

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего профессионального образования

«Национальный исследовательский ядерный университет «МИФИ»



Факультет кибернетики и информационной
безопасности

Кафедра №36

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему:

Исследование возможностей сервера OpenLDAP для аутентификации

пользователей СУБД PostgreSQL.

Группа К9-361

Студент _____ (подпись) (_____ Воронин Д.Л.)
(ФИО)

Руководитель проекта _____ (подпись) (_____ Муравьев С.К.)
(ФИО)

Оценка _____

Члены комиссии _____ (подпись) (_____)
(ФИО)

_____ (подпись) (_____)
(ФИО)

_____ (подпись) (_____)
(ФИО)

_____ (подпись) (_____)
(ФИО)

Москва 2013г.



КАФЕДРА ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

Дата выдачи задания «1» сентября 2013г.

ОТЗЫВ О РАБОТЕ СТУДЕНТА

В ходе выполнения учебно-исследовательской работы студент Воронин Д.Л. выполнил исследование возможностей OpenLDAP для аутентификации пользователей СУБД PostgreSQL.

Полученные в данной работе результаты позволяют реализовать централизованное управление учётными записями пользователей в рамках распределённой информационной системы.

Студент изучил различные методы аутентификации, поддерживаемые СУБД PostgreSQL и особенности функционирования системы ведения каталогов OpenLDAP. В процессе выполнения работы Воронин Д.Л. был трудолюбив и самостоятелен, успешно осваивал новый материал.

К недостаткам работы можно отнести лишь незначительные замечания к оформлению пояснительной записки. Несмотря на данный недостаток, выполненная работа заслуживает оценки "отлично".

Руководитель Муравьев С.К.

«23» декабря 2013г.

Содержание

Введение	5
1 Обзор основных методов аутентификации PostgreSQL	6
1.1 Trust	6
1.2 Password	6
1.3 Ident	7
1.4 Peer	8
1.5 PAM	8
1.6 LDAP	11
1.7 Выбор метода аутентификации пользователей в распределенной многопользовательской системе	12
2 Метод аутентификации LDAP	13
2.1 Обзор протокола LDAP	13
2.2 OpenLDAP	16
2.2.1 Использование динамической конфигурации сервера OpenLDAP	16
2.2.2 Инструменты манипуляций данными в OpenLDAP	18
3 Настройка PostgreSQL с аутентификацией LDAP	21
3.1 Настройка машин стенда	22
3.1.1 Настройка сервера LDAP	22
3.1.2 Создание пользовательских аккаунтов	25
3.1.3 Создание сертификата сервера LDAP	27
3.1.4 Настройка клиентской машины	29
3.1.5 Настройка сервера PostgreSQL	29
3.2 Проверка работы стенда	31
Заключение	37
Список литературы	38

Введение

Системы управления базами данных являются необходимыми программными продуктами при построении информационных систем. Однако в этих системах СУБД часто являются основными целями атак злоумышленников.

Для предотвращения нежелательного доступа к информации в базах данных используются различные подходы к реализации безопасности. Одним из подходов является аутентификация — процедура проверки подлинности. Правильно выбранный метод аутентификации в СУБД позволяет минимизировать шанс взлома, а следовательно, и предотвратить несанкционированный доступ к информации, хранящейся в ней.

Одной из наиболее развитых систем управления базами данных является PostgreSQL. PostgreSQL — это свободно распространяемая объектно-реляционная система управления базами данных. Благодаря открытой лицензии и подробной документации PostgreSQL широко используется в различных информационных системах.

В учебно-исследовательской работе предлагается обзор основных методов аутентификации в системе управления базами данных PostgreSQL, производится выбор метода аутентификации пользователей СУБД в распределенной многопользовательской системе с использованием сервера OpenLDAP и описывается процесс настройки данного метода аутентификации. Целью данной учебно-исследовательской работы является исследование принципов работы сервера OpenLDAP и возможностей дальнейшей его доработки для интеграции с реализацией принудительного контроля доступа SELinux.

1 Обзор основных методов аутентификации PostgreSQL

1.1 Trust

Метод аутентификации **trust** позволяет любому, кто подключится к серверу баз данных, получить доступ к любой базе данных, включая базу данных администратора.

Данный метод часто применяется для локальных подключений в однопользовательских рабочих станциях. Однако его не рекомендуется использовать на многопользовательских машинах. При использовании **trust** на многопользовательских рабочих станциях рекомендуется ограничивать права на сокет **postmaster**, используя при том права файловой системы. За это отвечают параметры **unix_socket_directory**, **unix_socket_permissions**, **unix_socket_group** в конфигурационном файле **postgresql.conf**.

Локальные подключения к базе данных при методе **trust** не ограничиваются.

Метод **trust** подходит для TCP подключений, если каждому пользователю каждой рабочей станции разрешено подключение к серверу. Однако данный метод используется редко для этих целей.

Достоинства метода аутентификации trust:

- Не требует создания дополнительных пользователей ОС.

Недостатки метода аутентификации trust:

- Не позволяет разграничивать доступ пользователей.
- Разрешает подключение к любой базе данных (в том числе и системной) любому пользователю ОС.
- Применение на многопользовательских рабочих станциях требуют дополнительной настройки.

1.2 Password

Методы аутентификации по паролю (*Password-based*) включают в себя методы **md5**, **crypt** и **password**. Эти методы работают следующим образом: пользователю предлагается ввести пароль при подключении. Пароль передается на сервер базы данных и сравнивается с хранимым в глобальной системной таблице **pg_shadow**. При совпадении этих паролей пользователь авторизуется в базе данных.

При использовании метода **password** пароль высылается в базу данных в открытом (текстовом) виде.

Метод **crypt** аналогичен методу **password**, но в нем пароль пересылается не в простом текстовом виде, а с применением простой схемы шифрования, которая не обеспечивает надежной защиты, но это все же лучше, чем простая пересылка пароля в текстовом виде, как в методе **password**.

Аналогично, при использовании **md5** PostgreSQL сравнивает зашифрованные по алгоритму **md5** пароли.

Для назначения или изменения пароля пользователей используются SQL-запросы **CREATE USER** и **ALTER USER**. По умолчанию, если пароль пользователя не установлен, то в качестве пароля этого пользователя хранится пустая строка и аутентификация пользователя проводится не будет.

Достоинства методов аутентификации по паролю:

- Не требуют создания дополнительных пользователей ОС.

Недостатки методов аутентификации по паролю:

- Пароли пользователей передаются по нешифрованному каналу.
- При шифровании паролей при методах `md5`, `crypt` используются простые методы шифрования. Поэтому они могут быть дешифрованы и скомпрометированы.
- Позволяют любому пользователю ОС зайти в базу данных с именем пользователя при вводе правильного пароля.

1.3 Ident

Протокол идентификации **Ident** (*Identification Protocol*, *ident*, *Ident Protocol*) [1] обеспечивает способ идентификации пользователя для конкретного соединения TCP. Используя на входе номера пары соединенных между собой портов TCP, протокол возвращает строку символов, идентифицирующую владельца данного соединения на стороне сервера. Первоначально протокол идентификации называли Authentication Server Protocol (протокол аутентификации сервера). Новое имя лучше отражает суть протокола.

Протокол представляет собой сервис на основе соединений TCP. Сервер слушает соединения TCP для порта 113. После организации соединения сервер читает строку данных, содержащую сведения о цели соединения. При существовании идентификатора пользователя для соединения сервер передает этот идентификатор в качестве отклика. После этого сервер может закрыть соединение или продолжить диалог «запрос-отклик». Серверу следует закрывать соединение по истечении заданного конфигурационными параметрами тайм-аута (60-180) при отсутствии каких-либо запросов. Клиент может закрыть соединение в любой момент, однако для компенсации возможных задержек в сети клиенту следует выждать по крайней мере 30 секунд после запроса прежде, закрыть соединение.

Достоинства метода аутентификации Ident:

- Используется в архитектуре клиент-сервер.
- Простота архитектуры и сообщений передачи данных протокола.

Недостатки метода аутентификации Ident:

- Возможность подмены хоста и/или информации на хосте.
- Не предназначен для аутентификации или контроля доступа.
- Некоторые сервера, реализующие протокол Ident, возвращают зашифрованное имя пользователя. В PostgreSQL нет возможностей расшифровки имен, поэтому имена пользователей по протоколу Ident передаются в открытом виде.
- Аутентификация пользователя в PostgreSQL проводится при сравнении имен пользователя ОС и пользователя базы данных.

Протокол Ident не может быть использован в качестве основного метода аутентификации в PostgreSQL ввиду возможности подмены хоста и/или пользователя.

1.4 Peer

Метод аутентификации **Peer** [2] в PostgreSQL работает следующим образом: PostgreSQL берет имя пользователя операционной системы и сопоставляет его с названием базы данных. Если сопоставление имен успешно, пользователю предоставляются права на доступ к базе данных.

Достоинствами метода Peer являются:

1. Простота реализации протокола.

Недостатками метода Peer являются:

1. Применим только для локальных подключений.
2. Может быть использован только в операционных системах, в которых существует функция получения текущего значения реер.
3. Нешифрованный канал.

Метод аутентификации Peer может быть применим только для локальных соединений. Передача данных по данному протоколу осуществляется по незашифрованному каналу. Для обеспечения безопасности пользователей баз данных требуется устанавливать сложные пароли от своих аккаунтов в операционной системе.

1.5 PAM

Pluggable Authentication Modules (PAM), подключаемые модули аутентификации) [3] представляют собой набор разделяемых библиотек, которые позволяют интегрировать различные низкоуровневые методы аутентификации в виде единого высокоуровневого API.

