OpenLDAP и защита данных

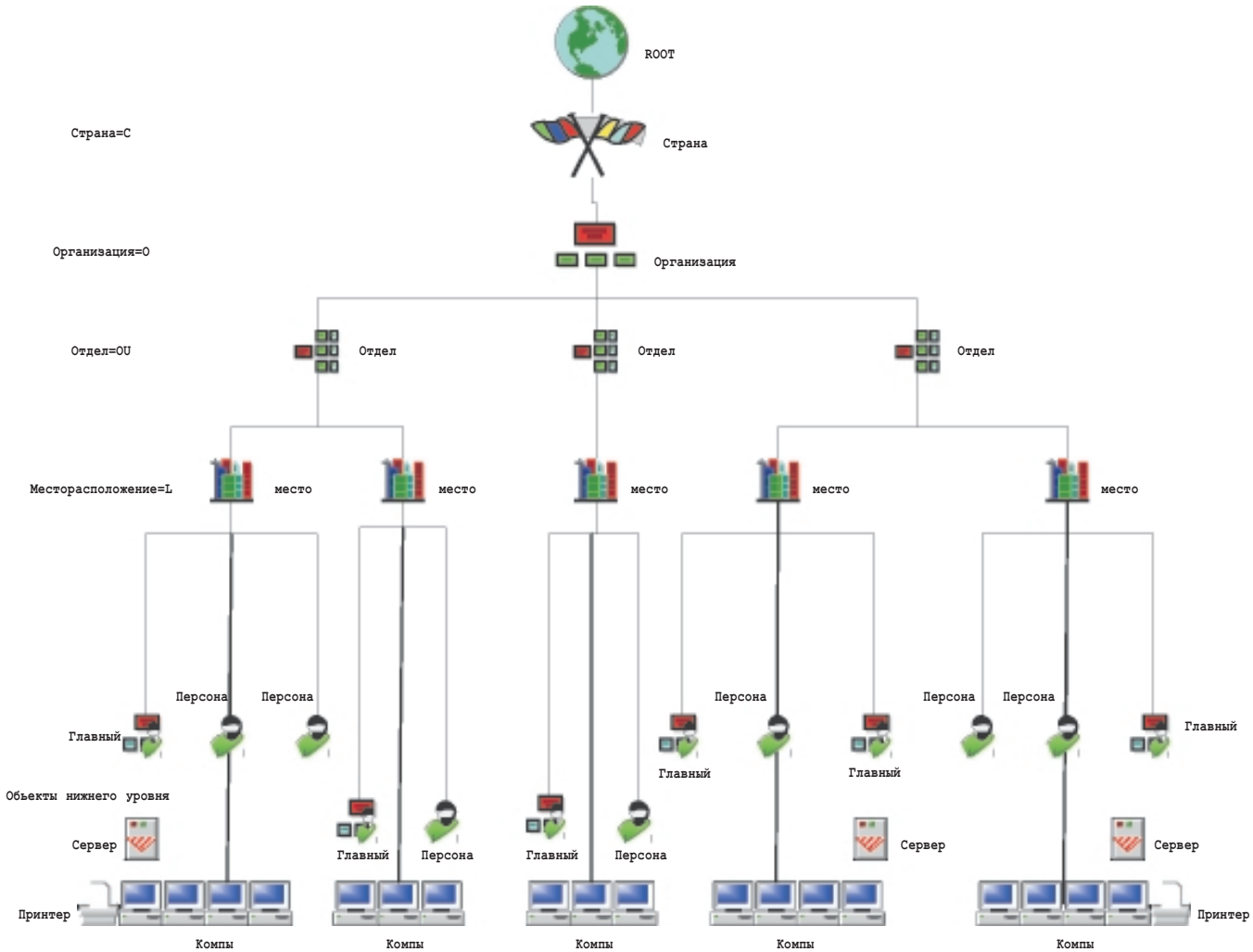
В настоящее время при администрировании крупных сетей часто используется система LDAP. Но документации на русском по ней практически нет, а документация на английском очень уж «размазана». Поэтому я решил полностью описать механизмы настройки сервера OpenLDAP и его взаимодействия с уже существующими сетевыми сервисами...

ВСЕВОЛОД СТАХОВ

При администрировании сети на определённом этапе возникает проблема централизованного контроля аутентификации. Это особенно актуально для крупных сетей (100 и более пользователей). Раньше единственным приемлемым выходом в такой ситуации была установка Novell Netware как центрального сервера или службы NIS, если сеть основана на *nix машинах, что бывает довольно редко. У Netware (версии 4 и старше) существует очень грамотный механизм NDS (служба директорий Novell), позволяющий создать единое дерево сети, в котором каждый компонент сети является объектом службы директорий. Причём каждый объект обладает определёнными параметрами для его идентификации. Например, объект «пользователь» имеет своё имя, телефон, отдел, служебное положение, скрипт для входа в систему. Каждый пользователь обладает определёнными правами, некоторые из которых наследуются из прав группы, которой данный пользователь принадлежит. При входе в систему каталогов пользователь получает доступ только к «своим» файлам. При этом сама система каталогов может находиться на нескольких серверах. Конечно, здесь очень много нюансов и очень много полезных возможностей, но я перейду сразу же к основной теме: службе директорий, работающей в TCP/IP-сетях – LDAP (Lightweight Directory Access Protocol).

Теория и терминология

Раньше в *nix существовала только одна возможность централизованной аутентификации – службы NIS, работающие через UDP. LDAP построен по объектно-ориентированному принципу и позволяет с лёгкостью добавлять и изменять объекты при помощи файлов схем. Кроме этого, LDAP обладает большими возможностями в плане структурированности и работы с клиентами различных платформ. Множество приложений поддерживают LDAP через PAM, но об этом далее. На основе LDAP легко построить гетерогенные сети, кроме этого, информация об объектах хранится в древовидной структуре. Например, вот как может выглядеть дерево для крупной организации, разбросанной по нескольким странам с дочерними фирмами:



Объекты верхнего уровня(C, O, OU, DC, L) являются контейнерными, т.е. Могут содержать другие объекты. Объекты нижнего уровня обычно такой особенностью не обладают. Приведённая схема жестко не закрепляет иерархию объектов, дерево может строиться в различном порядке. При использовании интернет-дерева может использоваться иерархия определений dc. Зачастую также применяют смешанные схемы стандартной и интернет нотификации. Для выбора конкретной ветви используют термин dn поиска - верхний объект, от которого начинается поиск.

Здесь можно сразу же определиться с сокращениями, принятыми в службе директорий:

- C(ountry) – страна;
- ST(ate) – имя региона, области или штата;
- L(ocation) – имя города, посёлка, деревни;
- O(rganization) – имя компании;
- O(rganizational)U(nit) – раздел компании.

Данные объекты дерева являются основополагающими, т.к. являются контейнерами для других объектов. На их основе обычно строится дерево каталогов. Самый верхний уровень представляет объект root (и здесь рут!), от него происходят скелетные объекты, внутри которых и заключаются конкретные объекты-листья дерева. Если представить себе дерево, то скелетные объекты-контейнеры будут играть роль веток, а обычные объекты – роль листьев. Вообще, древовидная структура имеет множество преимуществ по сравнению с линейной (/etc/passwd), т.к. позволяет точно определить нахождение пользователя или иного объекта. /etc/passwd предоставляет лишь жалкое подобие данной модели – поле GECOS, но работая с LDAP, просто необходимо указывать полное описание объекта. Для обозначения полного описания (+имени) объекта относительно скелета дерева используется термин DN (distinguished name – назначенное имя, контекст). На этом позвольте завершить описание теории и перейти к практике...

Общая настройка сервера slapd

Займёмся самым интересным: настройкой сервера. Я использовал OpenLDAP и всё нижесказанное относится только к данному пакету LDAP.

Для начала необходимо иметь следующие вещи (я предполагаю установку из пакетов, а не из сырцов; установка из сырцов также не должна вызвать затруднений, но у меня оных просто не было):

- библиотеки LDAP;
- сервер LDAP (slapd);
- pam_ldap и nss_ldap для аутентификации через LDAP.

