Впервые опубликовано на developerWorks 26.06.2008

Подготовка к экзамену LPI 301:

Тема 303. Конфигурирование

Профессионал Linux высокого уровня (LPIC-3)

Списки управления доступом в LDAP

Из этого раздела вы узнаете, как:

- Проектировать списки управления доступом LDAP
- Узнаете о синтаксисе управления доступом
- Предоставлять и отзывать права доступа LDAP

В дереве LDAP могут храниться самые различные данные, такие как телефонные номера, дни рождения или информация о денежных выплатах. Некоторые сведения могут быть публичными, а некоторые могут предназначаться только для определенного круга лиц. Также для различных пользователей могут быть определены различные типы доступов к этой информации. Например, можно указать, что только хозяин записи и администраторы могут изменять телефонный номер, а просматривать его может каждый. Все эти ограничения задаются через списки управления доступом (Access Control Lists, ACLs).

Проектирование списков управления доступом LDAP

Прежде чем приступить к написанию конфигурации, вы должны определить цели, которых вы хотите достичь. Какие разделы дерева каталога будут содержать конфиденциальную информацию? Какие атрибуты необходимо защитить и от кого? Каким образом будет использоваться дерево каталога?

Компоненты ACL

Каждая запись ACL содержит следующую информацию:

1. **Какие** элементы и атрибуты определены в ACL
2. **К кому** применяется ACL
3. Какой уровень **доступа** предоставляется

В условии "what" вы можете указать определенное различающееся имя (DN) объекта, фильтр запроса LDAP, список атрибутов или определенную комбинацию всех вышеперечисленных элементов. Фильтрация по различающимся именам позволит вам выбрать точное значение, такое как `ou=People,dc=ertw,dc=com`, или значения, соответствующие регулярному выражению . С помощью фильтра запроса LDAP можно выбрать определенный класс `objectClass` или другие атрибуты объекта. Список атрибутов представляет собой список имен атрибутов, разделенных запятыми. Более сложным условием может являться такое условие, как "Все пароли пользователей подразделения `ou=People,dc=ertw,dc=com`, являющихся администраторами".

При указании учетных записей, к которым применяется запись ACL, вам предоставляется

большая гибкость. Обычно для идентификации пользователей используется различающееся имя под названием *bindDN*, под которым они привязываются к дереву объектов. Каждый элемент LDAP может иметь атрибут `userPassword`, использующийся для аутентификации определенного пользователя. В некоторых ситуациях вы можете использовать ключевое слово *self*, соответствующее имени текущего пользователя, выполнившего вход в систему. Это может оказаться полезным в случаях, когда вы хотите разрешить пользователям редактировать их собственные данные.

Если пользователь не привязан к дереву LDAP, он считается *анонимным* (anonymous). По умолчанию анонимные пользователи могут просматривать данные каталога, поэтому вы должны решить, стоит ли оставлять им такой доступ. Далее вы можете сгруппировать анонимных пользователей (также как и любых других) по IP-адресам или методу подключения к каталогу LDAP, такому как использование открытых паролей или шифрованное подключение.

После того как вы решили, доступ к каким объектам и кому необходимо предоставить, вы должны определиться с уровнем доступа, который может варьироваться от *none* (запретить все) до *write* (разрешить изменения). Также вы можете разрешить пользователю проходить аутентификацию на основе определенной записи, но запретить ее чтение, или, предположим, разрешить выполнять операции чтения (read), поиска (search) и сравнения (compare).

Независимо от настроек ACL любые учетные записи, определенные как rootDN (пользователи, являющиеся администраторами LDAP), всегда будут иметь полный доступ к соответствующей базе данных. Не существует других способов изменить это поведение, кроме как удалить конфигурацию `rootDN` из файла slapd.conf.

Синтаксис записей управления доступом

Основная форма записи ACL, представленная в нормальной форме Бэкуса-Наура, имеет следующий вид:

```
access to <what> [ by <who> [ <access> ] [ <control> ] ]+
```

Нормальная форма Бэкуса-Наура

Нормальная форма Бэкуса-Наура (БНФ) является способом описания синтаксиса языков программирования, данных и протоколов, в том числе синтаксиса ACL. Являясь краткой, и в то же время очень точной, БНФ часто используется при разработке протоколов Интернета.

В нотации БНФ используется конструкция из двух частей, разделенных знаком ::=, который означает, что левая часть может быть заменена элементами, перечисленными в правой части. Элементы правой части конструкции БНФ, заключенные в угловые скобки (< и >), относятся к элементам левой части, заключенным в такие же скобки.

Элементы, заключенные в квадратные скобки ([и]),

являются необязательными. Вертикальная черта (|) означает "одно из нескольких", а символы + и * означают "одно или несколько из предшествующих" и "ничего, одно или несколько из предшествующих" соответственно. Если вы имели дело с регулярными выражениями, вам будут знакомы многие использующиеся здесь обозначения.

контексте описания ACL с помощью БНФ используется следующий синтаксис. Каждая запись ACL состоит из символьной строки "access to", после которой идет условие "what", определенное где-либо в другом месте. Далее идут одна или более строк вида "by <who> [<access>] [<control>]", в которых параметры who, access и control определены где-либо в другом месте, причем параметры access и control являются необязательными.

Остальной синтаксис мы рассмотрим в оставшейся части руководства.

Описание условия what

Условие *what* определяет, какие объекты и атрибуты попадают под действие правила ACL. Описание синтаксиса этого условия в нотации БНФ представлено в листинге 1.

Листинг 1. Описание синтаксиса условия what в нотации БНФ

```
<what>        ::= * |
    [dn[.<basic-style>]=<regex> | dn.<scope-style>=<DN>]
    [filter=<ldapfilter>] [attrs=<attrlist>]
<basic-style> ::= regex | exact
<scope-style> ::= base | one | subtree | children
<attrlist>    ::= <attr> [val[.<basic-style>]=<regex>]
    | <attr> , <attrlist>
<attr>        ::= <attrname> | entry | children
```

Некоторые из элементов листинга 1, такие как DN и regex, не определены непосредственно в этом фрагменте кода. Формат различающегося имени вам уже знаком, а регулярные выражения лучше всего изучать отдельно от БНФ.

Из листинга 1 видно, что условием what правила ACL может являться либо знак звездочки (*), который соответствует любому значению, либо комбинация различающегося имени, фильтра поиска LDAP и списка атрибутов. В последнем случае могут использоваться один или несколько из этих трех элементов, поскольку каждый из них заключен в квадратные скобки.

В листинге 2 приведены три условия what, определяющих различающееся имя (DN).

Листинг 2. Три примера условий what

```
dn.exact="ou=people,dc=ertw,dc=com"
dn.regex="ou=people,dc=ertw,dc=com$"
dn.regex="^cn=Sean.*,dc=com$"
```

Первому условию удовлетворяет исключительно объект `ou=people,dc=ertw,dc=com`; эта запись ACL не будет соответствовать ни каким-либо дочерним объектам, таким как `cn=Sean Walberg,ou=people,dc=ertw,dc=com`, ни родительскому объекту.

Второе условие похоже на первое за исключением того, что в нем используется регулярное выражение и *якорь* строки – знак доллара ($). Якорь определяет не часть строки, а ее расположение. Знак доллара означает конец строки, и поэтому второму условию удовлетворяет любая строка, оканчивающаяся на `ou=people,dc=ertw,dc=com`, в том числе строка `cn=Sean Walberg,ou=people,dc=ertw,dc=com`. Обратите внимание на то, что без использования якоря строка поиска может располагаться в любом месте целевой строки, такой как `ou=people,dc=ertw,dc=com,o=MegaCorp`.

В последнем примере листинга 2 используется еще один якорь – символ ^, означающий начало строки. Кроме того, в этом примере используется регулярное выражение **.***. Точка означает любой символ, а знак звездочки означает ноль или более предшествующих символов. Таким образом регекс .* соответствует любой строке, состоящей из нуля или более символов. Объединив все правила, мы увидим, что третьему условию удовлетворяет любая строка, начинающаяся на cn=Sean и оканчивающаяся на `dc=com`.

Также при определении условия what вы можете использовать фильтры запросов LDAP – наиболее полезное средство для поиска объектов на основе их классов (`objectClass`). Например, условию `filter=(objectClass=posixAccount)` удовлетворяют все объекты класса `posixAccount`. Для получения информации об атрибуте `objectClass` обратитесь к первому руководству этой серии – Подготовка к экзамену LPI 301: понятия, архитектура и модель (EN).

частью составления условий what является указание атрибутов. Как правило, данный способ используется для указания того, какие конфиденциальные атрибуты, в особенности пароли, могут быть доступны пользователям. Для того чтобы определить правило, относящееся к паролям, укажите атрибут `attrs=userPassword`.

После того как вы укажете, какие объекты и атрибуты попадают под действие правила ACL, вы должны будете указать пользователей, на которых будет распространяться это правило.

Описание условия who

Доступы предоставляются пользователю на основании различающегося имени DN, которое определяется в момент привязки клиента к каталогу. Обычно поиск различающегося имени производится в дереве каталога, но также этим именем может являться имя `rootDN`, указанное в файле slapd.conf.

Описание синтаксиса условия who в нотации БНФ представлено в листинге 3.

