не хватает, выполнив на каждом сервере запрос:

```
$  ldapsearch -ZZQLLLb cn=schema,cn=config dn
```

Где работаем: **ldap-srv-repl**

Загрузите недостающие схемы на сервер-реплику, пользуясь таким шаблоном (с использованием Kerberos):

```
$  ldapadd -ZZQf schema.ldif
```

За дополнительной информацией обратитесь к разделу 2.5.