После чего надо настроить сервер на работу в вашей службе каталогов. Открываем файл конфигурации /etc/[open]ldap/slapd.conf (при установке из исходников /usr/local/etc/openldap). Он содержит приблизительно следующее:

```
# Примерный файл конфигурации сервера slapd.conf.
include    /usr/share/openldap/schema/core.schema
include    /usr/share/openldap/schema/cosine.schema
include    /usr/share/openldap/schema/corba.schema
include    /usr/share/openldap/schema/inetorgperson.schema
include    /usr/share/openldap/schema/java.schema
include    /usr/share/openldap/schema/krb5-kdc.schema
include    /usr/share/openldap/schema/kerberosobject.schema
include    /usr/share/openldap/schema/misc.schema
include    /usr/share/openldap/schema/nis.schema
include    /usr/share/openldap/schema/openldap.schema
# Включаем базовые определения объектов директорий и дополнительные классы
# объектов, файлы схем.
# Включаем файл, описывающий права доступа к БД:
include    /etc/openldap/slapd.access.conf
# Стандартные файлы, необходимые для работы демона: файл идентификатора и
# командной строки.
pidfile    /var/run/ldap/slapd.pid
argsfile   /var/run/ldap/slapd.args
# Путь к дополнительным модулям сервера.
modulepath /usr/lib/openldap
# Описываем параметры соединения через SSL.
# Хост для безопасной аутентификации SASL.
sasl-host localhost
# Если вы хотите разрешить соединение через TLS то уберите комментарий из
# следующих строк (в данном примере он есть).
#TLSRandFile            /dev/random
#TLSCipherSuite         HIGH:+SSLv2
#TLSCertificateFile     /etc/openldap/ldap.pem
#TLSCertificateKeyFile  /etc/openldap/ldap.pem
# Путь к файлу ключа
#TLSCACertificatePath   /etc/openldap/
#TLSCACertificateFile   /etc/openldap/ldap.pem
# Путь к сертификату хоста
#TLSVerifyClient 0
# Проверка ключа клиента

# Самое главное - описание базы данных LDAP
database   ldbm
# Тип базы данных ldbm
suffix           "dc=test,dc=ru"
#suffix          "o=My Organization Name,c=US"
# Это основной суффикс БД. В определении системы директорий существует так
# называемый корневой объект root. Суффикс определяет этот объект. Вообще
# существует 2 стандарта LDAP имен для скелета дерева: метод для глобальных
# сетей и локальных. Первый имеет подобный вид и описывает URL адрес:
# person.domain.com(cn=person,dc=domain,dc=com), а второй описывает
# организацию(вообще-то этот вид является стандартным) и имеет следующий вид:
# person.organization_unit.organization.country(cn=person,ou=otd1,o=lab,c=RU)
# выбор метода зависит от конкретного назначения LDAP и особого значения не имеет.
# В данной статье я использовал интернет-наименования для краткости, но обычно в
# компаниях всё же чаще применяется второй способ построения скелета дерева.
rootdn           "cn=admin,dc=test,dc=ru"
#rootdn          "cn=Manager,o=My Organization Name,c=RU"
# Это поле содержит пароль администратора объекта root(то есть всей БД LDAP).
# Пароль может храниться как в открытом виде (не рекомендуется!), так и в
# зашифрованном виде с указанием алгоритма шифрования ({crypt}, {MD5}, {SSHA},
# {crypt}, {SMD5}, {SHA}) Пароль желательно устанавливать программой slappasswd,
# позволяющей установить нужный алгоритм шифрования.
 rootpw          secret
# rootpw          {crypt}ijFYNcSNctBYg
# Папка, где хранится база данных LDAP. Эта папка должна существовать до запуска
# LDAP, который иначе не запустится. Папка должна быть доступна для чтения и за-
# писи только пользователю, под которым работает slapd, и иметь маску доступа 700.
directory  /var/lib/ldap
# Определения первичных и вторичных индексов БД может ускорить поиск по БД
#index     objectClass                        eq
index      objectClass,uid,uidNumber,gidNumber eq
index      cn,mail,surname,givenname          eq,subinitial
# Права доступа по умолчанию
access to attr=userPassword
          by self write
          by anonymous auth
          by dn="uid=root,dc=test,dc=ru" write
          by * none
# Здесь определяется доступ к атрибуту пароль пользователя. Сам пользователь имеет
# право записи, анонимному пользователю предоставляется возможность пройти
# аутентификацию (после этого он представляет уже другой объект и доступа к паро-
# лям анонимного пользователя не происходит, как можно было бы подумать), а поль-
# зователь с контекстом "uid=root,ou=people,dc=test,dc=ru" имеет право на запись.
# Другие же пользователи доступа к паролю не имеют никакого. Т.е., другими словами,
# никто, кроме администратора и самого пользователя не имеют доступа к паролю.
access to *
          by dn="uid=root,dc=test,dc=ru" write
          by * read
# Доступ к остальным полям базы LDAP - все могут читать атрибуты (кроме пароля,
# так как запрет имеет более высокий приоритет), а пользователь с контекстом
# root может писать всё что угодно (а кто ж ему помешает ).
```

Добавлю, что в этом файле можно ещё указывать дополнительные параметры для взаимодействия нескольких серверов (реплик): в частности, основной сервер, вторичные сервера, пароли доступа к ним и т. д. Но это уже другая тема. В основном необходимо сделать 3 вещи:

- установить суффикс БД и выбрать тип контекста (глобальной сети или организации);
- указать контекст администратора LDAP;
- установить пароль администратора LDAP с помощью программы slappasswd.

Чтобы установить алгоритм шифрования пароля, запускайте программу slappasswd так:

```
slappasswd -h {алгоритм_шифрования},
```

вместе с символами {}, например:

```
slappasswd -h {crypt}
```

или

```
slappasswd -h {md5}
```

Вот и всё: сервер может хоть сейчас начинать работать! Но толку пока от него никакого: ведь мы не добавили объектов. Сейчас я расскажу, как создавать базу данных LDAP и проводить аутентификацию через неё.

Создание базы данных

Для первоначальной настройки неплохо было бы объяснить общий синтаксис файлов для создания базы данных (я обозначаю комментарии символом #, но в реальном файле этих комментариев быть не должно!):

```
dn: dc=test,dc=ru
# Уникальный контекст имени
objectclass: dcObject
# Класс объекта - контейнерный объект
objectclass: organization
# Тот же самый контекст может пред-
# ставлять собой различные объекты
o: test
# Имя организации
dc: test
# То же самое, но в системе
# глобальных имён
# ---------------------------------
dn: cn=admin,dc=test,dc=ru
# Контекст администратора
objectclass: organizationalRole
# Класс - должностное лицо
cn: admin
# Имя человека (псевдоним)
```

```
#----------------------------------
dn: ou=users,dc=test,dc=ru
# Контекст группы пользователей
ou: users
# Значение группы
objectclass: top

objectclass: organizationalUnit
# Класс группа
#----------------------------------
dn: uid=null,ou=users,dc=test,dc=ru
# Контекст конкретного пользователя
# из /etc/passwd
uid: null
# Его пользовательский псевдоним
cn: Neo
# Реальный псевдоним
objectclass: account
# Класс пользовательский профиль
objectclass: posixAccount
# Класс пользователя POSIX
objectclass: top

objectclass: uidObject
# Самая лучшая оболочка - zsh!
loginshell: /bin/zsh
# Все следующие параметры совпадают
# с аналогичными в /etc/passwd
uidnumber: 1000

gidnumber: 1000

homedirectory: /home/null

gecos: Neo

userpassword: $1$abcdvPbaLa6vs4ABab1N
```

Здесь, думаю, всё понятно. Но ведь если у вас система уже настроена и содержит порядка $8*10^{10}$ пользователей, то будет немного трудновато всё это заново переписывать. Поэтому во многих дистрибутивах есть пакет openldap-migration, содержащий набор скриптов для переноса существующих файлов и баз сетевых служб в LDAP. Если его нет, то посмотрите здесь: http://www.padl.com. Данный пакет представляет из себя набор скриптов на Perl, обрабатывающий соответствующие файлы и службы (/etc/passwd (+shadow), /etc/hosts, /etc/profile, /etc/services, NIS-службы). Использование скриптов предельно просто. Вначале надо исправить файл /usr/share/openldap/migration/migrate_common.ph:

```
# Почтовый домен по умолчанию
$DEFAULT_MAIL_DOMAIN = "test.ru";
# Имя корневого объекта LDAP
$DEFAULT_BASE = "dc=test,dc=ru";
# Хост для приёма/передачи почты
# по умолчанию
$DEFAULT_MAIL_HOST = "mail.test.ru";
# Используем расширенные файлы схем
$EXTENDED_SCHEMA = 1;
```

Далее выполняем:

```
migrate_base.pl > base.ldif
```

для создания структуры базы LDAP. Для добавления данного файла к базе можно использовать следующий синтаксис:

```
ldapadd -x -D "cn=root,dc=test,dc=ru"
-W -f base.ldif
```

или же

```
slapadd -f base.ldif
```

для добавления записи в режиме offline. Первый способ лучше, так как позволяет проверить работу LDAP в сети (фактически используются TCP-сокеты).