Листинг 3. Описание синтаксиса условия who в нотации БНФ

```
<who> ::= * | [anonymous | users | self[.<selfstyle>]
                         | dn[.<basic-style>]=<regex> | dn.<scope-style>=<DN>]
                  [dnattr=<attrname>]
                  [group[/<objectclass>[/<attrname>][.<basic-style>]]=<regex>]
                  [peername[.<peernamestyle>]=<peername>]
                  [sockname[.<style>]=<sockname>]
                  [domain[.<domainstyle>[,<modifier>]]=<domain>]
                  [ssf=<n>]
                  [transport_ssf=<n>]
                  [tls_ssf=<n>]
                  [sasl_ssf=<n>]

<style> ::=     {exact|regex|expand}
<selfstyle> ::= {level{<n>}}
<dnstyle> ::=   {{exact|base(object)}|regex
                 |one(level)|sub(tree)|children|level{<n>}}
<groupstyle> ::=        {exact|expand}
<peernamestyle> ::=     {<style>|ip|path}
<domainstyle> ::=       {exact|regex|sub(tree)}
<modifier> ::=  ={expand}
```

Также как и в условии what, здесь знак звездочки означает любое значение. Существует множество способов конкретизировать данное условие. В OpenLDAP определены три ключевых слова – `anonymous`, `users` и `self`, которые означают незарегистрированных пользователей, пользователей, прошедших проверку, и текущего пользователя, выполнившего вход в систему, соответственно. Ключевое слово `self` часто используется для того, чтобы разрешить вошедшему в систему пользователю изменять данные своего профиля. Данная функция основана на точном совпадении с DN пользователя; если информация о пользователе размещена в нескольких различных объектах, ссылка `self` действует только для той записи, с помощью которой была выполнена привязка к каталогу.

Ключевое слово `self` имеет интересную особенность: вы можете применять правило ACL к родительским или дочерним записям текущей записи пользователя, используя другое ключевое слово - `level`. Ссылка `self.level{1}` соответствует текущей и родительской записям пользователя, а ссылка `self.level{-1}` – текущей и всем непосредственно присоединенным дочерним записям.

Что касается различающегося имени, то вы можете отфильтровать атрибут DN на основе регулярного выражения или точного совпадения, используя конструкции `dn.exact="`*DN*`"` и `dn.regex="`*regex*`"`соответственно. Далее в этом руководстве будет рассмотрен пример динамического связывания условий what и who.

Произвольные элементы каталога могут быть защищены с помощью ключевого слова `dnattr`, которое используется совместно с именем атрибута. Если DN инициатора запроса содержится в указанном атрибуте целевого объекта, то считается, что условие ACL выполняется. Например, если вы добавите в ACL условие `dnattr=manager`, а в запись пользователя Fred Smith добавите атрибут manager: `cn=Joe Blow,ou=people,dc=ertw,dc=com`, то при обращении пользователя Joe Blow к записи пользователя Fred Smith условие ACL будет выполняться.

Ключевое слово group похоже на `dnattr` за исключением того, что его параметры относятся к группе, определенной где-либо в другом месте дерева, а не к атрибуту элемента каталога. По умолчанию классом объекта (`objectClass`) группы является `groupOfNames`, а участники группы перечислены в атрибуте `member`.

Ключевые слова peername, sockname и domain используются для идентификации клиентского подключения. Ключевое слово peername используется для указания IP-адреса клиента, например, peernameip=127.0.0.1. Ключевое слово sockname относится к редко используемым подключениям через именованные каналы (named pipes), а domain используется для указания связанного с IP-адресом имени узла, которое может быть легко подделано.

Заключительный ряд опций относится к уровню безопасности подключения, который в терминах OpenLDAP называется Security Strength Factor (SSF). Эти опции окажутся более понятными, если вы знакомы с механизмами безопасности, использующимися для подключений к OpenLDAP, такими как защита транспортного уровня (Transport Layer Security, TLS) и механизм аутентификации Simple Authentication and Security Layer (SASL).

Вы можете использовать различные комбинации всех вышеперечисленных опций. Например, вы можете разрешить изменять пароли только тем пользователям, которые одновременно являются администраторами, подключаются с заданного диапазона IP-адресов и используют определенный уровень шифрования. Вы можете определять и более простые условия; например, вы можете разрешить подключаться к каталогу только существующим в системе пользователям или же предоставлять доступ вне зависимости от результатов проверки подлинности.

Описание параметра access

После того как вы определили, кому и к каким объектам каталога разрешен доступ, вы должны указать уровень предоставляемого доступа. Описание параметра access в нотации БНФ представлено в листинге 4.

Листинг 4. Описание параметра access в нотации БНФ

```
<access>  ::=  [[real]self]{<level>|<priv>}
<level> ::= none|disclose|auth|compare|search|read|write
<priv> ::= {=|+|-}{w|r|s|c|x|d|0}+
```

Когда уровень доступа указывается в формате level, каждый последующий уровень включает в себя все предыдущие уровни. Таким образом, предоставляя разрешение read, вы также предоставляете разрешения search, compare, auth и disclose. Уровни доступа none и disclose запрещают любой доступ к данным и отличаются лишь тем, что некоторые сообщения об ошибках, которые могут раскрыть информацию о содержимом дерева каталога, не выводятся на уровне доступа none и выводятся на уровне disclose.

Альтернативным способом является указание уровня доступа в терминах разрешенных операций LDAP с использованием формата priv. В этом формате опции перечислены в обратном порядке относительно формата level; опция w означает операцию записи, а опция 0 – полный запрет любых операций. При указании уровня доступа в формате priv не происходит неявного наследования разрешений, как в случае использования формата level; если вы хотите предоставить полный доступ, вы должны сделать это с помощью строки wrscx.

Символы =/+/- перед буквенными опциями определяют способ совместного использования указанного разрешения и уже действующих разрешений в тех случаях, когда применяются несколько правил. Если перед опцией доступа стоит символ =, все определенные ранее разрешения игнорируются, и используется указанное разрешение.

Если используются символы + или -, то указанное разрешение соответственно добавляется или удаляется из действующих разрешений.

Параметр control

По умолчанию применение списков доступа в OpenLDAP происходит по методу первого совпадения. OpenLDAP находит первую запись ACL, удовлетворяющую условию what, и в пределах этой записи ищет первый элемент, удовлетворяющий условию who. Этот метод соответствует ключевому слову `stop`, указанному после описания уровня доступа. Два других ключевых слова – это `continue` и `break`. Если вы используете ключевое слово `continue`, то в текущей записи ACL производится поиск следующего элемента, удовлетворяющего условию who. Если вы используете ключевое слово `break`, обработка текущей записи ACL прекращается, но выполняется поиск следующей записи ACL, удовлетворяющей условию who.

Собираем вместе все компоненты правила ACL

Теперь, когда мы рассмотрели все три (четыре, если считать параметр control) компонента правила ACL, можно собрать их воедино и создать политику доступа. В листинге 5 приведен простой список ACL, который позволяет зарегистрированным в системе пользователям просматривать дерево каталога и изменять (но не просматривать) свои личные пароли.

Листинг 5. Простой список ACL

```
access to attrs=userPassword
    by self =xw
    by anonymous auth

access to *
    by self write
    by users read
```

Первое правило ACL применяется в случаях обращения пользователей к полю `userPassword`. Каждый пользователь может изменять свой пароль и использовать его для входа в систему. Эти разрешения предоставляются путем использования знака "=". Анонимным пользователям разрешено проходить аутентификацию. Поскольку в момент привязки к дереву каталога любой пользователь является анонимным, ему необходимо предоставить разрешение `auth` для того, чтобы он мог войти в систему и стать обычным, привилегированным пользователем.

Если пользователи обращаются к элементам каталога, которые не являются паролями, то в действие вступает второе правило ACL. Опять же, каждый пользователь имеет полный доступ ко всем полям своей записи (за исключением поля `userPassword` в силу действия первого правила ACL), в то время как все прошедшие проверку пользователи могут просматривать остальные данные каталога.

В листинге 6 приведена запись ACL, связывающая условия what и who и содержащая регулярные выражения.

Листинг 6. Немного фантазии с использованием регулярных выражений

```
access to dn.regex="cn=([^,]+),ou=addressbook,dc=ertw,dc=com"
        by dn.regex="cn=$1,ou=People,dc=ertw,dc=com" write
        by users read
```

Правило ACL, приведенное в листинге 6, разрешает пользователям изменять соответствующие им записи в ветке `ou=addressbook,dc=ertw,dc=com` дерева каталога. Регулярное выражение `[^,]+` соответствует строке любых символов за исключением запятой, а круглые скобки сохраняют эту строку в качестве значения переменной `$1`; следующая пара скобок сохраняется в переменную `$2`, и так далее до `$9` включительно. В условии who полученное имя пользователя повторно используется для определения того, кто может получить доступ к записи каталога. Если имя пользователя совпадает с именем записи, к которой он обращается, то ему предоставляется полный доступ. В противном случае прошедший проверку пользователь получает доступ только на просмотр этой записи.

Практические рекомендации

Поскольку в OpenLDAP используется метод первого совпадения, размещайте более конкретизированные записи ACL в начале списка; в противном случае существует большая вероятность того, что при соответствии записи ACL более общему условию последующие записи списка будут проигнорированы. Этот прием также можно использовать для предоставления или отзыва доступов у отдельного пользователя; для этого просто поместите запись ACL, содержащую условие для конкретного пользователя, в начало списка.

По возможности старайтесь создавать простые записи ACL. Это уменьшает вероятность ошибок и повышает производительность системы, поскольку записи ACL анализируются каждый раз при обращении к дереву каталога.