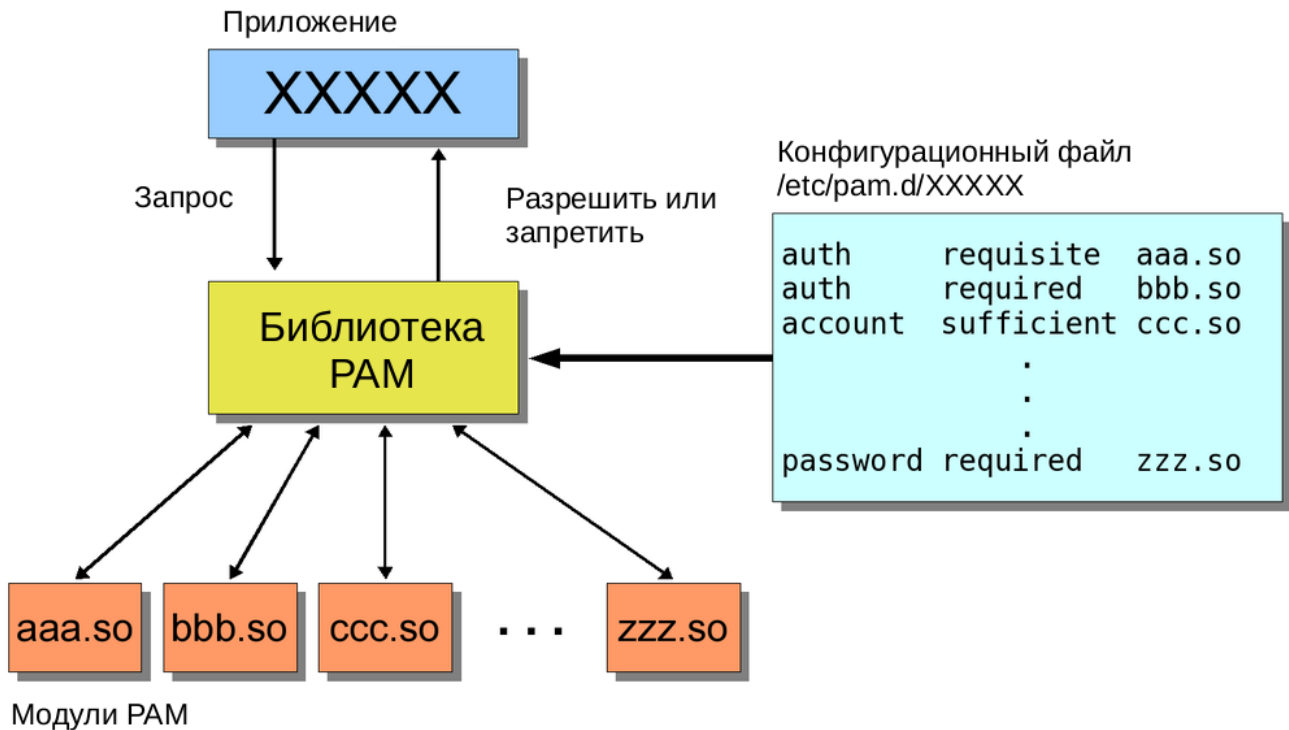


Рис. 1.1: Схема работы PAM с приложением

На *рисунке 1.1* представлена схема работы PAM с приложением.

Программы, которые используют данные паролей, не работают с этими данными напрямую, а делают запрос в службу PAM. PAM проводит процедуру аутентификации и возвращает результат: `PAM_SUCCESS` (доступ открыт) или `PAM_AUTH_ERR` (доступ закрыт). Таким образом, приложению не важно, каким образом PAM осуществляет аутентификацию.

PAM состоит из:

- Динамических библиотек — модулей, с помощью которых проводятся процедура аутентификации пользователя.
- Конфигурационных файлов PAM, описывающие порядок использования модулей при аутентификации.

Модули PAM в свою очередь, делятся на 4 типа:

1. **auth**. Это модули, которые выполняют функции аутентификации пользователей в системе (проверяют наличие пользователей в системе, сверяют имя и введенный пароль, проверяют права пользователя и разрешают доступ в ту или иную группу).
2. **account**. Проверяют доступность аккаунта на основе ресурсов системы (время суток или загрузка сервера).
3. **password**. Этот тип модулей осуществляет смену пароля по истечению срока действия.
4. **session**. Используются для выполнения определенных действий до начала или перед окончанием сеанса.

Принцип работы PAM следующий: приложение, которому требуется аутентификация, делает запрос в службу PAM. Служба PAM ищет конфигурационный файл, описывающий процесс аутентификации этого приложения, после чего выдает ответ, предоставить приложению права или нет. Схема работы метода аутентификации PAM приведен на *рисунке 1.1*.

Процесс аутентификации пользователей в какой-либо программе описывается в конфигурационном файле этой программы в каталоге `/etc/pam.d/` и представляет собой следующее:

```
<тип_модуля_1> <Флаг_контроля_1> <Путь_к_модулю_1> <Параметры_модуля_1>
...
<тип_модуля_N> <Флаг_контроля_N> <Путь_к_модулю_N> <Параметры_модуля_N>
```

В первом столбце указывается один из возможных типов модулей: `auth`, `account`, `password` или `session`.

Во втором — флаг контроля. Он указывает, как реагировать системе при успешном или неудачном прохождении модуля. Он предусматривает одно из нескольких значений:

- **required** — Аутентификация в этом модуле необходима для аутентификации в целом. В случае отказа в аутентификации в этом модуле проводится попытка аутентификации в оставшихся модулях с данным флагом.
- **requisite** — Аутентификация в этом модуле также необходима для аутентификации в целом, но в случае отказа в аутентификации в этом модуле, управление будет возвращено приложению.
- **sufficient** — Успешная аутентификация в этом модуле считается достаточной для признания всей аутентификации в целом, если не было отказа на предшествующих модулях с флагом **required**. Отказ в аутентификации в этом модуле не считается фатальной для всей последующей аутентификации.
- **optional** — Этот модуль не критичен для аутентификации и используется как дополнительный.

В следующем столбце указывается путь к динамической библиотеки модуля.

В последнем столбце указываются некоторые параметры модуля.

В следующем листинге приведен конфигурационный файл `/etc/pam.d/login` программы `login`:

```
##PAM-1.0
account      required      pam_nologin.so
# pam_selinux.so close should be the first session rule
session      required      pam_selinux.so close
session      required      pam_loginuid.so
session      optional      pam_console.so
# pam_selinux.so open should only be followed by sessions to be executed in
the user context
session      required      pam_selinux.so open
session      required      pam_namespace.so
session      optional      pam_keyinit.so force revoke
```

Обязательными модулями здесь являются: `pam_nologin`, `pam_selinux`, `pam_loginuid`, `pam_selinux`, `pam_namespace`. Выполнение аутентификации в этих модулях гарантируют аутентификацию в целом.

Достоинства метода аутентификации РАМ:

- Простота реализации.
- Расширяемость.
- Поддерживает клиент-серверную архитектуру.

Недостатки метода аутентификации РАМ:

- Возможность подмены модуля аутентификации.
- Необходима тонкая настройка аутентификации для использования в PostgreSQL.
- Может быть использован только для проверки пароля.

Подмену модуля РАМ сложно обнаружить. Поэтому при использовании РАМ-аутентификации требуется повысить безопасность компонентов библиотеки РАМ. Кроме того, использование данного метода аутентификации ограничено проверкой пароля пользователя, поэтому использовать его для сетевой аутентификации пользователей нежелательно.

1.6 LDAP

Данный метод аутентификации работает аналогично методам аутентификации по паролю, за исключением того, что используется LDAP для проверки подлинности. При этом серверу LDAP высылается имя пользователя и его пароль, после чего он выполняет поиск записи по этим данным. Если такая запись найдена, то пользователю разрешается доступ в базу данных, иначе — запрещается.

Метод аутентификации LDAP может работать в двух режимах.

В первом режиме сервер PostgreSQL создает запрос записи на сервер LDAP на основе логина пользователя и параметров подключения в файле `pg_hba.conf`.

Во втором режиме сервер PostgreSQL подключается к серверу LDAP с помощью фиксированного имени и пароля, указанного с помощью атрибутов `ldapbinddn` и `ldapbindpasswd` в файле `pg_hba.conf`. После чего выполняется поиск записи по имени пользователя базы данных и его пароля в поддереве `ldapbasedn`. Если такая запись найдена, то подключение к базе данных разрешается.

Достоинства метода аутентификации LDAP:

- Централизованное хранение аккаунтов пользователей, что не требует создания локальных пользователей ОС.
- Поддерживает клиент-серверную архитектуру.
- Расширяемость.
- Шифрованное соединение по протоколу TLS.

Недостатки метода аутентификации LDAP:

- Требуется наличия постоянно работающего сервера LDAP для аутентификации пользователей.

Количество серверов LDAP при возрастающей нагрузке может быть расширено за счет возможности репликации данных каталога. Между клиентами и серверами LDAP может быть настроено соединение по протоколу TLS, что делает данный метод аутентификации наиболее безопасным среди рассмотренных выше.

1.7 Выбор метода аутентификации пользователей в распределенной многопользовательской системе

При построении распределенной информационной системы одним из важных вопросов является аутентификация пользователей в данной системе.

Пусть информационная система представляет из себя 2 машины, одна из которых является сервером PostgreSQL, другая — клиент, который имеет возможность удаленно подключаться к ней.

Использовать метод аутентификации **Trust** для сетевой аутентификации пользователей PostgreSQL небезопасно, т.к. данный метод разрешает подключение всем пользователям к любой базе данных. Единственным возможным ограничением доступа является диапазон IP-адресов. Кроме того, соединение между клиентом и сервером не является шифрованным.

Методы, основанные на шифровании пароля также небезопасны, т.к. пароль шифруется с помощью простых схем шифрований.

Метод **Ident** требует создания локального пользователя для сопоставления его имени с именем пользователя базы данных, кроме того, данный протокол не предназначен для аутентификации или контроля доступа.

Метод **Peer** может быть использован только для аутентификации локальных клиентов.

Использовать **PAM** в качестве основного метода аутентификации затруднено ввиду необходимости длительной настройки и тестирования соединения. Кроме того, при использовании данного метода существует проблема подмены **PAM**-модуля и сложности обнаружения этой уязвимости.

Использование **LDAP** для аутентификации клиентов позволяет решить все перечисленные проблемы:

- Сервер **LDAP** можно использовать для аутентификации не только клиентов баз данных, но и пользователей системы. Таким образом нет необходимости создавать локальных пользователей на клиентских машинах.
- Написание политики доступа позволит предотвратить неконтролируемый доступ к информации пользователей.
- Сервер **LDAP** позволяет создавать иерархические структуры, т.о. разделять права доступа к информации.
- Шифрование соединения по протоколу **TLS**.
- Расширяемость. При увеличении нагрузки на сервер **LDAP** применяется репликация данных между серверами. Кроме того, существует возможность хранения поддеревьев информации на отдельной машине для более быстрого доступа к ней.

Поэтому предлагается использовать **LDAP** в качестве основного метода аутентификации в PostgreSQL в данной системе.

2 Метод аутентификации LDAP

2.1 Обзор протокола LDAP

LDAP - это аббревиатура от *Lightweight Directory Access Protocol* [6]. Как следует из названия, это облегченный протокол доступа к службам каталогов, предназначенный для доступа к службам каталогов на основе X.500 [7]. LDAP работает поверх TCP/IP или других ориентированных на соединение сетевых протоколов. LDAP стандартизирован в качестве протокола IETF [8]. Он описан в стандарте **RFC4510** [9].