Далее выполняем миграцию того, что нужно перенести в LDAP, например:

```
./migrate_hosts.pl /etc/hosts
hosts.ldif
```

Файл hosts.ldif будет содержать примерно следующее (в файле /etc/hosts было 192.168.1.23 work.test.ru work):

```
dn: cn=work.test.ru,ou=Hosts,
dc=test,dc=ru
objectClass: top
objectClass: ipHost
objectClass: device
ipHostNumber: 192.168.1.23
cn: work.test.ru
cn: work
```

Удобно, не так ли? Можно проделать подобное со всем, для чего уже написаны скрипты (можно и самому, в конце концов, написать! Что нам этот Perl). Кстати, вот ещё о чём я забыл: при переносе /etc/passwd нужно установить переменную окружения ETC_SHADOW:

```
root@ldap # ETC_SHADOW=/etc/shadow
./migrate_passwd.pl /etc/passwd
passwd.ldif
```

И ещё: не забывайте добавлять файлы .ldiff к базе с помощью ldapadd!

Если всего этого делать нет желания, то можно применить один из shell-скриптов migrate_all_[on|off]line, которые позволяют в интерактивном режиме перенести все существующие стандартные конфигурационные файлы в LDAP.

Ну вот, база создана, надо бы её проверить. Поищем в ней объекты с заданным атрибутом, пускай помучается:

```
ldapsearch -LL -H ldap://localhost
-b"dc=test,dc=ru" -x "(uid=null)"
#

# filter: (uid=null)

# requesting: ALL

#
# null, Users, test, ru

dn: uid=null,ou=Users,dc=test,dc=ru

uid: null

cn: Neo

objectClass: account

objectClass: posixAccount

objectClass: top

objectClass: uidObject

loginShell: /bin/zsh

uidNumber: 1000

gidNumber: 1000

homeDirectory: /home/null

gecos: Neo

# search result

search: 2

result: 0 Success

# numResponses: 1

# numEntries: 1
```

Естественно, что в поиске можно использовать регулярные выражения. «Ну и зачем мы это делали?» – закричат многие и будут неправы. На самом деле это уже хорошо: вся информация о системе хранится в древесном виде в БД. Там её можно искать, читать, писать, ставить нужные права. Идея просто супер! Причём всё это хозяйство без проблем достаётся из сети, реестр виндов отдыхает. Это сказка для любителей Novell и ещё один повод перейти на Linux даже в крупных сетях. Извините, оговорился, на *nix, так как openldap работает практически на всех *nix системах, а клиенты есть под любую ось. Ну а теперь разберёмся, как организовать связку с другими программами.

Настройка клиентов и аутентификации

Вначале нужно создать специального пользователя, который мог бы читать пароли других пользователей (это, конечно, плоховато для безопасности, но ведь пароль-то можно придумать какой-нибудь зловещий, например SHA1 хеш от результата crypt над строкой

$%YJH&*&*jszZn7867wey8YT6yeuwe%%&^&&*. Этот пользователь понадобится в будущем для клиентов LDAP, для выполнения аутентификации (ведь клиентам надо прочитать пароль, чтобы его сравнить с чем-то). Назовём этого пользователя proxy и сделаем следующее:

Добавим его в LDAP:

```
dn:  cn=proxy,dc=test,dc=ru
cn: prox
sn: proxy
objectclass: top
objectclass: person
userPassword:
{MD5}Xr4ilOzQ4PCOq3aQ0qbuaQ==
```

Затем предоставим ему права чтения пароля в sldap.conf:

```
access to attr=userPassword
 by self write
 by anonymous auth
 by dn="uid=root,dc=test,dc=ru" write
 by dn="cn=proxy,dc=test,dc=ru" read
   by * none
```

Далее настроим наших клиентов в файле /etc/ldap.conf для *nix клиентов, для других ОС необходимо смотреть документацию к софту, работающему с LDAP-сервером:

```
# Сервер LDAP - IP-адрес сервера или
# его имя, находящееся в /etc/hosts,
# или в DNS на удалённом узле
host 127.0.0.1
# Основной суффикс должен совпадать
# с суффиксом в slapd.conf
base  dc=test,dc=ru
# Это типа root, точнее, наш  горячо
# любимый proxy, сделано в целях
# безопасности и не позволяет самому
# пользователю изменить пароль.
rootbinddn  cn=proxy,dc=test,dc=ru
scope one
# Это фильтр для поиска
# пользовательских записей
pam_filter  objectclass=posixaccount
# Атрибут, определяющий имя
# пользователя и его группы
# для pam-модуля
pam_login_attribute  uid
pam_member_attribute  gid
pam_template_login_attribute  uid
# Тип пароля - шифрование UNIX crypt,
# кстати, шифрование LDAP и шифро-
# вание UNIX - несовместимы (crypt
# означает именно шифрование unix)!
pam_password crypt
# Базовые dn для поиска различных
# объектов, one - это область поиска.
# Данные параметры используются nss.
nss_base_passwd
ou=People,dc=test,dc=ru?one
nss_base_shadow
ou=People,dc=test,dc=ru?one
nss_base_group
ou=Group,dc=test,dc=ru?one
nss_base_hosts
ou=Hosts,dc=test,dc=ru?one
```

После этого вы должны создать файл, содержащий пароль коварного proxy в открытом(!) виде. Ну и конечно же, запретить чтение/запись всем, кроме root (chmod 0600 chown root:root). В принципе то же самое, что и /etc/shadow. Но беда, что пароль хранится в открытом виде. С другой стороны, получение пароля proxy само по себе ничего не даёт: нельзя менять пароли, но прочитать их можно (хотя толку мало, если пароли дикие). Но если proxy не может менять пароль, то никто не сможет изменить свой пароль, иначе как вы себе это представляете. Ну а дальше надо настраивать pam, чтоб он ходил на LDAP-сервер за паролями. Чтобы включить возможность аутентификации приложений через LDAP, необходимо установить модуль PAM pam-ldap. После этого можно включить в файлы /etc/pam.d (эти файлы содержат список способов аутентификации для определённых приложений: login, passwd, pop, smtp...) следующие строчки:

```
  auth       sufficient    /lib/
security/pam_ldap.so
  account    sufficient    /lib/
security/pam_ldap.so
  password   sufficient    /lib/
security/pam_ldap.so
  #(эта строчка нужна не везде, надо
  # смотреть, в каких файлах она уже
  # есть)
```

Особо надо отметить файл system-auth, здесь нужно указать некоторые вещи:

```
  #%PAM-1.0
  auth required /lib/security/
pam_env.so
  auth sufficient /lib/security/
pam_unix.so likeauth nullok
  auth sufficient /lib/security/
pam_ldap.so use_first_pass
  auth required /lib/security/
pam_deny.so

  account required /lib/security/
pam_unix.so
  account [default=bad success=ok
user_unknown=ignore \
  service_err=ignore
system_err=ignore] /lib/security/
pam_ldap.so

  password required /lib/security/
pam_cracklib.so retry=3
  password sufficient /lib/security/
pam_unix.so nullok use_authtok \
  md5 shadow
  password sufficient /lib/security/
pam_ldap.so use_authtok
  password required /lib/security/
pam_deny.so

  session required /lib/security/
pam_limits.so
  session required /lib/security/
pam_unix.so
  session optional /lib/security/
pam_ldap.so
```

Модуль LDAP является обычно дополнительным компонентом и не является необходимым, и поиск идёт вначале в локальных файлах, а потом уж в ldap (так как стоит он после других способов аутентификации). Таким образом, можно добавить аутентификацию через ldap в любой модуль pam, а последний используется множеством приложений для аутентификации клиента.