Регулярные выражения

Регулярные выражения используются для поиска и изменения текста на основе выбранных правил. Вы можете иметь общее представление о том, как может выглядеть текстовый фрагмент, или знать, каким определенным шаблонам он может соответствовать, и на основании этого построить регулярное выражение, которое поможет вам найти необходимый текст. Регулярное выражение, или сокращенно *регекс* (от англ. regex) состоит из символьных констант и метасимволов, на основе которых строятся различные шаблоны.

Простым регулярным выражением является `hello`, что соответствует любой строке символов, содержащей шаблон `hello`.

Вы можете не знать, является ли первая буква строки заглавной, или нет. В этом случае используйте метасимволы `[]`, соответствующие любому единичному символу из числа заключённых в скобки. Так, регекс `[Hh]ello` соответствует как строке `hello`, так и строке `Hello`.

Точка в регулярном выражении соответствует любому единичному символу. Регекс `.ello` соответствует строкам `Hello` и `hello`, но также и строкам `fellow` и `cello`. Тем не менее, этот регекс не соответствует строке `ello`, поскольку точка должна соответствовать какому-либо символу.

Символы `?`, `*` и `+` соответствуют нулю или одной копии предыдущего символа, нулю или нескольким копиям предыдущего символа, и одной или более копиям предыдущего символа соответственно. Таким образом, регулярное выражение `hello+` соответствует строкам `hello` и `hellooooo`, но не строке `hell`.

Регулярные выражения имеют множество различных опций и позволяют вам эффективно находить нужные строки в текстовых файлах. В контексте OpenLDAP регулярные выражения используются для определения общих фрагментов в различающихся именах (Distinguished Name, DN), таким образом, избавляя от необходимости набирать вручную сотни возможных вариантов.

Репликация LDAP

В какой-то момент использование единственного сервера LDAP может перестать удовлетворять вашим потребностям. Возможность выхода из строя сервера LDAP в вашей организации может оказаться неприемлемой, или же нагрузка на сервер может оказаться достаточно высокой, и вы решите распределить ее между несколькими серверам. Неважно, по какой причине, но у вас может возникнуть потребность в использовании нескольких серверов.

Вы можете разместить отдельные части каталога LDAP на нескольких различных серверах, однако это ведет к снижению уровня надежности, не говоря о сложности правильного распределения запросов. В идеальном случае на каждом сервере содержится одинаковая копия каталога. Все изменения, произошедшие на любом из серверов, передаются на остальные серверы определенным способом, обеспечивающим актуальность данных в любой момент времени. Этот процесс называется *replication*.

Этот сценарий, называющийся *multi-master*репликацией, является довольно сложным, поскольку в этом случае не существует единственного, четко определенного сервера, управляющего данными. Чаще всего используется репликация master-slave, при которой один главный (master) сервер управляет всеми изменениями каталога и рассылает их на подчиненные (slave) серверы. Запросы LDAP при этом могут быть обработаны любым из серверов. Данная схема может быть расширена до репликации с использованием *узла реплики*, когда данные с главного сервера реплицируются на один подчиненный сервер, а он в свою очередь реплицирует эти данные на все остальные подчиненные серверы.

В OpenLDAP существует два метода репликации. В первом методе используется *slurpd* – отдельный демон, который отслеживает изменения на главном сервере и передает их на подчиненные серверы. Во втором методе используется встроенный механизм репликации LDAP Sync сервера OpenLDAP, также известный как *syncrepl*. Репликация посредством slurpd считается устаревшей, и пользователям все больше предлагается использовать syncrepl. В этом разделе будут рассмотрены оба этих метода.

Репликация посредством slurpd

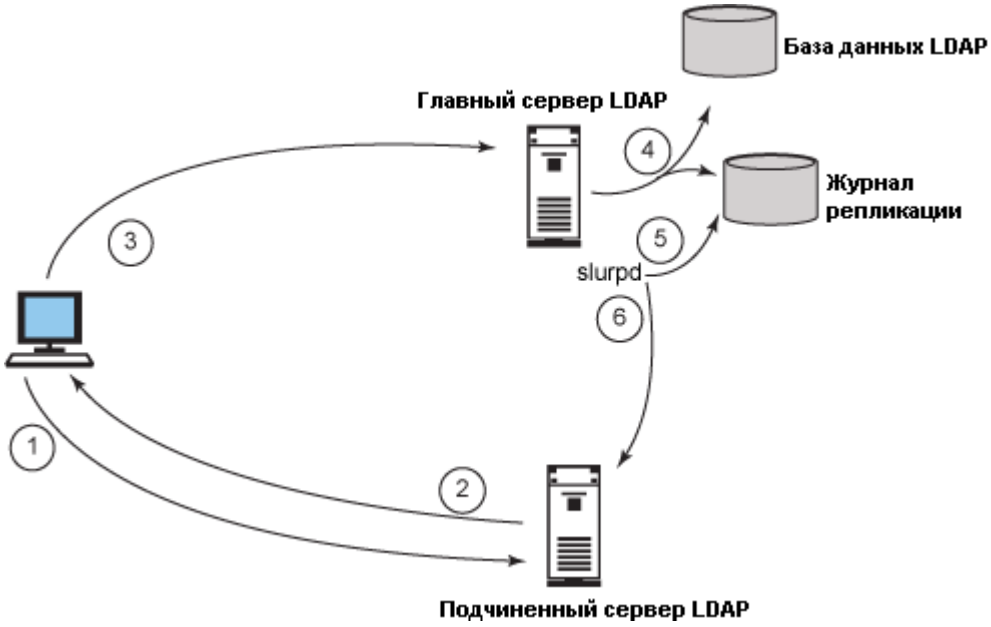
Репликация посредством slurpd является извещающей (push) репликацией, при которой главный сервер извещает об изменениях все подчиненные серверы. Если клиент пытается обновить данные на подчиненном сервере, этот сервер посылает клиенту так называемую *возвращаемую ссылку* (referral), перенаправляющую его на главный сервер. За подачу повторного запроса на главный сервер отвечает клиентский компьютер. Slurpd является отдельным демоном и настраивается в файле slapd.conf.

Обработка данных в модели репликации slurpd

Главный сервер – это сервер, обрабатывающий все поступающие от клиентов запросы на изменение каталога и содержащий достоверный источник данных. Любые изменения, произошедшие в дереве каталога главного сервера, записываются в *журнал репликации*, за которым наблюдает демон slurpd. Обнаружив изменения в журнале репликации, slurpd извещает о них все подчиненные серверы.

На рисунке 1 изображена схема работы демона slurpd.

Рисунок 1. Обработка данных в модели репликации slurpd



Описание процесса:

1. Клиентский компьютер посылает запрос на обновление данных, который случайным образом попадает на подчиненный сервер.
2. Подчиненный сервер знает о том, что он может выполнять операции записи только в случае получения их от своего партнера по репликации, и поэтому посылает клиенту возвращаемую ссылку, перенаправляющую его на главный сервер.
3. Клиент повторно посылает запрос на обновление данных, обращаясь к главному серверу.
4. Главный сервер выполняет обновление данных и записывает изменения в журнал репликации.
5. Slurpd, который также запущен на главном сервере, обнаруживает изменения в журнале репликации.
6. Slurpd направляет изменения на подчиненный сервер.

Таким образом, подчиненные серверы могут быть синхронизированы с главным сервером с небольшой задержкой. Slurpd всегда знает, какие подчиненные сервера нуждаются в обновлении, если произошла какая-либо задержка или сбой.

Настройка slurpd

Настройка репликации посредством slurpd состоит из следующих шагов:

1. Создание учетной записи, которую slurpd будет использовать для аутентификации подчиненной реплики.
2. Настройка имени подчиненного сервера на главном сервере.
3. Настройка подчиненного сервера в качестве реплики, в том числе настройка необходимых списков управления доступом.
4. Копирование базы данных с главного сервера на подчиненный сервер.

Создание учетной записи реплики не представляет особой сложности. Единственным требованием для этого является наличие в учетной записи атрибута `userPassword`. Вы можете использовать класс объекта `inetOrgPerson objectClass`, как в случае создания учетных записей, принадлежащих служащим, или более общий класс объекта,

такой как `account`, добавив к нему вспомогательный класс `simpleSecurityObject`. Возвращаясь к первому руководству, вспомним, что структурные классы объектов описывают запись (и поэтому вы можете использовать только один класс объекта для каждой записи), тогда как вспомогательные классы объектов добавляют атрибуты к записи независимо от ее структурного класса. В листинге 7 приведен код в LDIF-формате (LDAP Data Interchange Format – формат обмена данными LDAP) для добавления учетной записи реплики.

Листинг 7. LDIF код для добавления учетной записи реплики

```
dn: uid=replica1,dc=ertw,dc=com
uid: replica1
userPassword: replica1
description: Account for replication to slave1
objectClass: simpleSecurityObject
objectClass: account
```

В листинге 7 представлена простая запись, содержащая только имя пользователя, пароль и описание, чего вполне достаточно для целей репликации. Описание не является обязательным, однако рекомендуется использовать его в целях документирования. Запомните пароль – он понадобится на следующем шаге!

Теперь для работы slurpd необходимо настроить главный сервер на сохранение всех изменений в журнале репликации, а также настроить реплику. Важно помнить о том, что slurpd извещает об изменениях все подчиненные серверы, а его параметры конфигурации настраиваются в файле slapd.conf. Это в свою очередь поможет вам помнить о том, где нужно настраивать репликацию, и о том, что учетные данные для аутентификации располагаются на главном сервере. Поскольку учетные данные являются частью дерева каталога, подчиненный сервер всегда сможет проверить их. В листинге 8 приведены настройки главного сервера, обеспечивающие возможность репликации.

