AACLS module - Documentation

Sébastien BAHLOUL   Alexandre D'ALTON

January 21, 2004

Contents

Introduction

First of all, this is a preleminary version and this documentation has been written by a non english speaker. So don't hesitate to write to us to correct it !

The AACLS acronym stands for "Advanced ACL Server". This a backend of the LDAP OpenLDAP server, in other words, a LDAP content filtering gateway based on an ACL langage used to describe relations between entries. These ACL are stored in the directory and control the access given to entries and attributes.

Of course you can use other ACL systems like classical ACLs or ACI, but you cannot describe relation between entries and give rights depending on them ! You have ways to designed your directory in a such way you don't need this but you are going to loose all the trees power.

Because this server use a very complex way of determining access on information stored in a LDAP Server, you have to be warned about performance issues in using this software : for example, it is completely inappropriate to use an OS or clients which are very time dependant with this backend !!!

Just some words about history :

This software is based on the work done by Jean-Baptiste Nataf. This computer engineer is working for "Université Pierre et Marie Curie - Paris VI" in France. The main idea is that : *The need of a better ACL system is based on the fact that simple systems are too far from the real world !*

The first implementation has been done on an PHP application server. But the need of speed motivate us to write it as an OpenLDAP backend.

All the source code is available through http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/aacls. But at this time no package has been done.

Any help and comments are welcomed at seb@stien.info.

Chapter 1

General architecture

Here follows two simple schema about physical and logical architecture.

And now some explanations :