Ещё одной интересной возможностью ldap является возможность проверки хоста. Часто нельзя, чтобы любой человек, занесённый в базу ldap, мог пользоваться вашим компом (мало ли чего, вдруг это начальник). Поэтому можно добавить к учётным записям пользователей хосты, на которые эти пользователи могут логиниться (повторяю: не откуда может поступать запрос, а куда они могут входить, то есть нельзя удалённо пройти аутентификацию на машине X, если это не разрешено). Для этого необходимо добавить к каждой записи пользователя следующие атрибуты:

```
  host: name_of_host1
  host: name_of_host2
    ...
  host: name_of_hostn
```

Для модификации базы данных можно воспользоваться программой ldapmodify:

```
  ldapmodify -H ldap://localhost -D
"cn=root,dc=test,dc=ru" -x -W -f host-
auth.ldif
```

А сам файл host-auth.ldif должен содержать следующее:

```
  dn: uid=user_name, ou=People,
dc=test, dc=ru
  changetype: modify
  add: host
  host: name_of_host1
  host: name_of_host2
  host: name_of_hostn
```

Придётся повторить это для каждого пользователя! Разумнее написать скрипт, но я возложу это на ваши могучие плечи. После этого надо сделать ещё изменения в /etc/ldap.conf. В данном файле необходимо указать, что нужно ещё смотреть атрибут host:

```
  pam_check_host_attr  yes
```

Защита LDAP при помощи SSL

Как я уже говорил, в LDAP существует возможность защиты данных, передаваемых по сети. При этом используется два метода: TLS и SASL. Первый из них не меняет порта, на котором слушает LDAP (336), а просто организует аутентификацию асимметрическим шифрованием, SASL же меняет порт LDAP на ldaps:// и соединение идёт по другому механизму: через туннель SASL. TLS намного проще в настройке, поэтому я расскажу именно о нём. Для начала надо сгенерировать серверную пару ключей асимметрического шифрования. Для этого в Linux удобно воспользоваться единым центром сертификации OpenSSL (об этом я уже писал на страницах январского номера):

- создаем rsa ключ длиной 1024 бита и сохраняем его в файле ldap.key:

```
  $ openssl genrsa -out ldap.key
1024
```

- создаём запрос на сертификацию:

```
  $ openssl req -new -config .cfg
-key ldap.key -out ldap.csr
```

- создаём сертификат, по которому будем доверять (CACertificate); вначале делаем ключ длиной 2048 бит:

```
  $ openssl genrsa -des3 -out
ca.key 2048
```

- создаём self-signed сертификат сроком действия на год на основе сгенерированного ключа:

```
  $ openssl req -new -x509 -days
365 -key ca.key -out ca.cert
```

- теперь на основе созданного для LDAP ключа и доверенного сертификата создаём сертификат LDAP:

```
  $ openssl x509 -req -in
ldap.csr -out ldap.cert -CA
ca.cert -CAkey ca.key -
CAcreateserial \
         -days 365 -extfile .cfg
```

При этом файл конфигурации .cfg должен содержать следующие расширения:

```
  [ v3_req ]
  subjectAltName    = email:copy
  basicConstraints  = CA:false
  nsComment  = "LDAP server certifica-
te"
  nsCertType = server
```

Немного страшновато выглядит, но в ходе всех этих действий создаются два сертификата: доверенный сертификат и подписанный им сертификат сервера. Таким образом, сервер LDAP сможет проверить правильность своего сертификата через доверенный сертификат. Кстати, если у вас уже есть доверенные сертификаты, то можно воспользоваться ими: просто пропустите второй и третий шаг, и на четвертом шаге введите имя своего доверенного сертификата (можно также воспользоваться моим скриптом CA).

Итак, после всех этих действий перемещаем сертификаты и ключ сервера LDAP (ldap.key) куда-нибудь в /etc/ssl/ldap (принято хранить все ключи и сертификаты, созданные openssl в /etc/ssl) и делаем их доступными для чтения только владельцу:

```
  #chmod 0400 /etc/ssl/ldap/*
  и меняем владельца на ldap
  #chown ldap:ldap /etc/ssl/ldap/*
```

Настройка сервера предельно проста: необходимо просто прописать пути к сертификатам и ключам в файле slapd.conf следующим образом:

```
  # Сертификат сервера
  TLSCertificateFile /etc/ssl/ldap/
ldap.cert
  # Ключ сервера
  TLSCertificateKeyFile /etc/ssl/
ldap/ldap.key
  # Доверенный сертификат
  TLSCACertificateFile /etc/ssl/ldap/
ca.cert
```

Сервер можно конфигурировать и альтернативным методом – через туннель stunnel или ssh. Обычно применяют stunnel, но для начала опять же генерируют сертификат сервера (при этом сам LDAP не должен быть настроен для поддержки SSL, т.к. stunnel будет заменять встроенные механизмы SSL ldapa). После генерации сертификата делаем следующее:

```
  #stunnel  -r ldap  -d 636 -p /etc/
ssl/stunnel/stunnel.pem
```

Записываем эту команду в один из init-файлов и добавляем правило iptables для отбрасывания сетевого трафика с порта 389 с целью повышения безопасности (это запрещает доступ по сети к серверу ldap без механизма ssl). Для генерации сертификата в данном случае можно применить скрипт для генерации stunnel сертификатов, который обычно поставляется вместе с stunnel. Можно также применить следующее:

```
  $ openssl req -new -x509 -days 365
-nodes -config stunnel.cnf \
  -out stunnel.pem -keyout stunnel.pem
  $ openssl gendh 512 >> stunnel.pem
```

для генерации self-signed сертификата и параметров для ключей Дифлемана-Хельмана. На самом деле через туннель у меня все работало просто на ура, а встроенные механизмы работали несколько странно, хотя я перечитал все маны и руководства, но это не помогло...

Теперь перезапускаем slapd и настраиваем клиентов. Для настройки клиентов надо прописать в файле ldap.conf следующие строки:

```
  # Аутентификация через TLS
  ssl start_tls
  # По умолчанию клиент пытается
  # использовать обычное соединение,
  # закомментировав следующую строчку,
  # мы ставим жирный крест на его
  # безобразиях
  #ssl off
```

Интеграция

Теперь наш LDAP-сервер работает замечательно (или не работает совсем, что зависит от /dev/hands) и его уже можно использовать в сети, но многие хотели бы заменить старые системы аутентификации на LDAP. Самое время, чтобы рассказать о LDAP и NIS. NIS (службы сетевой информации) широко используются для единого управления паролями в сети. NIS – это более старый и менее защищённый протокол аутентификации. Возможности LDAP значительно шире, и поэтому имеет смысл перенести NIS в LDAP, чтобы использовать возможности последнего и устоявшиеся настройки первого (не вновь же всё ручками писать), благо в пакете migrationtools есть скрипты для миграции NIS и NISPLUS:

```
  migrate_all_nisplus_on[off]line.sh
  migrate_all_nis_on[off]line.sh
```