Листинг 8. Настройка репликации посредством slurpd на главном сервере

```
replica  uri=ldap://slaveserver.ertw.com
    suffix="dc=ertw,dc=com"
    binddn="uid=replica1,dc=ertw,dc=com"
    credentials="replica1"
    bindmethod=simple

replogfile /var/tmp/replicationlog
```

Настройка репликации происходит в режиме базы данных, поэтому убедитесь, что команда `replicareplica` стоит где-нибудь после настройки первого параметра `database`. Команда `replica` содержит ряд параметров в формате параметр=значение. Параметр `uri` определяет имя или IP-адрес подчиненного сервера в формате унифицированного идентификатора ресурса (Uniform Resource Identifier, URI). Перед именем подчиненного сервера ставится префикс `ldap://`.

После того, как вы указали имя подчиненного сервера, вы можете указать с помощью параметра `suffix` необязательное имя базы данных для репликации. По умолчанию

реплицируются все базы данных. Заключительным требованием является предоставление информации об учетных данных, чтобы slurpd мог подключаться к указанной ссылке `uri`. Для выполнения простой аутентификации вам будет достаточно настроить параметры `binddn`, `bindmethod` и `credentials` (параметр `userPassword` был настроен вами ранее).

На заключительном этапе настройки главного сервера необходимо указать службе slapd, где должен храниться журнал репликации. Необходимо указать полный путь, поскольку относительные пути не будут работать. Вам не нужно беспокоиться о создании файла журнала, поскольку slapd сделает это за вас; тем не менее, указанный вами путь должен быть доступен для записи пользователю, от имени которого выполняются демоны slapd и slurpd.

На подчиненном сервере вы должны настроить учетную запись репликации, а также указать, что получая запросы на изменение данных, он должен перенаправлять клиентов на главный сервер с помощью возвращаемой ссылки.

Листинг 9. Конфигурация подчиненного сервера

```
updatedn uid=replica1,dc=ertw,dc=com
updateref ldap://masterserver.ertw.com
```

Значением параметра `updatedn` является учетная запись, созданная ранее на главном сервере. Эту учетную запись будет использовать демон slurpd для извещения подчиненных серверов об изменениях, произошедших в дереве каталога. Параметр `updateref` – это еще одна ссылка URI, указывающая на главный сервер LDAP. В листинге 10 приведен пример обновления клиентом подчиненного сервера и получения возвращаемой ссылки после настройки вышеуказанной конфигурации.

Листинг 10. Получение клиентом возвращаемой ссылки при попытке обновления данных на подчиненном сервере

```
[root@slave openldap]# ldapadd -x -D cn=root,dc=ertw,dc=com -w mypass -f newaccount.ldif
adding new entry "cn=David Walberg,ou=people,dc=ertw,dc=com"
ldap_add: Referral (10)
        referrals:
                ldap://masterserver.ertw.com/cn=David%20Walberg,ou=People,dc=ertw,dc=com
```

Клиент командной строки OpenLDAP не переходит по возвращаемым ссылкам, но другие библиотеки LDAP делают это. Если вы используете LDAP в окружении с работающей репликацией, вы должны убедиться, что ваши приложения корректно переходят по возвращаемым ссылкам.

На заключительном этапе настройки процесса репликации необходимо обеспечить идентичность базы данных на главном и подчиненном серверах. Для этого выполните следующие шаги:

1. Остановите главный сервер LDAP.
2. Остановите подчиненный сервер LDAP.
3. Скопируйте все файлы базы данных с главного сервера на подчиненный сервер.
4. Запустите главный и подчиненный серверы.

5. Запустите демон slurpd.

Перед копированием базы данных оба сервера LDAP должны быть остановлены для того, чтобы во время этой процедуры в каталоге не могли произойти какие-либо изменения. Крайне важно, чтобы оба сервера начали работу с идентичным набором данных, иначе впоследствии может произойти их рассинхронизация. Репликация посредством slurpd, по существу, выполняет на подчиненном сервере все те же транзакции, что происходят на главном сервере, поэтому любые расхождения могут привести к проблемам.

В зависимости от дистрибутива и сценариев загрузки операционной системы slurpd может автоматически запускаться вместе с slapd. Если демон slurpd не запускается автоматически, запустите его вручную из командной строки, выполнив команду `slurpd`.

Прежде чем двигаться дальше, у вас должна быть настроена работающая репликация. Создайте учетную запись на главном сервере и проверьте работу службы репликации. Убедитесь также, что при получении запросов на обновление данных подчиненный сервер посылает клиенту возвращаемые ссылки.

Мониторинг репликации

Очень важно уметь выполнять мониторинг репликации, поскольку в процессе работы могут возникать различные ошибки, приводящие к рассинхронизации данных. Эти же знания пригодятся и при отладке.

Файлы slurpd хранятся в каталоге var/lib/ldap/replica (отдельно от журнала репликации службы slapd). В этом каталоге содержатся собственные журналы репликации службы slurpd и прочие так называемые *reject-файлы* (файлы отклонения). Если попытка slurpd обновить подчиненный сервер оканчивается неудачей, то данные сохраняются в файл с расширением .rej. В этом файле содержится LDIF код записи, а также описание возвращенной сервером ошибки, например, `ERROR: Already exists`. В листинге 11 приведен пример reject-файла, содержащего другую ошибку.

Листинг 11. Reject-файл репликации

```
ERROR: Invalid DN syntax: invalid DN
replica: slaveserver.ertw.com:389
time: 1203798375.0
dn: sendmailMTAKey=testing,ou=aliases,dc=ertw,dc=com
changetype: add
objectClass: sendmailMTAAliasObject
sendmailMTAAliasGrouping: aliases
sendmailMTACluster: external
sendmailMTAKey: testing
sendmailMTAAliasValue: testinglist@ertw.com
structuralObjectClass: sendmailMTAAliasObject
entryUUID: 5375b66c-7699-102c-822b-fbf5b7bc4860
creatorsName: cn=root,dc=ertw,dc=com
createTimestamp: 20080223202615Z
entryCSN: 20080223202615Z#000000#00#000000
modifiersName: cn=root,dc=ertw,dc=com
modifyTimestamp: 20080223202615Z
```

Файл отклонения, приведенный в листинге 11, начинается с текстового описания ошибки ("ERROR: Invalid DN syntax: invalid DN"), после которого следует код в формате LDIF. Обратите внимание на то, что первым атрибутом является **replica** – имя реплики, которая не

смогла выполнить обновление, а вторым атрибутом является **time** – время возникновения ошибки (в формате времени UNIX). Следующие несколько атрибутов относятся к записи, которая была отклонена.

Последние семь атрибутов называются **операционными атрибутами**. Эти атрибуты не относятся к исходному обновлению, а были добавлены сервером LDAP в целях внутреннего мониторинга. Записи LDAP был присвоен универсальный уникальный идентификатор (Universally Unique Identifier, UUID), а также некоторая информация о том, когда и кем эта запись изменялась.

Ошибка, показанная в листинге 11, скорее всего, произошла по причине отсутствия необходимой схемы на подчиненном сервере, который не смог опознать атрибут **sendmailMTAKey**, и, как следствие, посчитал DN записи некорректным. Прежде чем можно будет возобновить репликацию, необходимо обновить схему подчиненного сервера.

Чтобы применить отклоненную запись, вы должны найти ошибку и устранить причины ее возникновения. Когда вы уверены, что отклоненная запись будет применена без ошибок, воспользуйтесь режимом slurpd *one-shot mode*, выполнив команду `slurpd -r /path/to/rejection.rej -o`. Параметр `-r` указывает slurpd прочесть указанный журнал репликации, а параметр `-o` переводит slurpd в режим one-shot mode, который означает, что по завершении обработки журнала slurpd закончит свою работу, а не перейдет в режим ожидания с целью добавления других записей (этот режим используется по умолчанию).

Если репликация не работает вовсе, лучше всего начать диагностику с главного сервера. Прежде всего, остановите процесс slurpd с помощью команды `kill` и измените какой-нибудь объект дерева каталога. Затем проверьте, увеличивается ли размер файла журнала репликации. Если размер файла не увеличивается, значит, главный сервер настроен неправильно. Далее, запустите slurpd с параметром командной строки `-d 255`. Данный режим отладки позволит отслеживать все действия slurpd, выполняемые при обработке журнала репликации. Ищите ошибки, в особенности, относящиеся к открытию файлов и контролю доступа.

Наконец, выполните на подчиненном сервере команду `loglevel auth sync`, чтобы проверить, не возникают ли какие-либо ошибки в процессе репликации (slapd ведет запись событий в журнал syslog от имени источника `local4`, поэтому, возможно вам придется добавить строку `local4.* /var/log/slapd.log` в файл /etc/syslog.conf).

Репликация LDAP Sync

Репликация посредством slurpd является простым, открытым решением, но имеет ряд недостатков. Остановка главного сервера с целью синхронизации подчиненного сервера в лучшем случае может доставить неудобства, а в худшем – повлиять на качество предоставляемых услуг. Архитектура slurpd на основе извещений также имеет ряд ограничений. Для своего времени slurpd работал достаточно хорошо, но необходимо было создать нечто лучшее. В документации RFC 4533 описывается процесс синхронизации контента LDAP, реализованный в OpenLDAP посредством механизма LDAP Sync, также известного как *syncrepl*.