Каталоги LDAP используют модель данных, которая считает или представляет данные как иерархию объектов. Это не означает, что LDAP является объектно-ориентированной базой данных.

В LDAP-каталоге данные представлены как иерархия записей. Полученная в результате древовидная структура называется **информационным деревом каталога** (*Data Information Tree, DIT*). Верхнюю часть данного дерева обычно называют **корнем** (**root**), (а также базой (**base**) или суффиксом (**suffix**)).

Каждый элемент DIT называется записью (**object**). Каждая запись имеет ноль или более дочерних записей. Каждая дочерняя запись является одноуровневой (братской) по отношению к другим дочерним записям своей родительской записи. Каждая запись является экземпляром одного или нескольких объектных классов (**objectClass**).

Объектные классы содержат ноль или более атрибутов (**attribute**). Атрибуты имеют имена (и, иногда, аббревиатуры или псевдонимы) и обычно содержат данные.

Характеристики (свойства) объектных классов и их атрибутов описываются определениями **ASN.1.100** [10].

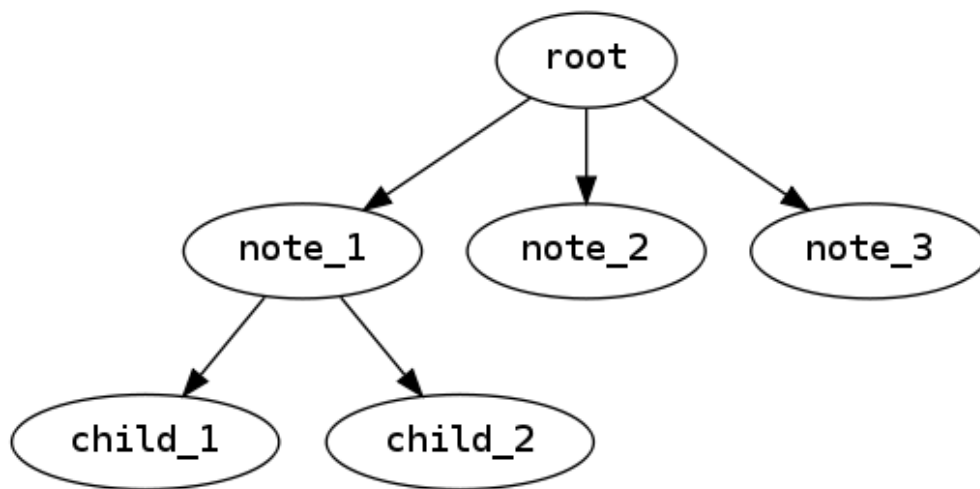


Рис. 2.1: Дерево информационного каталога LDAP

На *рис. 2.1* представлен пример дерева информационного каталога LDAP. Элемент **root** является корневым для данного дерева, а также является родителем для элементов **note_1**, **note_2**, **note_3**. Элементы **note_1**, **note_2** и **note_3** являются братскими между собой. Элементы **child_1** и **child_2** — дочерними для **note_1**.

Атрибуты

Каждый атрибут имеет имя и обычно содержит данные. Атрибуты всегда связаны являются членами одного или нескольких классов. У атрибутов есть ряд интересных особенностей:

1. Каждый атрибут определяет тип данных, которые он может содержать.
2. Атрибуты могут быть необязательными (MAY) или обязательными (MUST).
3. Атрибуты могут быть единственными (**single**) или множественными (**multi**). **Single** означает, что для атрибута может быть задано только одно значение. **Multi** — для атрибута может быть задано несколько значений.
4. Атрибуты могут иметь псевдонимы (**alias**) имен. При этом при ссылке на этот атрибут можно использовать как полное имя атрибута, так и его псевдоним.
5. На всех уровнях иерархии атрибуты должны быть уникальными.

Каждый атрибут описывается в классе по следующей схеме:

```
AttributeTypeDescription = "(" whsp      ;whsp - пробел
    numericoid whsp
    [ "NAME" qdescrs ]
    [ "DESC" qdstring ]
    [ "OBSOLETE" whsp ]
    [ "SUP" woid ]
    [ "EQUALITY" woid
    [ "ORDERING" woid
    [ "SUBSTR" woid ]
    [ "SYNTAX" whsp noidlen whsp ]
    [ "SINGLE-VALUE" whsp ]
    [ "COLLECTIVE" whsp ]
    [ "NO-USER-MODIFICATION" whsp ]
    [ X-ORDERED whsp type ]
    [ "USAGE" whsp AttributeUsage ]
    whsp ")"
```

Далее приводится описание наиболее важных элементов атрибута:

- **numericoid** определяет OID атрибута. Этот OID должен быть уникальным.
- **NAME** определяет глобально уникальное имя для данного атрибута.
- **DESC** задает описание атрибута.
- **SUP** указывает класс-родитель (если есть).
- **EQUALITY** определяет поведение атрибута в поисковом фильтре, где не используется поисковых шаблонов.
- **SUBSTR** определяет поведение атрибута в поисковом фильтре, использующем подстроки, содержащем один или несколько поисковых шаблонов.
- **SYNTAX** это OID, определяющий тип данных, а также какие правила (проверки данных) применяются к этим данным.

На листинге приведен пример определения атрибута `name`:

```
attributetype ( 2.5.4.41 NAME 'name'
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )
```

Объектные классы

Объектные классы представляют собой хранилище атрибутов, определяют перечень атрибутов, которые обязательно должны присутствовать в случае загрузки записи в LDAP, а также перечень необязательных атрибутов. Объектный класс может быть унаследован от другого класса, при этом дочерний класс наследует все атрибуты класса-родителя.

Объектный класс описывается по следующей схеме:

```
ObjectClassDescription = "(" whsp
    numericoid whsp
    [ "NAME" qdescrs ]
    [ "DESC" qdstring ]
    [ "OBSOLETE" whsp ]
    [ "SUP" oids ]
    [ ( "ABSTRACT" / "STRUCTURAL" / "AUXILIARY" ) whsp ]

    [ "MUST" oids ]
    [ "MAY" oids ]
    whsp ")"
```

- Поле `numerrricoid` указывает идентификатор класса.
- `NAME` определяет имя класса.
- `DESC` задает описание класса.
- `SUP` указывает класс-родитель данного класса.
- `STRUCTURAL` — класс может формировать запись в дереве каталогов. `ABSTRACT` — указывает на несуществующий класс в дереве каталогов, используемый для удобства. `AUXILIARY` определяет вспомогательный класс, который только используется при создании других объектных классов
- `MUST` определяет перечень атрибутов, которые обязательно должны содержаться в классе.
- `MAY` определяет перечень необязательных атрибутов класса.

В качестве примера далее приведено описание класса `pilotOrganization` с необязательным атрибутом `buildingName` и со всеми атрибутами (необязательными и обязательными) объектных классов `organization` и `organizationUnit`:

```
objectClasses: ( 0.9.2342.19200300.100.4.20 NAME 'pilotOrganization'
    SUP ( organization $ organizationalUnit ) STRUCTURAL
    MAY buildingName )
```

2.2 OpenLDAP

OpenLDAP — открытая реализация протокола LDAP.

slapd — сервер службы каталогов, реализующую 3-ю версию протокола LDAP. В качестве хранилища поддерживаются механизмы манипуляции данными. Одним из самых распространенных является BDB (высокопроизводительный механизм манипуляции с поддержкой транзакций) на базе Berkley DB [11]. Сервер **slapd** может быть настроен в режиме **master-slave**, в котором данные ведущего сервера пересылаются в фоновом режиме на ведомый.

2.2.1 Использование динамической конфигурации сервера OpenLDAP

Исторически OpenLDAP настраивался статически, то есть, чтобы произвести изменение, нужно было править конфигурационный файл **slapd.conf**, а затем останавливать и запускать **slapd**. При увеличении количества записей в каталоге перезапуск будет занимать все более продолжительное время, и, в случае высоконагруженных каталогов, такой метод переконфигурации становится все более неприемлемым. В OpenLDAP версии 2.3 была представлена новая возможность, посредством которой конфигурация может быть произведена во время исполнения с помощью модификации записей специального DIT, называемого **cn=config**.

Все конфигурационные файлы сервера OpenLDAP хранятся в каталоге **/etc/openldap/slapd.d**.

Для управления настройками рабочей среды в конфигурации времени исполнения (OLC) используется конфигурационное DIT с жестко установленным суффиксом **cn=config**. Концептуально, модификация записей в этом DIT (с помощью LDAP-браузера или файлов LDIF¹) приводит к немедленному изменению поведения рабочей среды **slapd** без необходимости перезапуска **slapd** (как это приходилось делать после изменения **slapd.conf**). На рисунке 2.2 представлена структурная схема каталога **cn=config**.

¹LDIF (*LDAP Data Interchange Format*, «формат обмена данными LDAP») — формат представления записей службы каталогов или их изменений в текстовом виде.

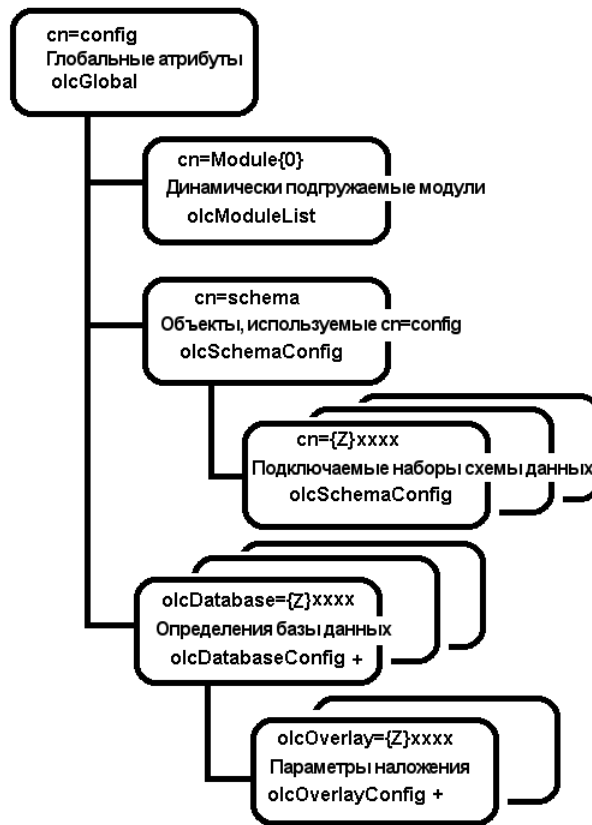


Рис. 2.2: Структурная схема каталога `cn=config` OpenLDAP

Корневая запись `cn=config` основывается на объектном классе `olcGlobal` и содержит глобальные атрибуты, влияющие на работу `slapd`.