Я уже рассказал о методах интеграции существущей системы и LDAP через PAM (встраиваемые модули аутентификации), а сейчас настало время рассказать ещё об одном механизме аутентификаци: NSS (дословно выбор службы имён). NSS выбирает, какой тип аутентификации будет использоваться при запросе клиентом определённого файла или механизма (с nss интегрировано множество программ, например, SAMBA). Для настройки NSS используется файл /etc/nsswitch.conf. Его синтаксис предельно прост: имя сервиса (shadow passwd hosts) и далее список параметров (в порядке убывания приоритета), определяющих возможные сетевые службы. Вот, например, файл nssswitch.conf с моей домашней машины:

```
# Legal entries are:
#
#   nisplus or nis+
# Использовать 3-ю версию NIS(NIS+)
#   nis or yp
# Использовать 2-ю версию NIS(YP)
#   dns
# Использовать DNS
#   files
# Использовать локальные файлы
#   ldap
# Использовать LDAP-базу
#   [NOTFOUND=return]
# Если не найдена информация,
# остановить поиск
#

passwd:     files nisplus nis
shadow:     files nisplus nis
group:      files nisplus nis

hosts:      files nisplus nis dns

networks:   files
protocols:  files
services:   files

netgroup:   nisplus
```

Чтобы использовать ldap, просто добавьте в нужные места данный метод. Если вы хотите использовать только ldap, то поставьте после слова «ldap» строчку [NOTFOUND=return], чтобы поиск прекращался при отсутствии элемента в базе LDAP. Например:

```
# Ищем вначале в локальных файлах,
# а потом в LDAP
passwd:     files ldap
shadow:     files ldap
group:      files ldap
hosts:      files dns ldap
```

Учтите ещё одну особенность: я бы не рекомендовал использовать ldap для оперделения хостов, т.к. он будет пытаться найти имя удалённого клиента в /etc/hosts, а потом начнёт перерывать какие-то данные, что вызывает нехилую задержку ~30 секунд (если имени нет в hosts). Так что лучше использовать старый добрый DNS (тем более, записи DNS можно хранить в LDAP, но об этом немного позднее). К тому же, когда я поставил у себя проверку хостов первоначально через LDAP, то последний повис наглухо, т.к. вызывает рекурсивно gethostbyname() для определения своего адреса.

Хорошо, nss мы настроили, теперь примемся за SAMBA. Честно говоря, SAMBA, на мой взгляд, немного кривая софтина (а как же: работает с такими замечательными ОС), но с версии 2.2.6 вопросы, касающиеся LDAP, вроде ушли из мейл-листа, что наводит на радужные мечтания (кстати, SAMBA 2.2.6 требует SP3 у Win2k клиентов). Для компиляции сервера с LDAP наберите ./configure –with-ldapsam. SAMBA имеет возможность указания сервера LDAP и базового dn поиска прямо в smb.conf. Для этого существует несколько опций:

- ldap server – основной параметр, определяет адрес LDAP-сервера;
- ldap ssl – параметр, указывающий надо ли использовать безопасные методы аутентификации (нужен всегда!). Имеет значение on для работы через SSL (порт 636), start tls для обычной аутентификации через tls (389 порт) и no для посылки данных в открытом виде (также порт 389);
- ldap admin dn – параметр, определяющий запись, имеющую права доступа к паролям SAMBA;
- ldap suffix – основной суффикс базы данных LDAP;
- ldap filter – фильтр, по которому SAMBA ищет свои объекты, например:

```
ldap filter =
    "(&(uid=%u)(objectclass =
sambaAccount))":
```

имя пользователя совпадает с логином виндов, а тип объекта – профиль SAMBA;

- ldap port – если ваш сервер LDAP висит на другом порту, чем это назначено по умолчанию (см. параметр ldap ssl), то здесь это можно указать.

Приведу простой пример настройки SAMBA с LDAP:

```
[global]
  # Юзеры должны залогиниться на
  # сервер
  security = user
  # Естественно, шифруем пароли
  encrypt passwords = yes

  netbios name = Linux
  workgroup = NET

  # Параметры LDAP

  # Определяем администрский dn.
  # Пароль к нему должен добавляться
  # непосредственно smbpasswd -w.
  # После смены dn администратора пароль
  # тоже надо сменить!
  ldap admin dn = "cn=ntadmin,
ou=people,dc=test,dc=ru"

  # Определяем сервер LDAP
  ldap server = ldap.test.ru

  # Заходим на LDAP-сервер через TLS
  ldap ssl = start tls

  # Определяем порт(хотя в данном
  # случае это необязательно)
  ldap port = 636

  # Определяем суффикс базы данных
  # ldap
  ldap suffix = "ou=people,dc=test,
dc=ru"
```

Кроме этого, естественно, надо завести администратора SAMBA в LDAP и обеспечить ему соответствующие права доступа в slapd.conf (можно для этой цели использовать рута – это тоже сработает):

```
access to attrs=lmPassword,ntPassword
    by dn="cn=ntadmin, ou=people,
dc=test, dc=ru" write
    by * none
```

После настройки администраторской записи в ldap SAMBA может корректно работать с деревом директорий. При правильной компиляции (с флагом –with-ldapsam) smbpasswd будет добавлять пользователей и машины в каталог LDAP, при этом smbpasswd будет использовать passwd и groups из nsswitch (при правильной настройке nss можно также брать всю инфу из LDAP, т.к. SAMBA обращается к passwd через nss), добавляем пользователей:

```
# smbpasswd -a <username>
```

добавляем машину (для SAMBA-контроллера домена):

```
# smbpasswd  -m -a <mach_name>$
```

После этого проверяем результат:

```
  # ldapsearch -ZZ -H localhost -b
"o=smb, dc=test, dc=ru" "uid=pc1$"
  --
  dn: uid=icb$, o=smb, dc=test, dc=ru
  objectClass: sambaAccount
  uid: pc1$
  pwdLastSet: 0
  logonTime: 0
  logoffTime: 2147483647
  kickoffTime: 2147483647
  pwdCanChange: 0
  pwdMustChange: 2147483647
  displayName: MustdiePC
  cn: PC1
  rid: 2054
  primaryGroupID: 1201
  lmPassword:
56D989A3C45BBAD3462E8109C329E116
  ntPassword:
 56D989A3C45BBAD3462E8109C129E116
  acctFlags: [W          ]
  --
```

Далее вы можете изменять информацию профиля пользователя. Например, весьма полезны атрибуты scriptPath (скрипт, выполняющийся при логине клиента), homeDrive (диск домашней директории), profilePath (путь к профилю), pwdMustChange (необходимость смены пароля пользователем). Для изменения базы данных LDAP на лету существует команда ldapmodify, которая, как и команда ldapadd, работает с файлами ldiff. Я не буду подробно описывать формат этого файла, а просто приведу пример изменения нужных атрибутов:

```
  dn: uid=001, o=smb, dc=test, dc=ru
  changetype: modify
  replace: profilePath
  profilePath: \\%N\profiles\user1
  -
  replace: scripthPath
  scripthPath: 001.bat
  -
  replace: homeDrive
  homeDrive: Z:
  -
  replace: pwdCanChange
  pwdCanChange: 0
  -
  replace: pwdMustChange
  pwdMustChange: 0
  -
  replace: primaryGroupID
  primaryGroupID: 513
  -
```

Для добавления информации в базу воспользуйтесь командой:

```
  $ ldapmodify -H localhost -D
"<admin DN>" -W -ZZ -l modifi-
cation.ldif
```

(Примечание: в руководствах говорится, что надо использовать ключ -f, а не -l, но у меня по-другому не работало, хотя, наверное, я что-то делаю не так).

Любителям готовых примеров посоветую сходить на http://www.unav.es/cti/ldap-smb/ldap-smb-2_2-howto.html – отличный пример настройки контроллера домена с LDAP (не буду же я здесь приводить все эти конфиги, а мои собственные исторической ценности не имеют).