Syncrepl является оверлейной программой, встраиваемой между ядром slapd и базой данных. Все операции записи в дерево каталога отслеживаются механизмом syncrepl; при этом не требуется использовать какую-либо отдельную службу. За исключением механизма репликации и поддержки ролей (этот вопрос будет рассмотрен далее) принципы работы

syncrepl аналогичны работе slurpd. Попытки обновления реплики отклоняются, а клиент получает возвращаемую ссылку на главный сервер.

Служба syncrepl запускается на подчиненном сервере, который теперь называется *получателем*. Главный сервер выступает в роли *provider*. При репликации посредством syncrepl получатель подключается к источнику для получения обновлений каталога. В наиболее часто использующемся режиме работы, который называется*refreshOnly*, получатель получает все измененные с момента последнего обновления записи, запрашивает маркер, содержащий сведения о последней синхронизации, и затем отключается. При следующем подключении этот маркер передается источнику, которые возвращает те записи, которые были изменены с момента последней синхронизации.

Другим режимом работы syncrepl является режим *refreshAndPersist*, работающий почти так же, как и refreshOnly, за исключением того, что получатель не отключается от источника после выполнения синхронизации, а продолжает получать все изменения, поддерживая подключение. Все изменения, произошедшие после первоначальной синхронизации, немедленно передаются от источника получателю.

Настройка syncrepl

В листинге 12 представлена конфигурация сервера-источника для обоих режимов работы syncrepl (refreshOnly и refreshAndPersist).

Листинг 12. Настройка репликации syncrepl на сервере-источнике

```
overlay syncprov
syncprov-checkpoint 100 10
syncprov-sessionlog 100
```

Первая строка в листинге 12 включает возможность использования оверлейной программы syncprov. Оверлей должен настраиваться в отношении определенной базы данных, поэтому код в указанном листинге должен располагаться после настройки параметра `database`. Следующие две строки являются необязательными, однако повышают надежность. Строка `syncprov-checkpoint 100 10` указывает серверу сохранять значения параметра `contextCSN` на жесткий диск каждые 100 операций записи или каждые 10 минут. Параметр `contextCSN` является частью упомянутого ранее маркера, помогающего серверу-получателю получать изменения, произошедшие с момента завершения предыдущей репликации. Строка `syncprov-sessionlog 100` регистрирует все операции записи в журнале, хранящемся на жестком диске, что также помогает поддерживать цикл синхронизации.

Для получения дополнительной информации о настройке источника обратитесь к man-руководству `slapo-syncprov(5)`.

В листинге 13 приведена настройка сервера-получателя, участвующего в двустороннем процессе репликации.

Листинг 13. Настройка репликации syncrepl в режиме refreshOnly на сервере-получателе

```
updateref ldap://masterserver.ertw.com
syncrepl rid=1
 provider=ldap://masterserver.ertw.com
 type=refreshOnly
 interval=00:01:00:00
 searchbase="dc=ertw,dc=com"
 bindmethod=simple
 binddn="uid=replica1,dc=ertw,dc=com"
 credentials=replica1
```

Так же, как и для команды `replica` в случае синхронизации посредством slurpd, для команды `syncrepl` требуется указать ссылку `updateref` и предоставить информацию о дереве каталога, которое вы пытаетесь реплицировать. Также необходимо предоставить учетные данные, которые будут использоваться для репликации. На этот раз учетные данные указываются на стороне получателя, а уровень доступа к источнику, предоставляемый для этих учетных данных, должен обеспечивать возможность чтения реплицируемой части дерева каталога. Все изменения базы данных на сервере-получателе выполняются от имени `rootdn`.

Параметры `rid`, `provider`, `type` и `interval` относятся к syncrepl. Параметр `rid` идентифицирует получателя на главном сервере. Получатель должен иметь уникальный идентификатор (ID), который может принимать значение от 1 до 999. Параметр `provider` является ссылкой LDAP в формате URI, указывающей на сервер-источник. Параметр `type` указывает, что синхронизация будет выполняться только в режиме `refreshOnly`, а параметр `interval` определяет периодичность синхронизации, равную одному часу. Значение параметра `interval` задается в формате `DD:hh:mm:ss`.

Запустите сервер-получатель с пустой базой данных, и его данные будут реплицироваться с сервером-источником каждый час.

Перевести синхронизацию в режим refreshAndPersist очень просто. Для этого удалите в листинге 13 строку `interval` и замените значение параметра `type` на `refreshAndPersist`.

Фильтрация syncrepl

Следует заметить, что вам не обязательно может потребоваться реплицировать дерево LDAP целиком. Для того чтобы выбрать только необходимые для репликации данные, вы можете использовать следующие команды.

Таблица 3. Команды для фильтрации трафика репликации

Команда	Описание
`searchbase`	Различающееся имя, указывающее на узел дерева, с которого начнется репликация. При необходимости OpenLDAP заполнит нужные родительские узлы, чтобы обеспечить целостность дерева.
`scope`	Может принимать одно из значений: `sub`, `one` или `base`. Этот параметр определяет глубину репликации данных относительно начальной точки, указанной в параметре `searchbase`. Значением по умолчанию является `sub`; это значение охватывает узел `searchbase` и все его дочерние узлы.
`filter`	Фильтр поиска LDAP, такой как (`objectClass=inetOrgPerson`), определяющий набор записей для репликации.
`attrs`	Список атрибутов, которые будут извлечены из выбранных записей.

Также как и остальные опции `syncrepl`, все вышеперечисленные параметры записываются в формате параметр=значение.

Обеспечение безопасности LDAP

До этого момента доступ к slapd осуществлялся по незащищенным каналам с использованием открытых, нешифрованных паролей. Этот метод называется *простой* аутентификацией. В данном разделе мы рассмотрим методы шифрования соединений между клиентами и сервером.

Использование SSL и TLS для защиты подключений

Вы можете быть знакомы с протоколом защищенных сокетов (Secure Sockets Layer, SSL) и протоколом защиты транспортного уровня (Transport Layer Security, TLS) как с протоколами, использующимися для защиты Web-транзакций. Всякий раз, используя ссылку URI с префиксом https, вы имеете дело с протоколом SSL или TLS. TLS является усовершенствованием протокола SSLv3 и в некоторых случаях имеет обратную совместимость с SSL. Из-за своей общей наследственности и совместимости эти два протокола часто рассматриваются вместе как единый протокол – SSL.

Протокол SSL использует сертификаты X.509 – файлы стандартизированного формата, содержащие цифровую подпись, поставляемую доверенной третьей стороной – центром сертификации (Certificate Authority, CA). Действительная цифровая подпись означает, что подписанные данные не были подделаны с момента их подписания. Если хотя бы один бит данных, содержащих цифровую подпись, будет изменен, эта подпись не сможет пройти проверку, и будет являться недействительной. Независимые участники процессов, такие как клиент и сервер, могут выполнять проверку цифровых подписей, поскольку на обоих из них настроены доверительные отношения с центром сертификации.

Сертификат сервера содержит информацию о владельце сервера, включающую в себя публичное Интернет-имя сервера. Таким образом, вы можете быть уверены, что подключаетесь именно тому серверу, к которому намереваетесь подключиться, поскольку его имя в точности соответствует имени, указанному в сертификате (при условии, что вы доверяете центру сертификации, проверившему и подписавшему сертификат). Кроме того, сертификат содержит открытый ключ сервера, который может использоваться для шифрования данных. Если данные зашифрованы таким ключом, то расшифровать и прочесть их сможет только владелец секретного (личного) ключа.

Открытый и секретный ключи формируют основу метода шифрования с использованием *открытого ключа*, или *ассиметричного шифрования*. Шифрование является ассиметричным по той причине, что информация, зашифрованная с помощью открытого ключа, может быть расшифрована только с помощью секретного ключа, и наоборот – информация, зашифрованная с помощью секретного ключа, может быть расшифрована только с помощью открытого. Для шифрования данных в традиционном понимании (например, сохранение в тайне определенного сообщения) используется первый метод – открытый ключ является публичным, а личный ключ хранится в секрете. Благодаря механизму ассиметричного шифрования, зашифровать сообщение можно с помощью секретного ключа, а любой владелец открытого ключа сможет расшифровать его – именно так работают цифровые подписи.

После того как клиент подключается к серверу и получает его сертификат, он может проверить, соответствует ли имя сервера заявленному. Это помогает защититься от атак типа *man in the middle* (человек посередине). Открытый ключ можно использовать в определенном протоколе, работа которого завершается тем, что клиент и сервер договариваются об использовании общего секретного ключа, который не может быть

определен ни одной наблюдающей за соединением стороной. В дальнейшем этот секретный ключ используется для шифрования оставшихся данных соединения между клиентом и сервером – этот процесс называется *симметричным* шифрованием, поскольку для шифрования и расшифровки данных используется один и тот же ключ. Разделение на асимметричный и симметричный методы шифрования существует по той причине, что последний из них работает на порядок быстрее. Шифрование на основе открытого ключа используется для аутентификации и установления договоренности об использовании общего секретного ключа, после чего в действие вступает симметричное шифрование.