Запись `cn=module{0},cn=config` основывается на объектном классе `olcModuleList` и содержит список всех динамически загружаемых объектов и путей к ним.

Запись `cn=schema,cn=config` использует объектный класс `olcSchemaConfig` и содержит все объекты, используемые системой `cn=config`.

Дочерние записи `cn={Z}xxx,cn=schema,cn=config` используют объектный класс `olcSchemaConfig` и содержат все объекты (атрибуты и объектные классы), определенные в данном наборе схемы. Здесь `{Z}` - числовой счетчик, нумерация которого начинается с 0, а `xxxx` - имя набора схемы данных. То есть, если первым подключается набор схемы `core.schema`, то данное значение будет `{0}core`.

Запись `olcDatabase={Z}xxx,cn=config` использует объектный класс `olcDatabaseConfig` (содержащий общие атрибуты, определяемые для всех типов баз данных), а также объектный класс, специфичный для конкретного типа базы данных. Здесь `{Z}` - числовой счетчик, нумерация которого начинается с 0, а `xxxx` - имя типа базы данных. То есть, если первой определяется база данных типа `bdb`, то данное значение будет `{0}bdb`, и эта запись будет строиться на объектном классе `olcBdbConfig`, содержащем атрибуты, специфичные для этого типа баз данных.

Запись `olcOverlay={Z}xxx,olcDatabase={Z}xxx,cn=config` использует объектный класс `olcOverlayConfig`, содержащий общие атрибуты, определяемые для всех наложений, а также объектный класс, специфичный для конкретного наложения. Здесь `{Z}` - числовой счетчик, нумерация которого начинается с 0, а `xxxx` - имя наложения. То есть, если первым из наложе-

ний определяется наложением `syncprov`, то данное значение будет `{0}syncprov`, и эта запись будет строиться на объектном классе `olcSyncProvConfig`, содержащем атрибуты, специфичные для этого наложения. Точно также другие наложения будут использовать специфичные для них объектные классы, содержащие их уникальные атрибуты.

2.2.2 Инструменты манипуляций данными в OpenLDAP

С пакетом `openldap-servers` [12] поставляются различные утилиты OpenLDAP, которые позволяют оперировать с данными каталога.

Утилита `ldapadd` (`ldapmodify`) позволяет добавить данные в каталог. Например запрос:

```
# ldapadd -H ldapi:/// -x -D "cn=admin,dc=ldap-server,dc=ru" -f /tmp/
addgroups.ldif -w password
```

добавляет в каталог `cn=admin,dc=ldap-server,dc=ru` содержимое файла `/tmp/addgroups.ldif` локально с применением проверки подлинности по паролю (указан после ключа `-w`).

`ldapdelete` открывает соединение с сервером LDAP, подключается и удаляет одну или несколько записей. Если команде был предоставлен один или несколько аргументов DN, записи с этими Distinguished Names будут удалены. Каждый DN должен быть оформлен в виде строкового представления LDAPv3, как определено в RFC 2253.

`ldapmodrdn` открывает соединение с сервером LDAP, подключается и модифицирует RDN записей.

Утилита `ldappasswd` позволяет проводить действия с паролями, в т.ч. генерировать зашифрованные пароли. Например запрос:

```
# ldappasswd -H ldapi:/// -D cn=admin,dc=ldap-server,dc=ru -W -A -S "cn=user1
,ou=Users,dc=ldap-server,dc=ru"
```

меняет пароль у пользователя `user1` из группы `Users`. При этом запрашивается старый пароль, новый пароль, после чего новый пароль записывается в каталог.

`ldapsearch` открывает соединение с сервером LDAP, подключается и производит поиск, используя указанные параметры. Указываемый фильтр должен соответствовать строковому представлению поисковых фильтров, определенному в RFC4515 [13]. Если фильтр не был указан, используется фильтр по умолчанию (`objectClass=*`). Если `ldapsearch` находит одну или несколько записей, возвращаются атрибуты, указанные как аргументы `attrs`. Если в списке атрибутов присутствует `*`, возвращаются все пользовательские атрибуты. Если присутствует `+`, возвращаются все операционные атрибуты. Если не было указано аргументов `attrs`, будут возвращены все пользовательские атрибуты (то есть подразумевается `*`). Например запрос:

```
# ldapsearch -x -LLL -b dc=ldap-server,dc=ru '(objectClass=inetOrgPerson)'
```

возвратит все записи класса `InetOrgPerson` из каталога `dc=ldap-server,dc=ru`.

`ldapwhoami` открывает соединение и получает информацию о пользователе, который подключается к `slapd`. Пример:

```
# ldapwhoami -x -D cn=postgres,ou=Admins,dc=ldap-server,dc=ru -w 123456
```

выведет строку:

```
cn=postgres,ou=Admins,dc=ldap-server,dc=ru
```

`slapacd` позволяет пользователю проверить возможность доступа от имени определенного DN подключения (`-b`) к указанным атрибутам, применяя текущие директивы политик, указанные в конфигурационных файлах `slapd` атрибутами `olcAccess`.

Например, запрос

```
# slapacd -b cn=user1,ou=Users,dc=ldap-server,dc=ru -D cn=user1,ou=Users,dc=ldap-server,dc=ru
```

Выдаст информацию, какие права имеет пользователь `cn=user1,ou=Users,dc=ldap-server,dc=ru` (указывается с помощью ключа `-D`) к своей записи (указана с помощью ключа `-b`):

```
authcDN: "cn=user1,ou=users,dc=ldap-server,dc=ru"
entry: write(=wrsxcd)
children: write(=wrsxcd)
cn=user1: write(=wrsxcd)
sn=seuser: write(=wrsxcd)
objectClass=seInetOrgPerson: write(=wrsxcd)
userPassword=****: write(=wrsxcd)
selinuxContext=user_u:user_r:user_t:s0-s3: write(=wrsxcd)
uid=501: write(=wrsxcd)
structuralObjectClass=seInetOrgPerson: write(=wrsxcd)
entryUUID=e5543424-df1a-1032-970f-83b11b1d9dcb: write(=wrsxcd)
creatorsName=cn=admin,dc=ldap-server,dc=ru: write(=wrsxcd)
createTimestamp=20131111124532Z: write(=wrsxcd)
entryCSN=20131111124532.017725Z#000000#000#000000: write(=wrsxcd)
modifiersName=cn=admin,dc=ldap-server,dc=ru: write(=wrsxcd)
modifyTimestamp=20131111124532Z: write(=wrsxcd)
```

В выводе команды представлен `authcDN` — пользователь, от имени которого осуществлен вход в дерево каталогов, а далее представлены права на доступ к записи в целом (`entity`), права на унаследованные записи (`child`) и поля (`cn=user1`, `sn=seuser` и т.д.). В текущей версии политики пользователь может иметь право на запись только к своей записи.

Однако если ввести запрос:

```
# slapacd -b cn=user1,ou=Users,dc=ldap-server,dc=ru -D cn=user2,ou=Users,dc=ldap-server,dc=ru
```

то будет выведено следующая информация:

```
authcDN: "cn=user2,ou=users,dc=ldap-server,dc=ru"
entry: read(=rscxd)
children: read(=rscxd)
cn=user1: read(=rscxd)
sn=seuser: read(=rscxd)
objectClass=seInetOrgPerson: read(=rscxd)
userPassword=****: read(=rscxd)
selinuxContext=user_u:user_r:user_t:s0-s3: read(=rscxd)
uid=501: read(=rscxd)
structuralObjectClass=seInetOrgPerson: read(=rscxd)
entryUUID=e5543424-df1a-1032-970f-83b11b1d9dcb: read(=rscxd)
creatorsName=cn=admin,dc=ldap-server,dc=ru: read(=rscxd)
createTimestamp=20131111124532Z: read(=rscxd)
entryCSN=20131111124532.017725Z#000000#000#000000: read(=rscxd)
modifiersName=cn=admin,dc=ldap-server,dc=ru: read(=rscxd)
modifyTimestamp=20131111124532Z: read(=rscxd)
```

Т.е. в текущей политике любой пользователь, не являющийся владельцем записи, имеет права только на чтение.

slapadd используется для добавления записей, указанных в формате LDIF в базу данных LDAP. Команда применяет LDIF к базе данных, определяемой по номеру базы данных или суффиксу. Входной LDIF считывается со стандартного входа, либо из указанного файла.

slapcat используется для генерации LDIF, основанных на содержимом базы данных LDAP. Она открывает базу данных, определяемую по номеру базы данных или суффиксу, и пишет соответствующий LDIF на стандартный вывод или в указанный файл.

Например:

```
# slapcat -a "((entryDN:dc=ldap-server,dc=ru))"
```

создаст дамп LDAP всего DN `dc=ldap-server,dc=ru`.

slappaswd необходим для генерации зашифрованного пароля `rootDN` и использования в LDIF. Например:

```
# slappaswd -s secret -h {MD5}
```

создаст хэш MD5 от пароля `secret`

slaptest используется для проверки конфигурационных файлов LDAP, а также может быть использован для конвертации конфигурационных файлов в LDIF.

3 Настройка PostgreSQL с аутентификацией LDAP

Для демонстрации работы метода аутентификации LDAP в PostgreSQL предлагается развернуть стенд, состоящий из трех машин (см. рис. 3.1). На каждой машине установлена операционная система Centos 6.5. На сервере LDAP установлен OpenLDAP 2.4.23, на сервере PostgreSQL используется СУБД PostgreSQL версии 9.3.