Напоследок скажу ещё об одном приложении, работающем непосредственно с LDAP, – это squid. Для него существует модуль, позволяющий выполнить http-аутентификацию, используя базу LDAP. В исходных текстах сквида можно найти модуль, squid_ldap_auth, представляющий собой внешнюю программу, его исходные тексты находятся в каталоге auth_modules/LDAP. Скомпилировав её обычным образом (./configure -> make -> make install), получаем обычный исполняемый файл squid_ldap_auth. После установки модуля добавляем такие строчки в squid.conf:

```
  authenticate_program  /usr/squid/
libexec/squid/squid_ldap_auth  -b
  dc=test,dc=ru  localhost
```

Вначале указываем путь к исполняемому файлу, затем основной суффикс базы LDAP и LDAP-сервер (localhost у меня). Далее в squid.conf добавляем права доступа(ACL):

```
  # Заставляем всех аутентифициро-
  # ваться для доступа к прокси
  acl password proxy_auth REQUIRED
  # Возможен доступ http всем, кто
  # прошёл проверку
  http_access allow password
  # Остальных шлем погулять
  http_access deny all
```

Теперь о возможности работы с LDAP апача. Апач имеет модуль mod_auth_ldap, который позволяет производить http-аутентификацию через LDAP-сервер. Здесь я опять же ограничусь только примером типичной настройки аутентификации через ldap:

```
  httpd.conf
  # Загружаем модуль аутентификации.
  # Обычно находится в extramodules
  LoadModule auth_ldap_module
extramodules/auth_ldap.so
  AddModule auth_ldap.c

  # Определения аутентификации через
  # ldap для конкретного каталога
  <Directory /var/www/secure>
  # Это основной контекст для поиска
  # соответствия пользователя и пароля
  # в базе LDAP, если данная опция не
  # указана, но используется анонимный
  # доступ (что обычно запрещается в
  # целях безопасности).
  AuthLDAPBindDN
cn=proxy,dc=test,dc=ru
  # Пароль для основного контекста
  AuthLDAPBindPassword secret
  # Включение механизма TLS для доступа
  # к серверу ldap (по умолчанию off)
  AuthLDAPStartTLS on
  # Основной параметр, который
  # определяет сервер ldap, dn поиска,
  # атрибуты и фильтр, по которому
  # выполняется аутентификация:
  # ldaps://host:port/
basedn?attribute?
  # scope?filter, где basedn -
базовый
  # dn для поиска (ищется только в дан-
  # ной ветви дерева и её потомках)
  # attribute - список атрибутов,
  # разделяемых запятой, по которым
  # производится поиск (по умолчанию
  # используется атрибут uid)
  # scope - флаг, определяющий тип
  # возвращаемых значений one - ищется
  # первое выражение, sub - ищутся все
  # выражения, соответствующие
  # фильтру (принято по умолчанию).
  # filter - строка, определяющая
  # фильтр  поиска элементов ldap,
  # заключается в скобки, по умолчанию
  # равна (objectclass=*). Фильтр может
  # содержать логические выражения |
  # и & которые должны стоять не между
  # двумя скобками, а ПЕРЕД ними.
  # Приведу несколько примеров фильт-
  # ров: (|(cn=admin)(uid=root))  -
  # или cn admin или UID root
  # (cn=admin) - только cn = admin
  # В данном примере допустимым являют-
  # ся только объекты, принадлежащие
basedn
  # O=myorg, OU=sysopka
  AuthLDAPUrl  ldaps://test.ru/
O=myorg, OU=sysopka
  # Требуем только пользователей,
  # прошедших проверку на ldap-сервере
  require valid-user
  </Directory>
```

Существует ещё несколько опций модуля, но они используются реже, и о них можно почитать в документации. Proftpd имеет схожий модуль auth-ldap, но здесь описывать его я не буду, т.к. настройка модуля ftp-сервера очень похожа на конфигурацию апача.

Настройка почтовых серверов также не представляет особой проблемы. Я, например, без проблем настроил postfix и курьер, последний настраивать особенно легко, т.к. он работает сразу же после того, как указывается LDAP-сервер, основной dn, запись администратора и её пароль в открытом виде для доступа к полям паролей пользователей (опять же можно использовать proxy). Конфигурация postfix несколько сложнее:

```
  main.cf
  virtual_maps = ldap:ldapvirtual

  # Сервер ldap и порт
  ldapvirtual_server_host = test.ru
  ldapvirtual_server_port = 389
  # Основной dn для поиска по базе
```

```
  ldapvirtual_search_base = ou=mail,
o=YourOrg, c=nl
  # Домен для поиска
  ldapvirtual_domain = test.ru
  # Атрибут, который должен читать
  # postfix при работе с ldap (по
  # e-mail адресу определяется путь).
  ldapvirtual_result_attribute =
maildrop
  # Считываем одну запись из выборки
  ldapvirtual_scope = one
  # Основной контекст и пароль
  ldapvirtual_bind_dn = cn=proxy,
dc=test, dc=ru
  ldapvirtual_bind_pw = secret
```

Реплики

Ну вот, с интеграцией LDAP разобрались... Пора рассказать о дополнительных возможностях данной системы. Итак, реплики. Реплики – это одна из мощных возможностей системы директорий LDAP. Реплики – это раскопирование базы данных LDAP по нескольким серверам, т.е. фактически само дерево «размазывается» по серверам. Реплика существует между несколькими серверами, один из которых является primary (это похоже на службу DNS, только немного для других целей). В таком случае первичный сервер обрабатывает запросы на запись и на чтение, а вторичные – только на чтение. При модификации данных на главном сервере он заходит на все вторичные и синхронизирует их базы со своей. Для создания реплик необходимо три вещи:

- настройка файлов slapd.conf главного и вторичных серверов;
- первоначальный перенос основной базы с основного сервера на вторичные;
- запуск демона slurpd, который и осуществляет синхронизацию серверов.

После осуществления всего этого можно разбивать пользователей на группы и указывать для их клиентов определённый LDAP-сервер и создать основной сервер, который будет все вторичные синхронизировать (звездообразная схема). При этом любые изменения в базе любого из LDAP-серверов будут раскопированы на все сервера, включая основной. Вот примерная схема действия реплики при получении запроса от клиента.

- Вторичный LDAP-сервер получает запрос на изменение и говорит клиенту, что надо изменить данные на основном сервере (то есть перенаправляет клиента к главному).
- Клиент посылает запрос главному серверу, который он обрабатывает. При этом сервер знает, что клиент был к нему перенаправлен (это записывается в лог реплики).
- slurpd замечает (после прошествия некоторого времени), что данные были обновлены, и осуществляет синхронизацию.
- Вторичные сервера получают запрос от slurpd главного сервера и, обновив данные у себя, посылают ответ slurpd, этот ответ также может быть записан в лог реплики.

Ну всё, хватит теории, начинаем настройку. Для начала поправим slapd.conf:

```
  # Это основной сервер!
  # Добавим несколько реплик. Каждая
  # директива соответствует одному
  # вторичному ldap-серверу. Указываем
  # адрес вторичного сервера(host),
  # dn для репликатора, метод
  # аутентификации и пароль репликации
  # (пароль для данного dn). В качестве
  # репликатора можно использовать ру-
  # та. Честно говоря, для реплик я бы
  # создал туннель через ssh или stun-
  # nel и присоединялся бы не к удалён-
  # ному хосту к порту 389, а к локаль-
  # ному на назначенный порт (для раз-
  # ных рабов можно сделать несколько
  # туннелей)
  replica host=slave1.test.ru:389
binddn="cn=replicator, dc=test, dc=ru"
  bindmethod=simple
credentials=replicator_password
  # Следующий раб
  replica host=slave2.test.ru:389
binddn="cn=replicator, dc=test, dc=ru"
  bindmethod=simple
credentials=replicator_password
  # Записываем данные реплик в log-
  # файл, это делать необязательно
  # (даже нежелательно, т.к. понижает
  # производительность). Эта опция еди-
  # на для всех вторичных серверов
  replogfile  /var/log/ldap/
replica.log
```

После чего на вторичных серверах вносим ещё пару строк в slapd.conf:

```
  # Это вторичный сервер!
  # Нельзя использовать директивы
  # replica replogfile. Необходимо
  # создать некий updatedn, совпадающий
  # с binddn основного сервера,
  # по которому главный будет
  # модифицировать данные.
  updatedn="cn=replicator,dc=test,dc=ru"
  # Включаем адрес основного сервера
  # ldap, куда будет перенаправляться
  # клиент, пытающийся модифицировать
  # данные
  updateref  master.test.ru:389
  # Надо дать необходимые привилегии
  # репликатору, т.к. ему необходим
  # доступ на запись данных
  access to *
  by dn="cn=root, dc=test, dc=ru" write
  by dn="cn=replicator, dc=test, dc=ru"
write
  by self write
```