Чтобы применить все вышеизложенное к OpenLDAP, необходимо создать сертификат сервера и настроить сервер на его использование. В нашем примере будет использоваться самоподписанный сертификат, а не сертификат, выпущенный центром сертификации. Это означает, что конечный сертификат будет подписан им же самим. Такой подход не обеспечивает того уровня доверия, как в случае использования подписи ЦС, но этого оказывается вполне достаточно для целей тестирования. В листинге 14 показан процесс генерации ключей.

Листинг 14. Генерация пары ключей TLS

```
[root@bob ssl]# openssl genrsa -out ldap.key 1024
Generating RSA private key, 1024 bit long modulus
.............................++++++
.......................++++++
e is 65537 (0x10001)
```

В листинге 14 показан процесс генерации ключа, запущенный путем выполнения команды `openssl genrsa`. Длина ключа составляет 1024 бита и на сегодняшний день считается достаточной для открытых ключей (обратите внимание, что использование более длинных ключей приводит к замедлению криптографических операций и может ввести в замешательство некоторых клиентов). Далее команда `openssl req` берет открытую часть только что созданной пары ключей, добавляет некоторую информацию о местоположении и запаковывает получившийся результат – запрос на подписание сертификата (Certificate Signing Request, CSR), который должен быть подписан центром сертификации. Этот процесс показан в листинге 15.

Листинг 15. Создание запроса на подписание сертификата

```
[root@bob ssl]# openssl req -new -key ldap.key -out ldap.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:CA
State or Province Name (full name) [Berkshire]:Manitoba
Locality Name (eg, city) [Newbury]:Winnipeg
Organization Name (eg, company) [My Company Ltd]:ERTW
Organizational Unit Name (eg, section) []:Directory Services
Common Name (eg, your name or your server's hostname) []:masterserver.ertw.com
Email Address []:sean@ertw.com
```

```
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Сформированный файл ldap.csr (а также платеж на довольно приличную сумму) можно послать в центр сертификации на подпись. Эта же процедура используется для генерации сертификата Web-сервера. Если вы посылаете ваш запрос на подпись в ЦС, убедитесь, что вся указанная вами информация написана без ошибок, аббревиатуры используются только для поля Country Name, а поле Common Name *в точности* совпадает с DNS-именем вашего сервера, которое будет использоваться клиентами для подключения к нему.

Вместо того чтобы подписать запрос CSR в центре сертификации, в нашем примере мы сделаем это самостоятельно, как показано в листинге 16.

Листинг 16. Подписывание запроса CSR

```
[root@bob ssl]# openssl x509 -req -days 1095 -in ldap.csr -signkey ldap.key -out ldap.cert
Signature ok
subject=/C=CA/ST=Manitoba/L=Winnipeg/O=ERTW/OU=Directory Services/
  CN=masterserver.ertw.com/emailAddress=sean@ertw.com
Getting Private key
```

В листинге 16 показан процесс подписания ключа, запущенный путем выполнения команды `openssl x509`. Опция `-req` говорит команде openssl о том, что входным файлом является запрос на подписание сертификата. Срок действия сертификата составляет 1095 дней, или 3 года. Теперь у вас есть файлы ldap.key (личный ключ) и ldap.cert (сертификат и открытый ключ).

Прежде чем продолжить, добавьте в файл /etc/openldap/ldap.conf строку `TLS_REQCERT allow`, которая указывает клиентским утилитам LDAP игнорировать факт использования самоподписанного сертификата. Если вы не добавите эту строку, а будете использовать параметры по умолчанию, сертификат будет считаться недействительным.

Настроить OpenLDAP на использование нового ключа и сертификата несложно. Предполагая, что сгенерированные ключи хранятся в папке /etc/openldap/ssl/, строки в листинге 17 настраивают ваш сервер на использование TLS-подключений после перезапуска slapd.

Листинг 17. Настройка slapd на использование SSL

```
TLSCertificateFile /etc/openldap/ssl/ldap.cert
TLSCertificateKeyFile /etc/openldap/ssl/ldap.key
```

Команды в листинге 17 указывают службе slapd местоположение сертификата и личного ключа. Чтобы проверить работу сервера после выполнения вышеуказанных настроек, выполните команду `ldapwhoami -v -x -Z`, которая анонимно привязывается к безопасному порту. Если вы получите сообщение "success", значит, все работает правильно. В противном случае опция -v поможет вам установить причину возникшей ошибки.

Таким же способом вы можете сгенерировать сертификат клиента, что не является обязательной процедурой. В этом случае вместо использования команд, приведенных в листинге 17, добавьте строки TLS_KEY и TLS_CERT с соответствующими значениями в файл ldap.conf. Эти строки указывают на местоположения ключа и сертификата, соответственно. Клиентские сертификаты необходимы только в тех случаях, когда вам требуется выполнять идентификацию клиентов на основе их сертификатов.

Вопросы, касающиеся работы брандмауэра

LDAP использует TCP-порт 389, а LDAPS (LDAP поверх SSL) – TCP-порт 636. Если между вашим сервером и клиентами работает брандмауэр, то для успешного подключения эти порты должны быть открыты. Клиенты всегда подключаются к серверам, а серверы могут подключаться к другим серверам в зависимости от вашей стратегии репликации.

Правила iptables в ОС Linux

Если для настройки брандмауэра на вашем сервере LDAP используются правила iptables, вам необходимо изменить их таким образом, чтобы позволить серверу принимать входящие подключения. Как правило, команд, приведенных в листинге 18, оказывается достаточно.

Листинг 18. Добавление правил iptables для разрешения подключений к LDAP