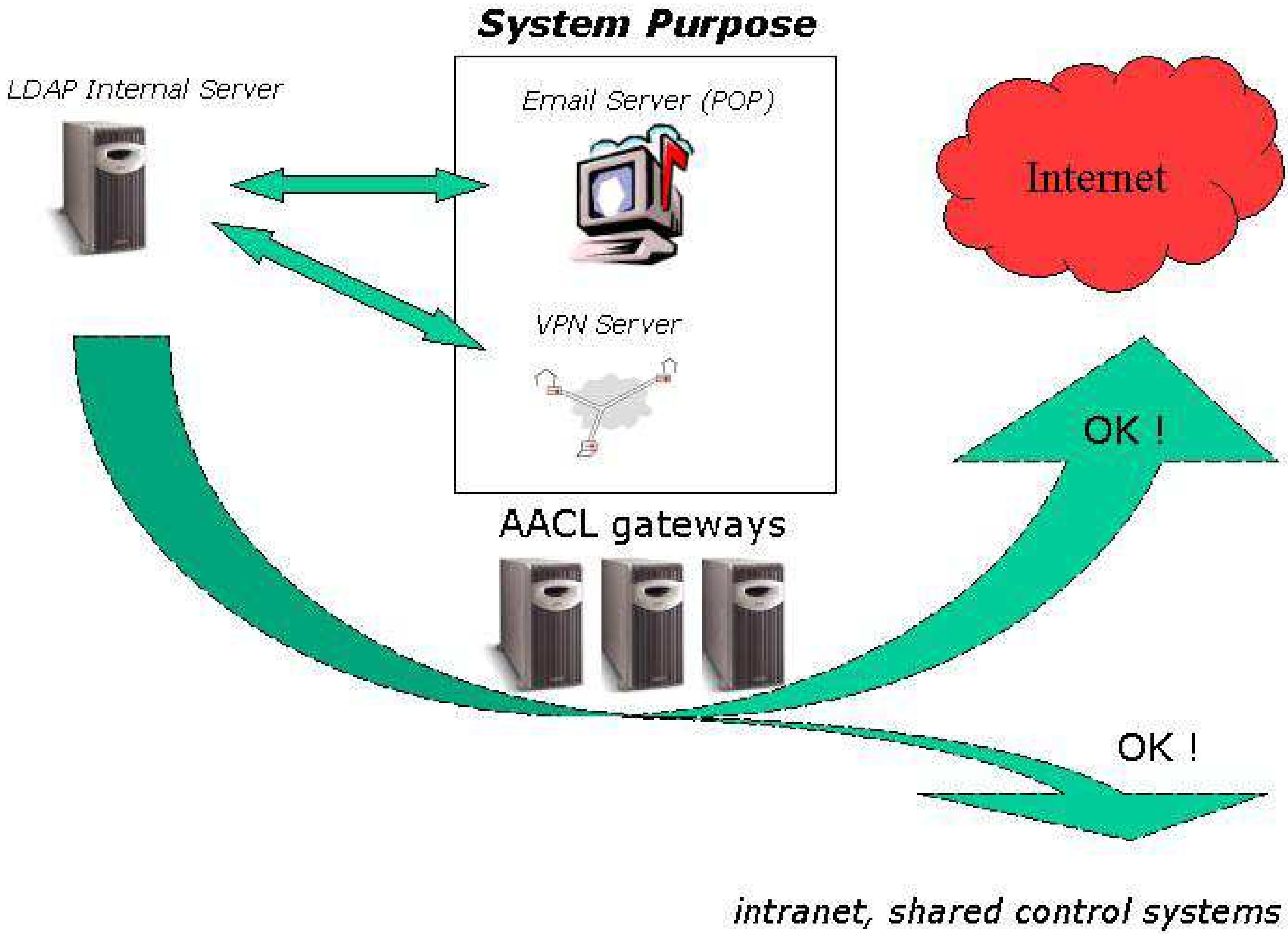

Figure 1.1: Physical architecture

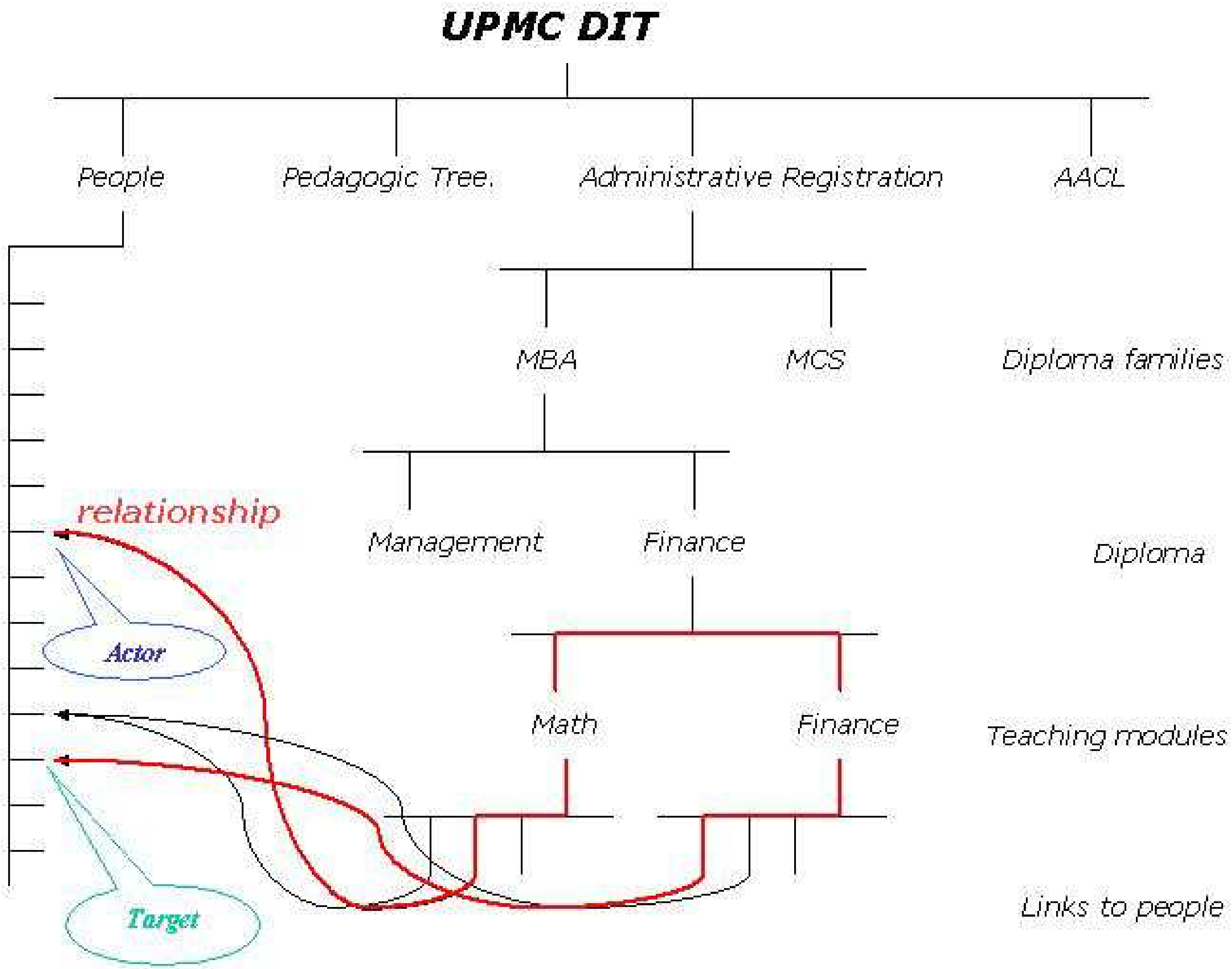

Figure 1.2: Logical architecture

Chapter 2

AACLS schema and syntax

In this part, you will find the concept, the AACLS LDAP schema and the way to use it with some examples.

2.1 AACLS Concept

The main idea of AACLS is relationship between entries. An AACL works with :

- the author which is the DN passed through the auth method of the LDAP server,
- the target which is the DN of the entry on which the operation is tried,
- one or more attributes on which the operation is tried,
- the operation,
- and a relation.

The AACL checks if the author and the target are linked through the relation. Of course the operation needs to be allowed on the specified attribute(s). The operation is allowed through the "rights" attribute.

By default, nothing is possible. Each AACL is tried successively. If one of them authorized the operation, the result is immediately sent back. When the AACLS server has tried each AACL with no authorization, it fails.

Both of the author and the target are described with a base and a filter. Only the base of the target is mandatory.

2.2 AACLS Syntax

Each AACL is described in a single entry which have the aacls objectClass. To define this, we use five ten attributes.

2.2.1 "targetBase"

First of all a AACL work on a "target base". This base reduce the subtree where the AACL is enabled. This is a mandatory multi - value attribute which is a DN.