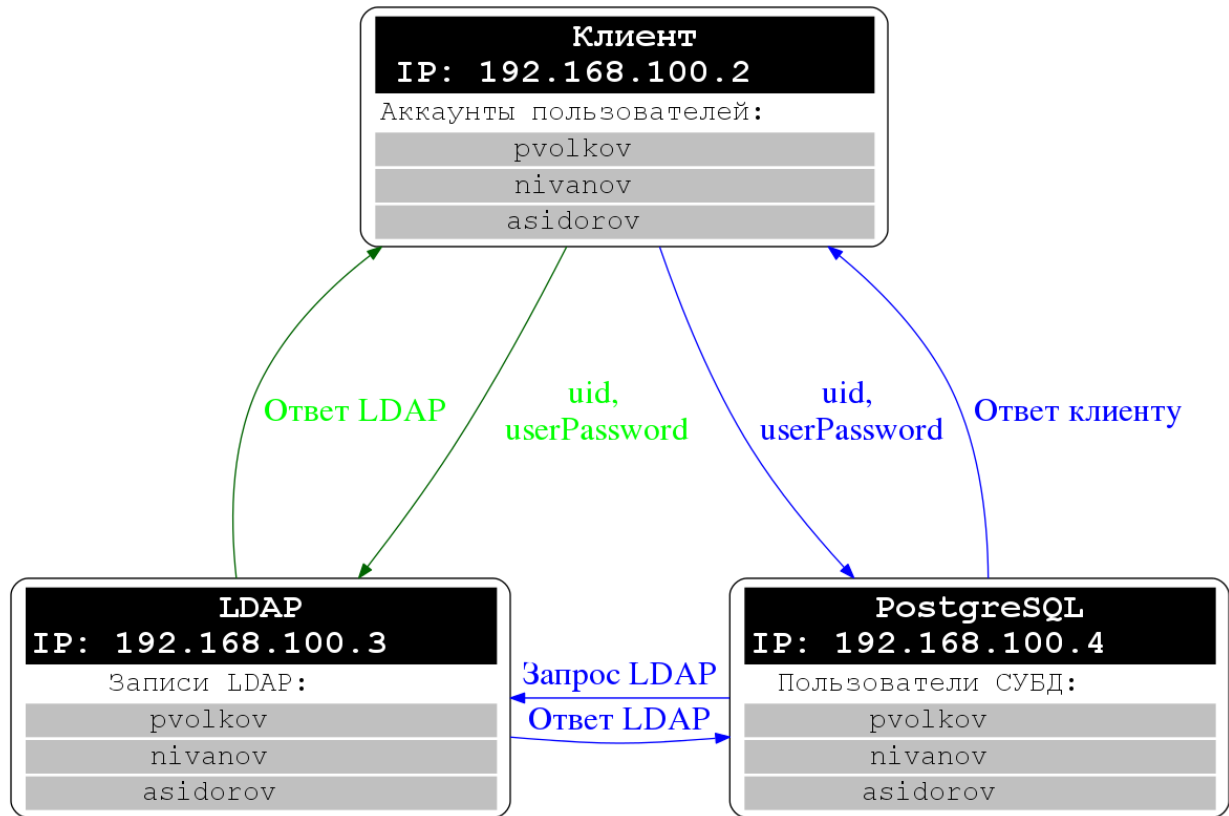


Рис. 3.1: Схема разворачиваемого стенда

Сервер LDAP (IP 192.168.100.3) хранит записи пользователей (pvolkov, nivanov и asidorov) и проводит аутентификацию этих пользователей на клиенте (IP 192.168.100.2).

Сервер базы данных PostgreSQL (IP 192.168.100.4) содержит администратора базы данных postgres, который является пользователем операционной системы, а также пользователей базы данных pvolkov, nivanov и asidorov.

Принцип работы стенда следующий:

1. Пользователь вводит имя своего аккаунта на клиентской машине и пароль. Пароль отправляется LDAP-серверу в закрытом виде и проверяется на совпадение.
2. Если переданный и хранимый пароль совпадают, пользователь успешно проходит аутентификацию на клиенте.
3. При подключении к базе данных с помощью клиента psql, пользователь передает свой uid (логин пользователя) и пароль в PostgreSQL.
4. PostgreSQL генерирует запрос на основе данных клиента.

5. LDAP-сервер выполняет поиск записи и сверяет пароль. Результат поиска отправляется PostgreSQL.
6. Если такой записи нет или пароль неверный, в подключении к базе данных отказывается.

3.1 Настройка машин стенда

3.1.1 Настройка сервера LDAP

Первоначальная настройка сервера OpenLDAP предполагает установку пакетов сервера OpenLDAP, далее изменение доменного имени, создание администратора каталога и его пароля, изменение правил доступа к атрибутам каталога. С помощью системы **rsyslog** выполняется логгирование сервера.

Сервер OpenLDAP устанавливается с помощью команды:

```
# yum install -y openldap-{clients,servers}
```

Команда установит необходимые компоненты сервера, а также клиентские приложения для выполнения запросов в службу каталогов LDAP.

Сервер OpenLDAP работает на порте 389. Для возможности доступа к серверу OpenLDAP с клиентской машины требуется открыть порт 389 в межсетевом экране **iptables**. Для этого нужно записать следующую строку

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 389 -j ACCEPT
```

в конфигурационный файл **/etc/sysconfig/iptables** после строк, заканчивающихся на **ACCEPT** и до строки **REJECT**.

Изменение конфигурационного файла межсетевого экрана требует его перезапуска:

```
# service iptables restart
```

Далее выполняется непосредственно настройка сервера OpenLDAP.

Сервер OpenLDAP использует для хранения BerkleyDB. Файл **DB_CONFIG.example** содержит конфигурацию BerkleyDB, оптимизированную для использования в OpenLDAP. Поэтому требуется скопировать этот файл в каталог базы данных LDAP:

```
# cp /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/DB_CONFIG
```

и назначить папке владельца **ldap.ldap**:

```
# chown -R ldap.ldap /var/lib/ldap
```

При установке сервера OpenLDAP создается пользователь **ldap**, не наделенный правами аутентификации на сервере. Это сделано из соображения безопасности. При этом нужно следить, чтобы конфигурационные файлы сервера **/etc/openldap/slapd.d** и файлы базы данных BerkleyDB **/var/lib/ldap** были маркированы как **ldap.ldap**, иначе сервер OpenLDAP не будет запущен.

Для хранения изменений конфигурационных файлов и пользовательских данных сервера в формате LDAP Data Interchange Format (LDIF, «Формат обмена данными LDAP») создается папка **ldif**:

```
# mkdir /etc/openldap/ldif
```

Для изменения доменного имени создается файл **/etc/openldap/ldif/root.ldif**:

```

1 dn: olcDatabase={2}bdb,cn=config
2 changetype: modify
3 replace: olcSuffix
4 olcSuffix: dc=ldap-server,dc=ru

```

В первой строке указывается путь до конфигурационного файла базы данных: `olcDatabase={2}bdb.ldif`, который содержится в папке `cn=config`. Далее указывается, что файл модифицируется. В третьей строке указывается, что изменяется параметр `olcSuffix`, отвечающий за имя DIT. В последней строке устанавливается желаемое имя информационного каталога DIT в нотации LDAP.

Для загрузки этого файла в OpenLDAP используется следующая команда:

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f /etc/openldap/ldif/root.ldif
```

Ключом `-Y` передается тип механизма SASL, используемый при запросе. В данном случае используется аутентификация, уже выполненную протоколом низкого уровня.

Ключом `-H` указывается хост LDAP-сервера. В данном случае аргумент `ldapi:///` означает выполнение запроса на локальном хосте.

С помощью ключа `-f` указывается файл запроса.

При успешном выполнении команды будет выведено следующее:

```

SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "olcDatabase={2}bdb,cn=config"

```

Администратор LDAP может изменять сущности каталога, добавлять или удалять сущности пользователей, а также редактировать их сущности. Создание сущности администратора в LDAP предполагает:

1. Создание записи администратора.
2. Установка пароля администратора.

Перед тем, как создать LDIF-файл для администратора, нужно создать зашифрованный пароль с помощью утилиты `slappaswd`:

```
# slappaswd
```

Потребуется ввести желаемый пароль, а также его подтвердить. В результате работы этой команды будет выведена строка с хэшем пароля. Необходимость шифрования пароля обусловлена недопущением раскрытия пароля администратора службы каталогов.

Запись администратора каталога создается в файле `/etc/openldap/ldif/admin.ldif`:

```

dn: olcDatabase={2}bdb,cn=config
changetype: modify
replace: olcRootDN
olcRootDN: cn=admin,dc=ldap-server,dc=ru

dn: olcDatabase={2}bdb,cn=config
changetype: modify
add: olcRootPW
olcRootPW: {SSHA}US0VGNxhxro/QD3B4wIbjRa5re9i8cX1

dn: olcDatabase=cn=config
changetype: modify
replace: olcRootPW
olcRootPW: {SSHA}US0VGNxhxro/QD3B4wIbjRa5re9i8cX1

```

В поле `olcRootDN` устанавливается имя администратора `admin`.

В поле `olcRootPW` записывается пароль администратора, зашифрованный с помощью команды `slappaswd`.

В третьем запросе создается пароль администратора каталога `cn=config`. Установка пароля требуется, чтобы можно было проводить запросы на изменение конфигурации сервера OpenLDAP.

Аналогично предыдущему, загрузка осуществляется с помощью команды:

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f /etc/openldap/ldif/admin.ldif
```

ACL — это набор правил, определяющие права доступа к атрибутам записи LDAP. Порядок правил важен при создании, т.к. их обработка происходит от первого к последнему. Если пользователь запрашивает доступ к атрибуту, но в ACL не существует ни одного, в котором разрешается, то в доступе отказывается. В стандартной политике LDAP, поставляемой с пакетами `openldap` доступ к атрибутам разрешается всем пользователям. В связи с этим, требуется установка новых правил ACL.

Создается файл `/etc/openldap/ldif/policy.ldif`, определяющий права доступа к атрибутам записей LDAP:

```
dn: olcDatabase={2}bdb,cn=config
changetype: modify
replace: olcAccess
olcAccess: {0}to attrs=userPassword by self write by dn.base="cn=admin,dc=
ldap-server,dc=ru" write by anonymous auth by * none

dn: olcDatabase={2}bdb,cn=config
changetype: modify
add: olcAccess
olcAccess: {1}to * by dn.base="cn=admin,dc=ldap-server,dc=ru" write by self
write by * read

dn: olcDatabase={1}monitor,cn=config
changetype: modify
replace: olcAccess
olcAccess: {0}to * by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=
external,cn=auth" read by dn.base="cn=admin,dc=ldap-server,dc=ru" read by * none
```

Первый запрос устанавливает права пользователей на поле `userPassword` записей пользователей. В данное поле разрешено записывать (`write`), т.е. изменять его, себе (`self`) и администратору (`cn=admin,dc=ldap-server,dc=ru`). Анонимные пользователи для изменения этого поля должны пройти аутентификацию в LDAP (`anonymous auth`), всем остальным, не попадающие под предыдущее, запрещены любые действия (`by * none`).

Во втором запросе устанавливаются права пользователей на остальные поля записей.

В третьем запросе устанавливаются права на чтение записей базы `cn=monitor`, с помощью которой можно узнать текущее состояние каталога.

Загрузка политик ACL в LDAP:

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f /etc/openldap/ldif/policy.ldif
```