```
iptables -A INPUT -p tcp --dport 389 -j ACCEPT
iptables -A INPUT -p tcp --dport 636 -j ACCEPT
```

Команды листинга 18 работают, если у вас используется простая политика. Команда `-A INPUT` добавляет правило в таблицу INPUT, которая предназначена для проверки всех входящих пакетов. Вы можете добавить эти правила в начало списка (используя команду `-I INPUT` вместо `-A INPUT`) или использовать инструменты для работы с брандмауэром из состава вашего дистрибутива, чтобы открыть TCP-порт 389, а также порт 636 (если вам нужна функциональность LDAPS).

ваш брандмауэр Linux используется в качестве маршрутизатора (например, клиенты подключены к одному сетевому интерфейсу, а сервер LDAP – к другому), то вместо `INPUT` вы должны использовать цепочку `FORWARD` . Возможно, вы захотите определить интерфейс для входящих пакетов с помощью опции `-i`, например `-i eth0`, чтобы указать, что приниматься будут только пакеты, приходящие на интерфейс etho. Если пакет был принят, то также будут приняты и ответные пакеты.

Защита посредством использования оболочек TCP

Одной из опций конфигурирования, доступной при компиляции пакета OpenLDAP, является опция `--enable-wrappers`, которая связывает конечные исполняемые файлы с библиотеками оболочек TCP (TCP Wrappers). Для принятия или отклонения клиентских подключений оболочки TCP используют два файла: /etc/hosts.allow и /etc/hosts.deny соответственно.

Прежде всего, проверьте с помощью команды `ldd /usr/sbin/slapd | grep libwrap`, использует ли slapd оболочки TCP. Если вы получите какой-либо ответ на вышеуказанный запрос, значит, оболочки TCP используются. В противном случае

необходимо выполнить повторную компиляцию slapd с опцией `--enable-wrappers` или использовать правила iptables, как было описано выше.

Включив поддержку TCP Wrappers, вы можете запретить доступ к серверу LDAP для всех клиентов, добавив в файл /etc/hosts.deny строку `slapd: ALL`. После этого вы можете разрешить доступ с определенных IP-адресов, например, добавив строку `slapd: 192.168.1.,127.0.0.1`, которая разрешает подключаться любым клиентам сети 192.168.1.0/24 или любым локальным клиентам. Обратите внимание на то, что отклонение клиента оболочками TCP происходит следующим образом: сначала клиент подключается, а затем автоматически отключается. Этот процесс отличается от работы брандмауэра, когда пакет отбрасывается прежде, чем он достигнет службы slapd.

Формат файлов hosts.allow и hosts.deny позволяет разрешать и отклонять подключения многими различными способами. Для получения дополнительной информации обратитесь к man-руководству hosts_access(5).

Дополнительные сведения об аутентификации

До сих пор обсуждение вопросов аутентификации не выходило за рамки рассмотрения открытых паролей, определенных в файле slapd.conf, и методов простой аутентификации между клиентом и сервером. Проблема использования нешифрованных паролей решается с помощью команды `slappasswd`. Введите в командной оболочке команду `slappasswd`, после чего вам будет предложено ввести и подтвердить пароль. На выходе вы получите безопасный хэш пароля, такой как `{SSHA}oxmMsm9Ff/xVQ6zv+bgmMQjCUFL5x22+`. Математические методы гарантируют, что этот хэш необратим, хотя если кто-то получит его в свои руки, он может начать пробовать перебирать различные пароли и проверять, будет ли их хэш совпадать с исходным.

Вы уже имели дело с анонимной привязкой к каталогу, когда имя пользователя и пароль не указываются, а также с привязкой на основе аутентификации, когда клиент должен указать свои действующие имя пользователя и соответствующий ему пароль. Кроме этого OpenLDAP поддерживает привязку без аутентификации, когда указывается имя пользователя, но не указывается пароль. Привязка без аутентификации обычно отключена; для ее включения необходимо добавить в файл конфигурации строку `allow bind_anon_cred`. Если включена привязка без аутентификации, то все подключения, выполненные таким способом, считаются анонимными.

Альтернативу простой аутентификации составляет простая аутентификация и слой безопасности (Simple Authentication and Security Layer, SASL) – платформа для поддержки подключаемых модулей аутентификации и шифрования. Более подробно архитектура SASL будет рассмотрена в руководстве, которое должно скоро появиться, а пока ограничимся тем, что SASL предусматривает различные методы аутентификации – от открытых паролей до Kerberos.

При рассмотрении списков управления доступом в предыдущей части руководства было упомянуто, что доступ может быть предоставлен на основе метода подключения, определяющего *фактор уровня защиты* (Security Strength Factor, SSF). Нешифрованное соединение имеет SSF, равный 0, а шифрованному соединению обычно соответствует значение SSF, равное длине ключа шифрования. Таким образом, определенное правило ACL может содержать в условии who требование к уровню защиты подключения (строка `ssf=1`).

Оптимизация производительности сервера LDAP

OpenLDAP является базой данных. Вы обращаетесь к OpenLDAP с целью выполнения запроса или задания, после чего OpenLDAP ищет данные и возвращает их вам. Чтобы этот процесс занимал как можно меньше времени, вы должны распределить ресурсы так, чтобы они использовались наиболее эффективно, например, настроить кэширование часто используемой информации и индексацию баз данных.

Оценка производительности

Прежде чем приступать к оптимизации slapd, вы должны оценить его текущую производительность. Для этого можно измерять время выполнения определенной операции в вашем приложении, пытаясь затем найти способ улучшить результат, или же можно вручную выполнить несколько запросов и оценить среднее время их выполнения. Измерять можно не только временные показатели; можно, например, попытаться уменьшить загрузку жесткого диска, если текущая конфигурация сервера LDAP приводит к большому количеству операций чтения и записи.

И в том, и в другом случае полезно произвести несколько измерений различных показателей до и после изменений конфигурации. В этом вам могут помочь следующие команды:

- `vmstat` – показывает статистику ввода/вывода и загрузку центрального процессора, а именно, время, затрачиваемое на выполнение процессов пользователя, и время ожидания
- `iostat` – показывает более подробную информацию об операциях чтения и записи на жесткий диск, а также о загрузке дискового контроллера
- `ps` – показывает статистику использования памяти процессом slapd (использование большого объема памяти само по себе не является чем-то плохим, но важно убедиться, что вы не используете больше ОЗУ, чем установлено в системе)
- `time` – команда для временной оценки различных операций командной строки

Оптимизация работы демона

Оптимизация всегда требует принятия компромиссных решений. Часто вы увеличиваете объем системных ресурсов (обычно это оперативная память или жесткие диски), выделяемый под определенный процесс, чтобы он выполнялся быстрее. Это приводит к тому, что под остальные процессы выделяется меньше ресурсов. Точно так же, если определенный процесс выполняется быстрее, он зачастую потребляет больше ресурсов, таких как циклы центрального процессора или операции чтения/записи на диск, которые при этом становятся недоступными для остальных процессов.

Необходимость находить компромиссы может существовать и на уровне приложений. Пожертвовав некоторой производительностью операций записи, вы в большинстве случаев можете довольно существенно повысить производительность операций чтения. Также вы можете повысить быстродействие вашего приложения, отключив некоторые функции безопасности, такие как ведение журналов транзакций. При этом в случае сбоя вы можете столкнуться с тем, что вам придется восстанавливать вашу базу данных из резервной копии; однако только вам предстоит решить, насколько приемлемым является этот компромисс.

Большинство людей используют базу данных Berkeley Database (BDB), которая основана на быстрой встроенной БД Sleepycat Berkeley Database, в настоящее время принадлежащей

корпорации Oracle. Эта база данных не поддерживает язык запросов, а основана на поиске записей в хэш-таблицах. Оптимизация BDB может производиться в двух файлах: файл конфигурации slapd.conf и специальный файл, используемый исполняемыми модулями BDB.

Конфигурационные директивы файла slapd.conf

База данных BDB не является отдельной серверной службой, как большинство SQL-серверов, а линкуется вместе с исполняемыми файлами, которые ее используют. По существу, за некоторые аспекты работы базы данных BDB отвечает приложение, использующее ее. Все директивы, которые могут использоваться в файле slapd.conf, описаны в man-руководстве slapd-bdb(5); мы же рассмотрим только наиболее важные из них.

Так же, как и многие другие базы данных SQL, базы данных BDB записывают все изменения в журналы транзакций для обеспечения надежности, а также хранят данные в памяти для уменьшения количества операций записи на диск. Операция, которая выгружает содержимое памяти на жесткий диск и выполняет запись в журнал транзакций, называется *контрольной точкой*. Команда `checkpoint` указывает slapd, как часто нужно выполнять выгрузку данных на диск, оперируя такими параметрами, как количество килобайт данных и количество минут, прошедших с момента последней контрольной точки. Добавление строки `checkpoint 128 15` в файл slapd.conf означает, что данные будут выгружаться при достижении объема в 128 килобайт или, как минимум, каждые 15 минут. По умолчанию никакие операции с контрольными точками не выполняются, что равносильно использованию команды `checkpoint 0 0`.

Записи, к которым происходят частые обращения, могут кэшироваться в ОЗУ для ускорения доступа к ним. По умолчанию кэшируются 1000 записей. Для изменения этого значения используйте команду `cachesize` с указанием количества записей. Чем больше значение параметра `cachesize`, тем больше вероятность того, что запись будет помещена в оперативную память, однако, при этом slapd расходует большее количество оперативной памяти. Выбор значения этого параметра зависит от того, сколько различных записей содержится в дереве каталога, а также от шаблона доступа. Убедитесь, что объема оперативной памяти достаточно для того, чтобы уместить элементы (такие как список пользователей), обращения к которым происходят наиболее часто.

Команда `cachesize` аналогична команде `idlcachesize` и определяет, сколько памяти отводится под кэширование индексов. Выбор значения этого параметра зависит от того, сколько индексов у вас определено (это будет обсуждаться позже), но разумно указывать то же самое значение, что и для параметра `cachesize`.

Оптимизация баз данных BDB

Как было замечено ранее, некоторые конфигурационные параметры BDB содержатся в отдельном файле, который считывается исполняемыми модулями программы и игнорируется службой slapd. Этот файл называется DB_CONFIG и расположен в той же папке, что и ваша база данных. Самым важным параметром в этом файле является параметр `set_cachesize`, который устанавливает размер внутреннего кэша BDB (отдельно от кэша slapd). Эта команда имеет следующий формат: `set_cachesize <GigaBytes> <Bytes> <Segments;>`. Параметры GigaBytes и Bytes относятся к размеру кэша (эти два параметра суммируются), а параметр Segments позволяет вам разделять кэш между отдельными блоками памяти для преодоления ограничений 32-разрядной адресации (значения этого параметра, равные как 0, так и 1, работают одинаково, позволяя использовать только один сегмент памяти). Чтобы задать кэш

размером в 1 Гб, используйте команду `set_cachesize 1 0 0`.

Самый простой способ определить оптимальный размер кэша BDB – это посмотреть статистику использования кэша в работающей системе и при необходимости увеличить его размер. Статистику использования кэша BDB можно посмотреть с помощью команды `db_stat -h /path/to/database -m`. Наиболее значимая информация содержится в первых 20 строках вывода этой команды. Если из кэша удаляется большое количество страниц или количество страниц, найденных в кэше, составляет менее 95% от общего количества, подумайте об увеличении его размера. В некоторых дистрибутивах команда `db_stat` может называться `slapd_db_stat`, что позволяет обособить ее от системных библиотек и инструментов BDB.

Кроме оптимизации кэша вам необходимо обеспечить наблюдение за журналами транзакций. Укажите путь к журналам транзакций в качестве значения параметра `set_lg_dir`. Если вы сможете разместить базу данных и журнал транзакций на отдельных физических дисках или дисковых массивах, быстродействие системы существенно повысится.

Несмотря на то, что BDB является простой базой данных, существует необходимость блокировок файлов для выполнения операций записи. Обычно количество блокировок по умолчанию достаточно велико, но вам следует отслеживать максимальное количество используемых блокировок с помощью команды `db_stat -h /path/to/database -c`. Блокировки BDB делятся на три типа, для каждого из которых ведется отдельная статистика: lockers, locks и lock objects. Разница между этими тремя типами несущественна, тем не менее, максимальное количество блокировок каждого типа задается с помощью трех отдельных параметров – `set_lk_max_lockers`, `set_lk_max_locks` и `set_lk_max_objects` соответственно.

Каждый раз, когда вы вносите изменения в файл DB_CONFIG, вы должны перезапустить slapd. В листинге 19 приведен пример конфигурационного файла DB_CONFIG с использованием всех вышеупомянутых директив.

Листинг 19. Пример файла DB_CONFIG