После настройки всех серверов необходимо перекопировать содержимое базы данных LDAP основного сервера на все вторичные сервера, для этого необходимо перенести содержимое DatabaseDir физически(!). После всего этого желательно проиндексировать базу, иначе поиск по ней замедляется:

```
  slapindex -f /etc/ldap/slapd.conf
-b "dc=test, dc=ru"
```

Затем на главном сервере запускаем slurpd:

```
  # slurpd
```

Использование stunnel в данном случае также приемлемо – на главном сервере устанавливаются клиенты туннеля, которые направлены ко вторичным LDAP-серверам, но на сервере эти туннели будут висеть на разных портах, и LDAP-сервер будет соединяться с локальными туннелями, висящими на портах, которые соответствуют рабам ldap:

```
  stunnel -c -d 9636 -r
slave1.test.ru:636
  stunnel -c -d 10636 -r
slave2.test.ru:636
```

И это всё! Если вторичный сервер не сможет пробиться к основному (например, электрик дядя Ваня дерзко перерубил оптоволокно), то клиент, подключенный к вторичному серверу, сможет читать информацию, но изменять её не сможет. А если в результате боевых действий рухнул винт, то с любого вторичного сервера можно будет восстановить первозданную базу LDAP (какой она была до краха основного сервера). По такой распределённой схеме можно создавать сколь угодно большую сеть, так как основными являются запросы на чтение, и нагружаться будут вторичные сервера.

Создание объектов LDAP Файлы схем

LDAP, как уже было сказано, является расширяемой объектно-ориентированной системой, и вы можете добавлять новые классы с новыми атрибутами в базу LDAP. Для этой цели су-

ществуют файлы схем. В файле схемы описываются классы и их атрибуты, чтобы подключить схему к LDAP, необходимо прописать в файле slapd.conf полный путь к нужной схеме. По умолчанию к LDAP подключены несколько основных схем, существуют также сторонние схемы, например для SAMBA или некоторых почтовых серверов. Чтобы эти схемы использовать, в slapd.conf прописывают нечто подобное:

```
  # Включаем схемы по умолчанию
  # командой include
  include /usr/share/openldap/schema/
core.schema
  include /usr/share/openldap/schema/
cosine.schema
  include /usr/share/openldap/schema/
inetorgperson.schema
  # Включаем схему самбы
  include /usr/share/openldap/schema/
samba.schema
  # Включаем свою схему
  include /usr/share/openldap/schema/
my_cool.schema
```

Теперь разберёмся, как эти схемы создавать. Если вы откроете какую-нибудь схему, то сначала всё покажется довольно жутким. Хочу сразу же вас обрадовать: это действительно жутко! Чтобы хоть чего-то объяснить расскажу об OID. OID (уникальный идентификатор объекта) используется LDAP, чтобы не было наложений классов и их атрибутов. OID состоит из числовых значений разделённых точкой и составляющих в целом иерархию OID:

```
1.1
   OID  данной организации(выбира-
   ется любой)
1.1.2
   Сами идентификаторы LDAP
1.1.2.1
   Атрибуты классов
1.1.2.1.1
   Конкретный атрибут(меняем послед-
   нюю цифру для каждого атрибута)
1.1.2.2
  Сами классы
1.1.2.2.1
   Конкретный класс
```

Для получения OID можно зайти на узел www.iana.org (начинается с 1.3.6.1.4). Вообще, как я понял, составление иерархии OID – дело довольно проблематичное. Проще всего взять какой-либо OID для атрибутов (обычной базой OID является 1.3.6.1.4) и для классов, а затем просто добавить ещё несколько цифр. А вообще, лучше просто посмотреть, как сделано в готовых схемах (я думаю, что изменить последнюю пару чисел готового OID не составит труда, а описывать это мне представляется совершенно ненужным). После сочинения OID перейдём к добавлению новых атрибутов. Для этой цели используется директива attributetype. Вот как выглядят описания атрибутов в схеме SAMBA (приведён маленький фрагмент):

```
attributetype ( 1.2.840.113556.1.4.8
NAME 'userAccountControl'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
SINGLE-VALUE )

attributetype (1.2.840.113556.1.4.166
NAME 'groupMembershipSAM'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
SINGLE-VALUE )

attributetype (1.2.840.113556.1.4.213
NAME 'defaultClassStore'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12)
```

Как видно, в скобках идёт вначале OID атрибута, затем NAME и имя атрибута в кавычках, а затем тип атрибута. Вот тут немного остановлюсь. Видно, что тип атрибута – это также OID (ужас-то какой, но это позволяет добавлять даже свои типы). Стандартно предопределено несколько типов:

```
логический
    1.3.6.1.4.1.1466.115.121.1.7
имя LDAP(DN)
    1.3.6.1.4.1.1466.115.121.1.12
строка формата UTF-8
    1.3.6.1.4.1.1466.115.121.1.15
строка ASCII
    1.3.6.1.4.1.1466.115.121.1.26
целое
    1.3.6.1.4.1.1466.115.121.1.27
DN и идентификатор пользователя(UID)
    1.3.6.1.4.1.1466.115.121.1.34
строка из чисел
    1.3.6.1.4.1.1466.115.121.1.36
OID
    1.3.6.1.4.1.1466.115.121.1.38
```

Кроме типа атрибута можно указывать дополнительные его параметры:

- DESC – строка (обычный текст), которая описывает данный атрибут.
- SUP – строка (имя) или OID родительского атрибута (происходит наследование синтаксиса родительского атрибута).
- EQUALITY – строка (наименование) или OID метода поиска, например, caseIgnoreMatch для поиска данного атрибута без учёта регистра.
- SINGLE-VALUE – просто флаг (после него должен обязательно стоять пробел!), который указывает, что данный атрибут может принимать только единственное значение (по умолчанию ему можно присваивать несколько значений).
- NO-USER-MODIFICATION – флаг, запрещающий модификацию данного атрибута пользователем (по умолчанию модификация разрешена).

При составлении схемы учтите, что после каждой лексемы (включая скобки и флаги) должен идти один пробел. Приведу пример составления атрибута картинки как пути к jpg-файлу:

```
attributetype ( 1.1.2.1.2 NAME 'pic'
DESC 'my jpeg picture'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
SINGLE-VALUE )
```

И, например, картинки на сайте, представленной в виде URL:

```
attributetype ( 1.1.2.1.3 NAME 'URLpic'
DESC 'URL to picture'
SUP pic )
```

Создание нового класса несколько проще, приведу пример класса cronEntry из cron.schema:

```
objectclass (1.3.6.1.4.1.7006.1.3.2.1
    NAME 'cronEntry'
    SUP top
    STRUCTURAL
    MUST ( cn $ cronCommand $ uid )
    MAY ( cronHost $
            cronMinute $
            cronHour $
            cronDay $
            cronMonth $
            cronDayOfWeek $
            owner $
            description ) )
```

Вначале идёт OID, имя (не забудьте кавычки), родительский класс (наследуются атрибуты), тип класса (для знакомых с объектно-ориентированной моделью объясняю: ABSTRACT – абстрактный (нельзя делать объекты данного класса, можно только наследовать) и STRUCTURAL – обычный (по умолчанию)). Далее идут поля атрибутов для класса: MUST – обязательные атрибуты, MAY – необязательные. Список атрибутов заключается в скобки и разделяется по странной прихоти не запятыми, а знаками доллара (очевидно, это намёк разработчиков). Думаю, тут можно ещё раз напомнить о необходимости пробелов после каждой лексемы (и долла-

ров тоже). Вот видите, создание объектов – не такая уж и тяжёлая вещь. Примеров приводить не буду, т.к. класс cronEntry является идеальным объектом для рассмотрения. Честно говоря, я всё-таки рекомендовал бы всем, кто собирается создавать новую схему, посмотреть существующие, т.к. во-первых, необходимо убедиться в уникальности OID, а во-вторых, меньше будет ошибок в своей схеме.