Система ведения логов позволяет вести в реальном времени информацию о работе демона `slapd`, в том числе, выводить ошибки. Система ведения логов демона в OpenLDAP реализуется с помощью системного демона `rsyslog`. Для его настройки выполняются следующие действия:

1. Создать файл `/var/log/openldap.log` — файл логов сервера LDAP:


```
# touch /var/log/openldap.log
```

2. Назначить владельца логов `ldap.ldap`:

```
# chown ldap.ldap /var/log/openldap.log
```

3. Установить права на чтение и запись пользователю `ldap` на лог-файл:

```
# chmod +rw /var/log/openldap.log
```

4. Добавить следующую строку, позволяющую демону системы ведения логов `rsyslog` перенаправлять все сообщения LDAP-сервера в созданный ранее файл:

```
local4.*          -/var/log/openldap.log
```

5. Изменение конфигурационного файла демона `rsyslog` требует его перезапуска:

```
# service rsyslog restart
```

Изменяются параметры конфигурации клиента OpenLDAP `/etc/openldap/ldap.conf`, в котором указываются база данных для подключения по-умолчанию (параметром `BASE`) и адрес LDAP-сервера (`URI`), указан локальный сервер `ldap:///`:

```
URI ldap:///
BASE dc=ldap-server,dc=ru
```

Аутентификация пользователей клиента требует постоянно работающего сервера LDAP, поэтому требуется добавить его в список автозагрузки:

```
# chkconfig slapd on
```

После чего перезагрузить сервер OpenLDAP:

```
# service slapd restart
```

3.1.2 Создание пользовательских аккаунтов

Создание пользовательских аккаунтов происходит на сервере LDAP.

Сначала создается корневая сущность каталога `/etc/openldap/base.ldif`. В данном файле описывается запись корня, сущность групп (`ou=Group`), которая хранит группы пользователей, а также группа пользователей (`ou=People`)

```
dn: dc=ldap-server,dc=ru
dc: ldap-server
objectClass: top
objectClass: domain
```

```
dn: ou=Group,dc=ldap-server,dc=ru
ou: Group
objectClass: top
objectClass: organizationalUnit
```

```
dn: ou=People,dc=ldap-server,dc=ru
ou: People
objectClass: top
objectClass: organizationalUnit
```

Для загрузки в каталог LDAP используется следующая команда:

```
# ldapadd -D cn=admin,dc=ldap-server,dc=ru -f /etc/openldap/ldif/base.ldif -W
```

При выполнении данной команды потребуется ввести пароль администратора каталога `admin`.

Для создания записи пользователя `pvolkov` требуется выполнить следующие действия:

1. Создается запись группы пользователя `pvolkov_group.ldif`:

```
dn: cn=pvolkov,ou=Group,dc=ldap-server,dc=ru
objectClass: posixGroup
objectClass: top
cn: pvolkov
userPassword: {SSHA}x
gidNumber: 500
```

Каждая запись группы содержит название группы (`cn`), поле `userPassword`, отвечающий за пароль, а также `gidNumber` — порядковый идентификатор группы в операционной системе². Каждая запись использует классы `posixGroup` и `top`.

2. Командой `ldappasswd` шифруется пароль пользователя.

3. Создается файл `/etc/openldap/ldif/pvolkov.ldif` следующего содержания:

```
dn: uid=pvolkov,ou=People,dc=ldap-server,dc=ru
uid: pvolkov
cn: Pavel Volkov
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword: {SSHA}US0VGNxhxro/QD3B4wIbjRa5re9i8cX1
shadowLastChange: 15997
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 500
gidNumber: 500
homeDirectory: /home/pvolkov
```

Каждая запись использует классы `account`, `posixAccount`, `top` и `shadowAccount`. В поле `userPassword` хранится пароль пользователя, зашифрованный командой `ldappasswd` на предыдущем шаге. Атрибуты `shadow` являются системными и хранят информацию для возможности смены пароля. Атрибут `loginShell` определяет интерпретатор командной строки, который будет использовать пользователь. `uidNumber` и `gidNumber` определяют идентификаторы пользователя и группы пользователя соответственно. Атрибут `homeDirectory` хранит путь к домашнему каталогу пользователя.

4. Для загрузки записей группы и пользователя используются команды:

```
# ldapadd -D cn=admin,dc=ldap-server,dc=ru -f /etc/openldap/ldif/
pvolkov_group.ldif -W
# ldapadd -D cn=admin,dc=ldap-server,dc=ru -f /etc/openldap/ldif/pvolkov
.ldif -W
```

²Диапазон `uid` пользователей в CentOS 6 начинается с 500

Аналогичные действия проводятся для пользователей `nivanov`, `asidorov`.

3.1.3 Создание сертификата сервера LDAP

Сервер LDAP пересылает всю информацию, включая пароли по сети в нешифрованном виде. Эта проблема решается с помощью TLS (*Transport Socket Layers* — «уровень защищенных сокетов»). TLS — криптографические протоколы, обеспечивающие защищенную передачу данных между узлами сети.

LDAP поддерживает работу TLS на порте 389.

Установка соединения с использованием TLS включает в себя следующие шаги [14]:

1. Клиент подключается к TLS — поддерживаемому серверу и запрашивает защищенное соединение.
2. Клиент предоставляет список поддерживаемых алгоритмов шифрования и хеш-функций.
3. Сервер выбирает из списка, предоставленного клиентом, наиболее устойчивые алгоритмы, которые так же поддерживаются сервером, и сообщает о своем выборе клиенту.
4. Сервер отправляет клиенту цифровой сертификат для собственной идентификации. Обычно цифровой сертификат содержит имя сервера, имя доверенного центра сертификации и открытый ключ сервера.
5. Клиент может связаться с сервером доверенного центра сертификации и подтвердить аутентичность переданного сертификата до начала передачи данных.
6. Для того чтобы сгенерировать ключ сессии для защищенного соединения, клиент шифрует случайно сгенерированную цифровую последовательность открытым ключом сервера и посылает результат на сервер. Учитывая специфику алгоритма асимметричного шифрования, используемого для установления соединения, только сервер может расшифровать полученную последовательность, используя свой закрытый ключ.

Создаются папки, которые будут хранить ключ сервера и сертификат сервера, после чего выполняется переход в созданную папку:

```
# mkdir /etc/openldap/certs/  
# mkdir /etc/openldap/certs/keys  
# cd /etc/openldap/certs
```

На машине сервера создается удостоверяющий центр:

```
# /etc/pki/tls/misc/CA -newca
```

При создании удостоверяющего центра должна быть введена информация о расположении удостоверяющего центра, организации и подразделения, а также пользователи или о хосте машины, от имени которого создается удостоверяющий центр:

```
Enter PEM pass phrase: <пароль>  
Verifying - Enter PEM pass phrase: <пароль>  
...  
Country Name (2 letter code) [XX]:ru  
State or Province Name (full name) []:msk  
Locality Name (eg, city) [Default City]:msk
```

```
Organization Name (eg, company) [Default Company Ltd]:mephi
Organizational Unit Name (eg, section) []:kaf36
Common Name (eg, your name or your server's hostname) []:root
Email Address []:root@kaf36
```

В результате выполнения этой команды будет создан удостоверяющий центр с публичным ключом `cacert.pem`. Удоверяющий центр гарантирует, что обладатель подписанного сертификата является доверенным субъектом в сети.

Создается закрытый ключ LDAP-сервера и заявка на подпись сертификата (*Certificate signing request*):

```
# openssl req -new -nodes -subj '/CN=ldap-server/O=mephi/C=ru/ST=msk/L=msk' -
keyout ldapkey.pem -out ldapcert.csr
```

В данной команде явно указывается будущий владелец ключа с помощью ключа `-subj`. После чего удостоверяющий центр выпускает сертификат по заявке:

```
# openssl ca -out ldapcert.crt -infiles ldapcert.csr
```

Сертификат сгенерирован в папке `/etc/openldap/certs/`, созданный ключ находится в папке `/etc/openldap/certs/keys/`.

Устанавливается владелец ключей `ldap:ldap` и права на доступ для исключения возможности несанкционированного доступа:

```
# chown -R ldap:ldap /etc/openldap/certs/*
# chmod 0400 keys/ldapkey.pem
# chmod 0600 ldapcert.pem
```

Для того, чтобы OpenLDAP использовал ключи для шифрования соединения, требуется выполнить следующий запрос, добавляющий информацию о сгенерированных сертификатах (удостоверяющего центра и сервера) и ключа сервера:

```
dn: cn=config
changetype: modify
replace: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/openldap/cacerts/cacert.pem

dn: cn=config
changetype: modify
replace: olcTLSCertificateFile
olcTLSCertificateFile: /etc/openldap/certs/ldapcert.pem

dn: cn=config
changetype: modify
replace: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/openldap/certs/keys/ldapkey.pem
```

Загрузка изменений осуществляется с помощью команды:

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f /etc/openldap/ldif/tls.ldif
```

Добавить в `/etc/openldap/ldap.conf` строку:

```
TLS_REQCERT allow
```

После чего выполняется перезагрузка сервера LDAP:

```
# service slapd restart
```

3.1.4 Настройка клиентской машины

Настройка клиентской машины состоит в установке клиента `psql` и настройке аутентификации пользователей с использованием демона SSSD.

Для подключения репозитория PostgreSQL выполняется команда:

```
# yum install -y http://yum.postgresql.org/9.3/redhat/rhel-6-x86_64/pgdg-redhat93-9.3-1.noarch.rpm
```

После чего выполняется обновление данных пакетного менеджера:

```
# yum update
```

Следующей командой выполняется установка клиента `psql`:

```
# yum install -y postgresql93
```

SSSD (*System Security Services Daemon*) [15] позволяет обращаться к удаленным механизмам аутентификации, называемым поставщиками. Таким образом стирается граница между аутентификацией локального и сетевого доступа и допускается использование разных механизмов. Информацию о пользователях предоставляет база данных, называемая доменом, которая может служить источником данных для поставщика.

Для установки демона SSSD используется команда:

```
# yum install -y sssd
```

Добавить демон SSSD в автозагрузку:

```
# chkconfig sssd on
```

Следующая команда создает конфигурационный файл `/etc/sss/sss.conf`, с помощью которой настраивается аутентификация пользователей на сервере LDAP:

```
authconfig --updateall \
--enableldap \
--enableldapauth \
--ldapserver=ldap://192.168.100.3 \
--ldapbasedn=dc=ldap-server,dc=ru \
--enableldaptls \
--enableldapstarttls \
--enablesss \
--enablesssdauth \
--disablecachecreds
```

Данная команда устанавливает аутентификацию с использованием сервера LDAP, находящегося по адресу `ldap://192.168.100.3`. При этом используется соединение TLS при подключении к нему. Опции `--enablesss` и `--enablesssdauth` описывают использование SSSD-демона при аутентификации. Последняя опция указывает не сохранять данные уже авторизованных пользователей на клиентской машине в кэш. Таким образом, аутентификация клиента будет возможна только при работающем сервере LDAP.

Дополнительно в конфигурационный файл демона SSSD устанавливается опция

```
ldap_tls_reqcert = allow
```

3.1.5 Настройка сервера PostgreSQL

Установка и настройка репозитория PostgreSQL приводится в разделе 3.1.4.

С помощью команды:

```
# yum install postgresql93-server postgresql93
```

устанавливаются необходимые пакеты СУБД PostgreSQL. При этом в системе создается администратор базы данных — **postgres**.

Установить пароль пользователю **postgres**:

```
# passwd postgres
```

Инициализировать сервер базы данных:

```
# service postgresql93 initdb
```

Результатом выполнения данной команды является создание каталога конфигурационных файлов в `/var/lib/pgsql/9.3/data`.

Создается приватный ключ сервера и заявка на подпись сертификата PostgreSQL³:

```
$ cd /var/lib/pgsql/9.3/data
$ openssl req -new -nodes -subj '/CN=postgres/O=mephi/C=ru/ST=msk/L=msk' -
keyout ldapkey.pem -out ldapcert.csr
$ chmod 0600 server.key
```

Заявка копируется на удостоверяющий центр для выпуска сертификата PostgreSQL:

```
$ scp server.csr root@192.168.100.3:/etc/pki/tls/
```

Удостоверяющий центр выпускает сертификат, копируются сертификаты сервера PostgreSQL и удостоверяющего центра в каталог базы данных:

```
# openssl ca -out server.crt -infiles server.csr
```

Подписанный сертификат сервера и сертификат удостоверяющего центра копируются на сервер базы данных:

```
# scp server.crt postgres@192.168.100.4:/var/lib/pgsql/9.3/data/
# scp /etc/pki/CA/cacert.pem postgres@192.168.100.4:/var/lib/pgsql/9.3/data/
ca.crt
```

На сервере базы данных устанавливается владелец сертификатов и права:

```
# chown postgres.postgres /var/lib/pgsql/9.3/data/server.crt
# chown postgres.postgres /var/lib/pgsql/9.3/data/ca.crt
# chmod 0600 /var/lib/pgsql/9.3/data/server.crt
```

В конфигурационном файле `postgresql.conf` разрешается подключение с любого IP-адреса к серверу базы данных:

```
listen_addresses = '*'
```

Изменяются опции SSL-соединения:

```
ssl = on
ssl_cert_file = 'server.crt'
ssl_key_file = 'server.key'
ssl_ca_file = 'ca.crt'
```

Файл `pg_hba.conf` представляет собой таблицу со следующими полями:

```
TYPE DATABASE USER ADDRESS METHOD
```

³Выполнение команд с символом '\$' нужно выполнять от пользователя **postgres**

Поле **TYPE** устанавливает тип подключения (**local** — локальное, **host** — удаленное нешифрованное по протоколу TCP/IP и т.д.).

DATABASE указывает имена баз данных, к которым разрешено подключаться.

USER определяет пользователей, которые могут подключаться.

ADDRESS — IP-адреса, с которых возможно подключение.

METHOD — Метод аутентификации, используемый в подключении.

Для установки метода аутентификации **ldap** используется следующая строка [16]:

```
hostssl all all 192.168.100.0/24 ldap ldapserver=192.168.100.3 ldapprefix="
uid=" ldapsuffix=",ou=People,dc=ldap-server,dc=ru"
```

Она означает, что разрешено подключение ко всем базам данных всем пользователям, IP-адреса клиентов которых находятся в подсети 192.168.100.0 при успешной аутентификации по методу **ldap**. При этом подключение между клиентом и сервером будет зашифровано по протоколу SSL (на это указывает параметр **hostssl**).

В качестве аргументов метода аутентификации **ldap** указывается IP-адрес LDAP-сервера — 192.168.100.3, префикс **ldapprefix** — идентификатор записи в каталоге LDAP, а также суффикс **ldapsuffix** — адрес записи в каталоге.

СУБД PostgreSQL принимает по умолчанию подключения на порте 5432. Для возможности удаленного подключения требуется открыть этот порт 5432 в межсетевом экране **iptables**:

```
-A INPUT -m state --state NEW -m tcp -p tcp --sport 5432 -j ACCEPT
```

Межсетевой экран и сервер базы данных перезагружаются:

```
# service iptables restart
# service postgresql93 restart
```

Для подключения к базе данных с помощью учетных записей, хранящихся в LDAP, требуется создать пользователей базы данных:

```
$ psql -c "CREATE ROLE pvolkov login;"
$ psql -c "CREATE ROLE nivanov login;"
$ psql -c "CREATE ROLE asidorov login;"
```

С помощью данной команды разрешается подключение к базе данных через клиент **psql**.

Демон базы данных добавить в список автозагрузки:

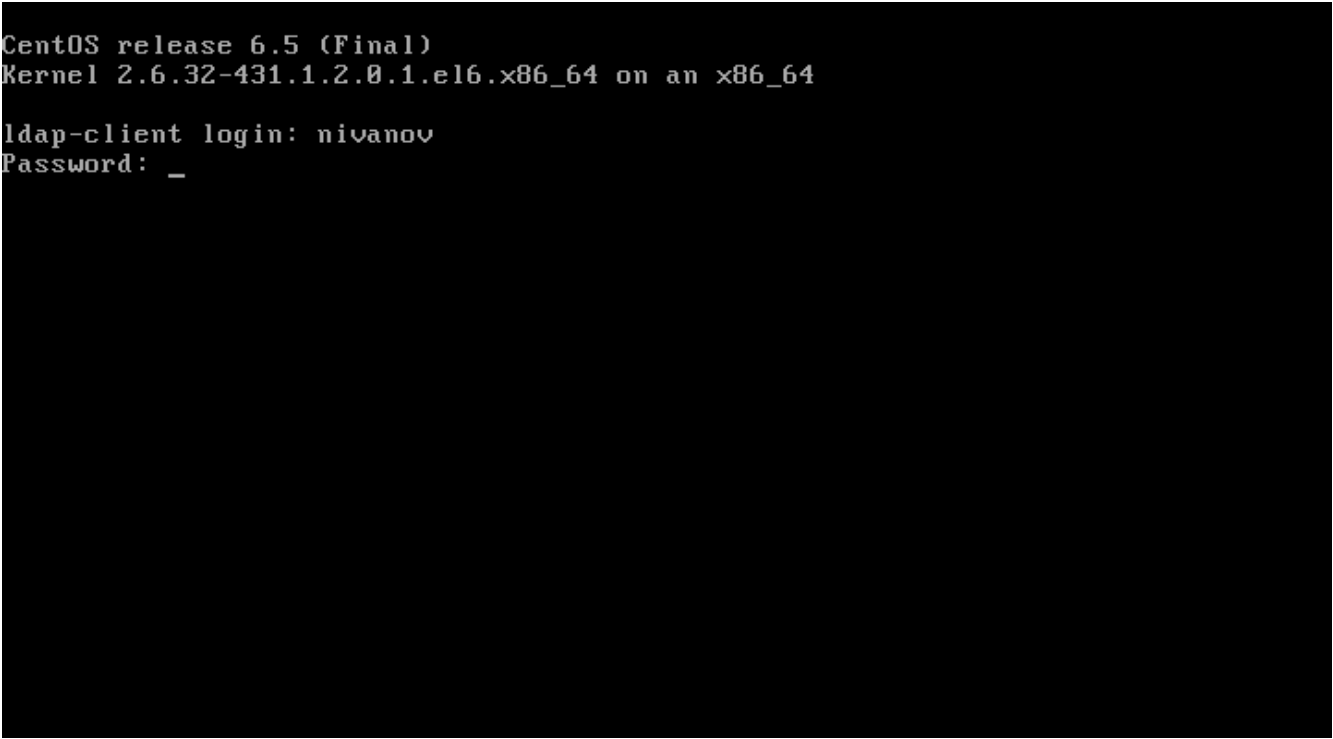
```
# chkconfig postgresql93 on
```

3.2 Проверка работы стенда

Для тестирования работы стенда создается тестовая база данных **testdb** на сервере PostgreSQL:

```
$ createdb testdb
```

Пусть пользователь хочет зайти в систему на клиенте как пользователь **nivanov**. Для этого ему нужно ввести свое имя в программу аутентификации **login** (рисунки 3.2):

A terminal window with a black background and white text. The text shows the CentOS release information and the ldap-client login process for user nivanov.

```
CentOS release 6.5 (Final)
Kernel 2.6.32-431.1.2.0.1.el6.x86_64 on an x86_64

ldap-client login: nivanov
Password: _
```

Рис. 3.2: Вход в систему на клиентской машине

После ввода пароля демон SSSD высылает запрос, содержащий логин пользователя и введенный пароль. Если запись найдена в каталоге, то пользователь войдет в систему.

Чтобы убедиться, что данные аккаунта берутся из сервера LDAP, выполняется команда:

```
$ getent passwd nivanov
```



```
CentOS release 6.5 (Final)
Kernel 2.6.32-431.1.2.0.1.el6.x86_64 on an x86_64

ldap-client login: nivanov
Password:
Last login: Mon Dec 23 13:46:48 on tty1
[nivanov@ldap-client ~]$ getent passwd nivanov
nivanov:*:501:501:Nikolay Ivanov:/home/nivanov:/bin/bash
[nivanov@ldap-client ~]$ _
```

Рис. 3.3: Вывод команды `getent`

Вывод данной команды представлен на *рисунке 3.3*.

Для того, чтобы проверить, что соединение между LDAP-сервером и клиентом зашифровано, устанавливается пакет `wireshark` на LDAP-сервере:

```
# yum install -y wireshark
```

После установки запускается утилита `tethereal`. Она подключается к интерфейсу `eth0` по протоколу TCP на порт `ldap`:

```
# tethereal -x -i eth0 tcp port ldap
```

Изначально утилита будет ожидать любого подключения по порту `ldap`. Как только пользователь введет свой пароль, утилита будет регистрировать сетевую активность между хостами `192.168.100.2` и `192.168.100.3` (*рисунок 3.4*):