```
# 256K cache
set_cachesize 0  268435456 0
set_lg_dir /var/log/openldap
set_lk_max_lockers 1000
set_lk_max_locks 1000
set_lk_max_objects 1000
```

Индексация базы данных

Большинство операций LDAP выполняют поиск определенных атрибутов, таких как имя пользователя, телефонный номер или адрес электронной почты. Без использования дополнительных средств slapd должен выполнять поиск, перебирая каждую запись. При добавлении индекса к определенному атрибуту создается специальный файл, который позволяет slapd находить данные гораздо быстрее. Недостатками использования индексов являются более низкая скорость записи в базу данных, а также увеличение загрузки жесткого диска и оперативной памяти. По этой причине лучше всего индексировать только те атрибуты, обращения к которым происходят чаще всего.

В зависимости от типа выполняемого поиска в OpenLDAP поддерживаются различные типы индексов. Все они перечислены в таблице 4.

Таблица 4. Типы индексов OpenLDAP

Тип	Ключевое слово	Описание	Пример запроса
Наличие (Presence)	pres	Используется для выяснения существования атрибута.	`uid=*`
Эквивалентность (Equality)	eq	Используется для нахождения определенного значения.	`uid=42`
Вхождение (Substring)	sub	Используется для нахождения строки, содержащейся в значении атрибута. Помимо основного типа sub вы можете использовать три его подтипа, оптимизированные для различных условий.	`cn=Sean*`
	subinitial	Индекс вхождения, использующийся для нахождения строки, содержащейся в начале значения атрибута.	`cn=Sean*`
	subany	Индекс вхождения, использующийся для нахождения строки, содержащейся в середине значения атрибута.	`cn=*jone*`
	subfinal	Индекс вхождения, использующийся для нахождения строки, содержащейся в конце значения атрибута.	`cn=*Smith`
Подобие (Approximate)	approx	Используется для нахождения значений, звучащих подобно указанной строке поиска.	`cn~=Jason`

Чтобы создать индекс для атрибута, используйте следующий синтаксис: `index [attrlist] [indices]`, где `[attrlist]` – это разделенный запятыми список атрибутов, а `[indices]` – разделенный запятыми список типов индексов, перечисленных в таблице 4. Можно использовать несколько строк с директивами `index`. Ключевое слово `default` определяет список типов индексов по умолчанию, который будет использоваться в том случае, когда опущен параметр `[indices]`. Ознакомьтесь с индексами, определенные в листинге 20.

Листинг 20. Примеры индексов

```
index default eq,sub
index entryUUID,objectClass eq
index cn,sn,mail eq,sub,subinitial,subany,subfinal
index userid,telephonenumber
index ou eq
```

В первой строке листинга 20 определен список типов индексов по умолчанию, включающий в себя индексы эквивалентности и вхождения. Во второй строке создаются индексы эквивалентности для атрибутов `entryUUID` (полезно для повышения производительности syncrepl) и `objectClass` (для общего поиска). В третьей строке создаются индекс эквивалентности и все типы индексов вхождения для атрибутов `cn`, `sn` и `mail`, поскольку

по этим полям часто выполняется поиск с использованием различных специальных символов. Для атрибутов `userid` и `telephonenumber` создаются индексы по умолчанию, поскольку не указаны никакие дополнительные параметры. Наконец, для атрибута `ou` создается индекс эквивалентности.

После определения индексов вы должны перестроить их, остановив slapd и выполнив команду `slapindex` от имени пользователя ldap (если вы работаете под учетной записью root, не забудьте назначить пользователя ldap владельцем всех файлов, расположенных в каталоге базы данных, после выполнения команды `slapindex`). Запустите slapd, после чего ваши индексы начнут использоваться.

Конфигурирование

Содержимое файла slapd.conf было широко рассмотрено ранее в этом руководстве, а также в предыдущем руководстве. Определенный интерес представляют параметры командной строки и команды для работы с файлами журналов slapd.

Параметры командной строки

Запуск slapd без каких-либо аргументов является наиболее простым. При таком запуске slapd считывает конфигурационный файл по умолчанию, переходит в фоновый режим работы и освобождает терминал.

В таблице 5 перечислены некоторые полезные параметры командной строки.

Таблица 5. Параметры командной строки slapd

Параметр	Значение	Описание
`-d`	Целое число	Запускает slapd в режиме расширенной отладки; при этом slapd работает на переднем плане.
`-f`	Имя файла	Указывает альтернативный конфигурационный файл.
`-h`	Список URL	Указывает IP-адреса и порты, на которых работает slapd.
`-s`	Уровень отладочной информации syslog	Указывает уровень syslog, использующийся для вывода отладочной информации.
`-l`	Целое число	Указывает локальное имя источника syslog (такое как local4), используемое для вывода отладочной информации.
`-u`	Имя пользователя	Запускает slapd от имени указанного пользователя.
`-g`	Имя группы	Запускает slapd от имени указанной группы.

Список URL позволяет вам привязывать slapd к нескольким различным интерфейсам. Например, команда `-h "ldap://127.0.0.1/ ldaps:///"` указывает, что slapd будет прослушивать TCP-порт 389 (нешифрованные подключения LDAP) только на loopback-интерфейсе, а TCP-порт 636 (защищенные подключения) – на всех интерфейсах. Вы также можете изменять номера портов: например, команда `ldap://:5565/` указывает, что slapd будет прослушивать TCP-порт 5565 (нешифрованные подключения) на всех интерфейсах.

Регистрация событий

Для регистрации событий slapd использует демон ОС Unix под названием syslog. По умолчанию все сообщения посылаются от имени источника `LOCAL4`. По этой причине вам необходимо добавить в файл syslog.conf, как минимум, строку `local4.* /var/log/openldap.log` для записи сообщений в файл /var/log/openldap.log. Далее, команда `loglevel` в файле slapd.conf указывает slapd, какие типы сообщений необходимо регистрировать. Эти типы перечислены в таблице 6.

Таблица 6. Типы регистрируемых сообщений slapd

Ключевое слово	Соответствующее целочисленное значение	Описание
`trace`	1	Трассировать вызовы функций
`packet`	2	Отладка обработки пакетов
`args`	4	Тщательная отладочная трассировка
`conns`	8	Управление соединением
`BER`	16	Печать принятых и отправленных пакетов
`filter`	32	Обработка фильтра поиска
`config`	64	Обработка конфигурационного файла
`ACL`	128	Обработка списка контроля доступа
`stats`	256	Регистрировать статистику соединения/обработки/результатов
`stats2`	512	Регистрировать статистику отправленных элементов
`shell`	1024	Печать коммуникаций с shell механизмом баз данных
`parse`	2048	Печать отладки анализа элемента
`sync`	16384	Репликация LDAPSync

Для параметра `loglevel` вы можете использовать список ключевых слов или целочисленных значений, разделенных пробелами, а также сумму целочисленных значений. Например, каждая из команд `loglevel args ACL`, `loglevel 4 128` и `loglevel 132` включает регистрацию тщательной отладочной трассировки и обработки списков ACL.

Резюме

Из этого руководства вы узнали о списках контроля доступа, репликации, безопасности, оптимизации, а также о других основных аспектах конфигурирования LDAP.

Списки контроля доступа (ACLs) определяют, кому и к каким элементам предоставляется доступ определенного уровня. Для настройки списков контроля доступа вы должны использовать следующий синтаксис: `access to <what> [ by <who> [ <access> ] [ <control> ] ]+`. Для определения условия what вы можете использовать различные конструкции, включая прямые соответствия и регулярные выражения. Для определения условия who вы можете использовать такие ключевые слова, как `self`, `users` и `anonymous` в дополнение к вышеупомянутым конструкциям. Кроме того, в условии who можно определять такие параметры, как фактор уровня защиты подключения или номер сети, из которой подключается пользователь.

Репликация позволяет обеспечить идентичность данных удаленного и главного серверов LDAP. Существует два метода репликации: slurpd и syncrepl. При выполнении репликации посредством slurpd на главном сервере работает отдельный демон, передающий все изменения на подчиненные серверы. Подчиненные серверы должны запускаться только тогда, когда на них имеется копия данных главного сервера, что приводит к простоям последнего. При выполнении репликации посредством syncrepl на источнике данных (главный сервер) выполняется оверлейная программа, обрабатывающая задачи репликации. Получатели данных (подчиненные серверы) подключаются к источнику и скачивают все изменения. Если получатель скачивает изменения периодически, он работает в режиме refreshOnly. Если же после загрузки обновлений получатель не разрывает соединение, то он работает в режиме refreshAndPersist и продолжает получать обновления, выполняющиеся на сервере-источнике.

Протоколы TLS и SSL позволяют устанавливать шифрованные соединения между клиентом и сервером, а также шифровать трафик репликации. Для использования TLS вы должны сгенерировать ключи сервера, а затем подписать их в центре сертификации. Для передачи обычного трафика LDAP используется TCP-порт 389, а для передачи шифрованного трафика LDAP – TCP-порт 636; с учетом этого должны быть соответствующим образом настроены ваши брандмауэры.

Оптимизация производительности включает в себя распределение системных ресурсов под различные области для промежуточного хранения данных, а также использование индексов для наиболее часто используемых атрибутов. Управление системными ресурсами производится посредством редактирования двух файлов: slapd.conf и DB_CONFIG. В зависимости от задач оптимизации поиска, существуют индексы следующих типов: индексы наличия, эквивалентности, вхождения и подобия.

Большинство параметров работы slapd задается в файле slapd.conf, поэтому существует лишь несколько опций командной строки; эти опции задают номера портов, на которых работает slapd, имя пользователя, в контексте которого он работает, а также некоторые параметры регистрации событий. Информация, которую slapd регистрирует в журналах событий, определяется директивой `loglevel` в файле slapd.conf.

На данном этапе вы обладаете достаточными знаниями для того, чтобы установить, настроить и управлять работающим сервером OpenLDAP, а также разбираться в вопросах защиты, репликации и оптимизации производительности. В следующих двух руководствах будут рассмотрены такие приложения LDAP, как интеграция LDAP с почтовой системой и системой аутентификации, и выполнение поиска в дереве каталога из командной строки.