В качестве заключения

Я думаю, что если у вас в распоряжении находится крупная сеть, то применение LDAP является идеальным решением централизованного управления аутентификацией на всех уровнях. Использование реплик делает LDAP надёжным, а хорошие методы шифрования обеспечивают приличную степень безопасности. Кроме этого, LDAP является довольно быстрым механизмом поиска и, что очень важно, хранит всё в древовидной базе данных. Для небольших сетей, я думаю, лучше использовать текстовые файлы: просто и сердито, а главное быстрее чем LDAP. Но в крупных сетях это приведёт к тихому ужасу, связанному с конфликтами разных пользователей, машин, серверов. Я сам был свидетелем сетки из ~1000 машин, соединённых банальной сеткой мелкомягких (без доменов!), где NovellNetware использовалась лишь как файл-сервер. LDAP и SAMBA в этом случае снимают большинство этих проблем. Итак, на этот раз всё. В следующий раз я расскажу о популярных LDAP-клиентах и серверах, о создании собственных клиентов.

Список полезных ссылок:

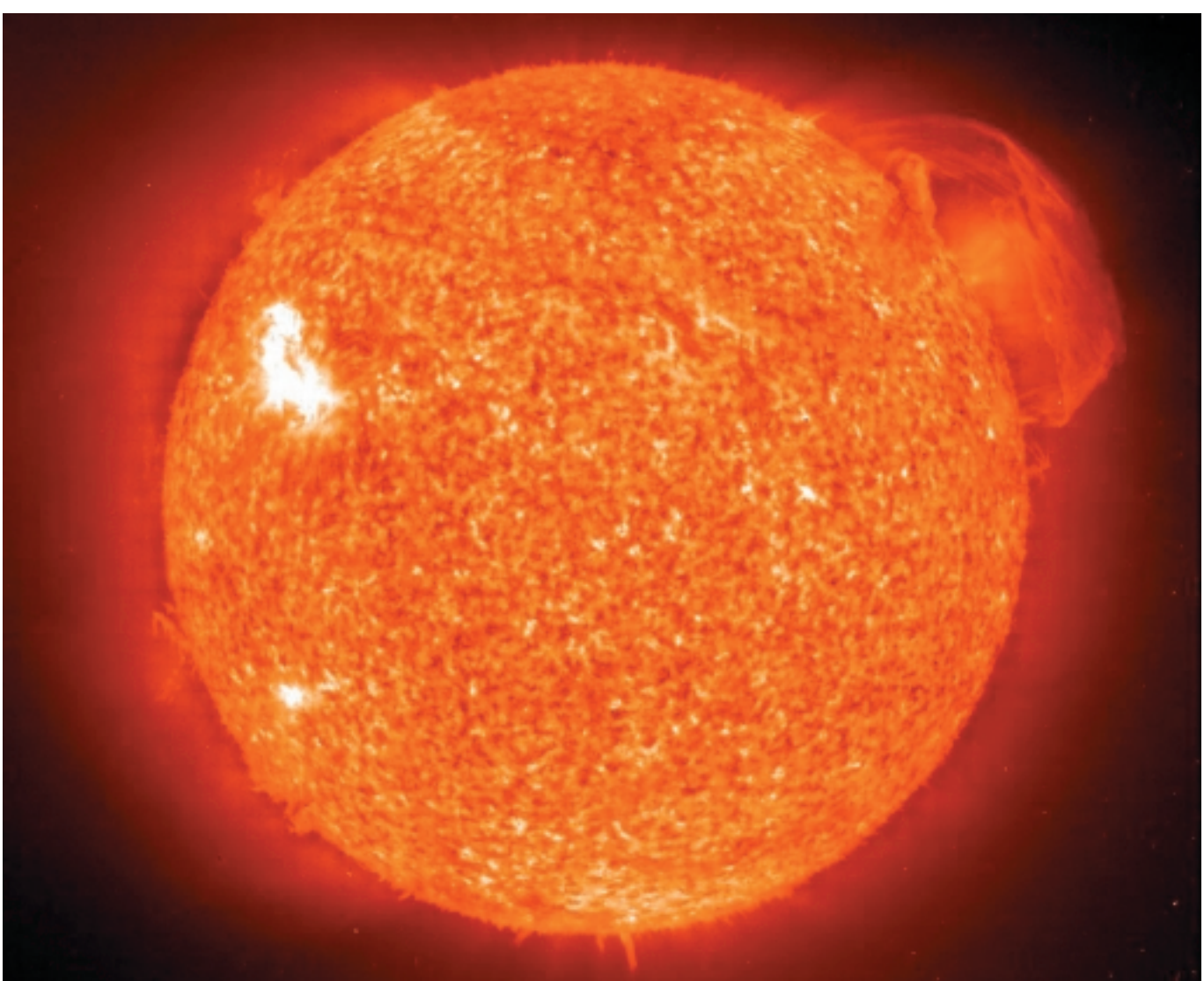
http://www.tldp.org – LDAP HOWTO и LDAP-implementation HOWTO;

http://www.openldap.org – ldap guide (aka HOWTO, обычно самое новое руководство);

http://www.mandrakesecure.net/en/docs/ldap-auth.php – очень приличная статья Vincent Danena, которая описывает принципы настройки LDAP;

http://www.opennet.ru – статья по работе с LDAP на русском языке, но она довольно старая, хотя там немало полезного материала.

К сожалению, в данной статье я упустил из виду некоторые вещи по работе с LDAP, например, работу с SASL, Kerberos, описание програмных компонентов ldap. Постараюсь закрыть эти недостатки в следующих статьях. LDAP – довольно сложная система, поэтому охватить всё не удалось. Надеюсь, что кому-то мои труды пошли на пользу.



Переполнение буфера в Winamp

Winamp поддерживает проигрывание b4s-файлов, которые составляются посредством xml. Синтаксис таков:

```
<?xml version="1.0" encoding='UTF-8' standalone="yes"?>
<WinampXML>
<!-- Generated by: Nullsoft Winamp3 version 3.0 -->
<playlist num_entries="[кол-во записей]"
label="[название_list'a]">

<entry Playstring="file:[путь_к_звуковому_файлу]">
<Name>[названи_песни]</Name>
<Length>[величина_в_байтах]</Lengt>
</entry>

</playlist>
</WinampXML>
```

Переполнение буфера происходит при обработке чрезмерно длинного названия листа (4-ая строчка). Причем есть некоторые нюансы:

При получении 16 398 байт переполнение происходит только при выходе из WinAmp'a. 4-мя байтами затирается eax и retaddr (причем ret находится в регистре eax). Если же название листа еще увеличить на ~100b, переполнение произойдет сразу же при запуске. 12-ю байтами перезапишутся ecx, esi и retaddr. Хотя, что касается адресного пространства, данные могут разниться в зависимости от билда WinAmp'a и вообще операционной системы. Так или иначе, составить b4s'ку, протроянивающую систему, не составляет никакого труда.

Уязвимость обнаружена D4rkGr3y из Damage Hacking Group.

Межсайтовый скриптинг в Apache

Уязвимость межсайтового скриптинга обнаружена в типовом сценарии "printenv", поставляемом с Apache веб-сервером. Уязвимость позволяет удаленному атакующему создать злонамеренную ссылку, содержащую произвольный код сценария в параметрах уязвимого скрипта, который будет выполнен в браузере пользователя в контексте уязвимого сайта при клике на такую ссылку. Уязвимость позволяет атакующему красть данные, хранящиеся в куки пользователя. Пример:

```
http://www.example.com/cgi-bin/printenv/<a
href="bad">test</a>
```

Уязвимость обнаружена в Apache Software Foundation Apache 1.3.22-2.0.43.

Множественные уязвимости в KDE

В некоторых случаях KDE неправильно приводит параметры инструкций, переданные в командную оболочку для исполнения.

Эти параметры могут содержать данные типа URL, имен файлов и адресов электронной почты, и эти данные могут быть дистанционно преданы жертве в электронном сообщении, веб-странице или через другие источники.

Тщательно обрабатывая такие данные, нападающий может выполнить произвольные команды на уязвимой системе с привилегиями текущего пользователя.

Уязвимость обнаружена в KDE 2.0-3.0.5.