```
ldapuser@ldap-server:/home/ldapuser
File Edit View Search Terminal Help
Running as user "root" and group "root". This could be dangerous.
Capturing on eth0
0.000000 192.168.100.2 -> 192.168.100.3 TLSv1 631 Application Data

0000 08 00 27 81 c5 71 08 00 27 13 36 e6 08 00 45 00  ..'..q..'6...E.
0010 02 69 c8 05 40 00 40 06 27 33 c0 a8 64 02 c0 a8  .i..@.@.'3..d...
0020 64 03 8e 73 02 7c ea 07 fa b4 61 87 e3 6b 80 18  d..s.|....a..k..
0030 0b 58 d0 d3 00 00 01 01 08 0a 00 28 16 cf 00 27  .X.....(...'
0040 cb 92 17 03 01 02 30 0d fb cd c3 ad 49 65 44 40  .....0.....IeD@
0050 a0 98 95 13 aa 55 9f f3 ff d0 ea c8 8a 1e 13 0c  .....U.....
0060 0f ce da f7 c6 81 d9 28 ab 6f fa 38 4e 46 9d b1  .....(.o.8NF..
0070 c7 84 8d dd 3e de 2f e3 71 c1 f3 27 ad 85 71 cd  ....>./..q..'..q.
0080 91 5e 95 ea 80 e2 f8 64 d3 c3 c7 3c e4 c9 66 e1  .^.....d...<..f.
0090 68 68 21 4b d9 60 31 7e 4f f5 4c ef d1 18 b2 0c  hh!K.`1~0.L.....
00a0 7f 84 ee a1 ab 31 a0 9f 49 06 83 1f fe ae 8c 62  ....1..I.....b
00b0 60 95 53 f5 5a ff ad cb 47 31 73 6e db 21 0c 95  `..S.Z...Glsn.!..
00c0 64 1e 95 db 24 fa 21 61 ea de 73 c2 41 b3 07 70  d...$.!a..s.A..p
00d0 39 b4 c8 fd 98 c9 57 13 ca 01 1d e4 df 6b 12 f3  9....W.....k..
00e0 ea 56 38 ad 7c f3 2f d3 10 10 ea 4d 92 6e d1 42  .V8.|./....M.n.B
00f0 bf a4 b1 09 68 04 bf d0 a4 f0 54 31 36 8b bf bd  ....h.....T16...
0100 21 fd b7 e8 a2 6b 76 b5 7c df 84 66 33 22 2e 72  !....kv.|..f3".r
0110 cb 17 0f 00 12 30 0d bc 11 4a fc df b9 97 93 cd  ....0...J.....
0120 f0 6b 86 13 da 77 53 98 0d 0b 6f c7 fd 05 44 71  .k...wS...o...Dq
0130 b7 9a 66 ab 7a a8 43 8e 62 dc c8 d4 40 67 33 06  ..f.z.C.b...@g3.
0140 61 fa cc 49 bc f1 6e 50 c7 d6 d1 0d 67 4f c9 1d  a..I..nP....g0..
0150 ba 66 81 78 5b 0a 2b 51 39 9c b5 ea 26 f9 5b f1  .f.x[.+Q9...&.[.
0160 4d 6e 13 87 f3 de e7 ce 30 c6 00 04 ea 34 ac f1  Mn.....0....4..
```

Рис. 3.4: Сетевая активность между клиентом и сервером LDAP с TLS

При этом информации о записях каталога LDAP в открытом виде видно не будет. Это обеспечивается благодаря работы протокола TLS.

В противном случае, если соединение TLS настроено не было, утилита будет выводить следующее (рисунки 3.5):

```

ldapuser@ldap-server:/home/ldapuser
File Edit View Search Terminal Help
0030 03 e1 45 ae 00 00 01 01 08 0a 00 2a 29 2f 00 2a ..E.....*)/*
0040 24 3f $?

0.003266 192.168.100.2 -> 192.168.100.3 LDAP 594 searchRequest(2) "dc=ldap-server,dc=ru"
wholeSubtree

0000 08 00 27 81 c5 71 08 00 27 13 36 e6 08 00 45 00 ...'.q...'.6...E.
0010 02 44 c4 35 40 00 40 06 2b 28 c0 a8 64 02 c0 a8 .D.5@.@.+(...d...
0020 64 03 9d 40 01 85 9b 04 ee 22 40 ae 2d 7a 80 18 d..@....."@.-z..
0030 03 e1 df 93 00 00 01 01 08 0a 00 2a 29 2f 00 2a .....*)/*
0040 24 3f 30 82 02 0c 02 01 02 63 82 02 05 04 14 64 $?0.....c.....d
0050 63 3d 6c 64 61 70 2d 73 65 72 76 65 72 2c 64 63 c=ldap-server,dc
0060 3d 72 75 0a 01 02 0a 01 00 02 01 00 02 01 00 01 =ru.....
0070 01 00 a0 2d a3 0e 04 03 75 69 64 04 07 70 76 6f ...-....uid..pvo
0080 6c 6b 6f 76 a3 1b 04 0b 6f 62 6a 65 63 74 63 6c lkov....objectcl
0090 61 73 73 04 0c 70 6f 73 69 78 41 63 63 6f 75 6e ass..posixAccoun
00a0 74 30 82 01 ad 04 0b 6f 62 6a 65 63 74 43 6c 61 t0.....objectCla
00b0 73 73 04 03 75 69 64 04 0c 75 73 65 72 50 61 73 ss..uid..userPas
00c0 73 77 6f 72 64 04 09 75 69 64 4e 75 6d 62 65 72 sword..uidNumber
00d0 04 09 67 69 64 4e 75 6d 62 65 72 04 05 67 65 63 ..gidNumber..gec
00e0 6f 73 04 0d 68 6f 6d 65 44 69 72 65 63 74 6f 72 os..homeDirector
00f0 79 04 0a 6c 6f 67 69 6e 53 68 65 6c 6c 04 10 6b y..loginShell..k
0100 72 62 50 72 69 6e 63 69 70 61 6c 4e 61 6d 65 04 rbPrincipalName.
0110 02 63 6e 04 0f 6d 6f 64 69 66 79 54 69 6d 65 73 .cn..modifyTimes
0120 74 61 6d 70 04 0f 6d 6f 64 69 66 79 54 69 6d 65 tamp..modifyTime
0130 73 74 61 6d 70 04 10 73 68 61 64 6f 77 4c 61 73 stamp..shadowLas
0140 74 43 68 61 6e 67 65 04 09 73 68 61 64 6f 77 4d tChange..shadowM

```

Рис. 3.5: Сетевая активность между клиентом и сервером LDAP без использования TLS

Для аутентификации пользователя базы данных, пользователь вводит пароль от своего аккаунта в клиенте `psql` (рисунки 3.6). Логин пользователя пересылаются на LDAP-сервер, в виде запроса:

```
uid=pvolkov,ou=People,dc=ldap-server,dc=ru
```

Если запись такая найдена, сверяется присланный и хранимый в LDAP пароли. При совпадении подключение разрешается:

```
CentOS release 6.5 (Final)
Kernel 2.6.32-431.1.2.0.1.el6.x86_64 on an x86_64

ldap-client login: pvolkov
Password:
Last login: Thu Dec 19 15:20:53 on tty1
[pvolkov@ldap-client ~]$ psql -h 192.168.100.4 testdb -U pvolkov
Пароль пользователя pvolkov:
psql (9.3.2)
SSL-соединение (шифр: DHE-RSA-AES256-SHA, бит: 256)
Введите "help", чтобы получить справку.

testdb=> _
```

Рис. 3.6: Шифрованное соединение клиента psql

С помощью команды:

```
\conninfo
```

выводятся данные о текущем подключении к серверу PostgreSQL:

```
Вы подключены к базе данных "testdb" как пользователь "pvolkov" (сервер
"192.168.100.4" порт "5432").
SSL-соединение (шифр: DHE-RSA-AE256-SHA, бит: 256)
```

Так можно удостовериться в активности шифрования соединения между клиентом и сервером.

Заключение

Данная учебно-исследовательская работа посвящена исследованию возможностей сервера OpenLDAP для аутентификации пользователей баз данных PostgreSQL.

Были достигнуты следующие основные результаты:

1. Произведено исследование основных методов аутентификации в PostgreSQL. Выявлены их достоинства и недостатки.
2. Исследованы возможности сервера OpenLDAP.
3. Описан процесс настройки метода аутентификации LDAP в PostgreSQL.

Таким образом, можно судить о достижении поставленной цели. Продолжением темы учебно-исследовательской работы может послужить программная реализация механизма хранения метки SELinux в OpenLDAP и ее передачи для присвоения контекста безопасности пользователям операционных систем и баз данных.

Список литературы

- [1] Identification protocol (RFC1413) [Электронный ресурс] — <http://www.faqs.org/rfcs/rfc1413.html>
- [2] PostgreSQL: Documentation: 9.3: Authentication Methods: 19.3.7. Peer Authentication [Электронный ресурс] — <http://www.postgresql.org/docs/9.3/static/auth-methods.html>
- [3] Как работает PAM [Электронный ресурс] — http://www.opennet.ru/base/net/pam_linux.txt.html
- [6] Руководство администратора OpenLDAP [Электронный ресурс] — <http://www.pro-ldap.ru/tr/admin24/>
- [7] Серия стандартов X.500 [Электронный ресурс] — <http://ru.wikipedia.org/wiki/X.500>
- [8] Инженерный совет интернета (IETF) [Электронный ресурс] — <http://ru.wikipedia.org/wiki/IETF>
- [9] Стандарт RFC4510 [Электронный ресурс] — <http://tools.ietf.org/html/rfc4510>
- [10] Introduction to ASN.1 [Электронный ресурс] — <http://www.itu.int/ITU-T/asn1/introduction/>
- [11] Berkley DB [Электронный ресурс] — http://ru.wikipedia.org/wiki/Berkeley_DB
- [12] LDAP for Rocket Scientists [Электронный ресурс] — <http://www.pro-ldap.ru/tr/zytrax/>
- [13] Стандарт RFC4515 [Электронный ресурс] — <http://tools.ietf.org/html/rfc4515>
- [14] TLS [Электронный ресурс] — <http://mind-control.wikia.com/wiki/TLS>
- [15] SSSD [Электронный ресурс] — http://linuxshare.ru/docs/distro/redhat/el6/Migration_Planning_Guide/ch07s02.html
- [16] PostgreSQL: Documentation: 9.3: Authentication Methods: 19.3.8. LDAP Authentication [Электронный ресурс] — <http://www.postgresql.org/docs/9.3/static/auth-methods.html>