Take care : if you use different bases, the AACL will be evaluated for each base !

2.2.2 "targetFilter"

When it is set, "targetFilter" is used to search through the destination directory for the entries before trying to apply the AACL.

This is an optional mono-value attribute.

2.2.3 "authorBase"

In the same way, "authorBase" is used to reduce the load because of avoiding evaluating non necessary AACLs. This is a optional multi - value attribute which is a DN.

2.2.4 "authorFilter"

When it is set, "authorFilter" is used to search through the destination directory for the author before trying to apply the AACL.

This is an optional mono-value attribute.

2.2.5 "attribute"

This attribute describes the attributes on which the AACL applies.

This is an mandatory multi - value attribute.

2.2.6 "rights"

This attributes contains the rights of the AACL which are one or more of the following :

- *r* reading
- *w* writing (adding a new value to an existing attribute, modifying an existing attribute, removing an attribute)
- *c* creating (creating a new attribute and a new entry)
- *d* deleting (deleting an entry)

Both of the creating and deleting are not clearly defined at the moment.

This is a mandatory mono-value attribute.

2.2.7 "cn"

This attribute is used to identify the AACL. It has to be a number because the AACLs order is based on it !

This is an mandatory mono-value attribute.

2.2.8 "description"

This attribute describes the AACL.

This is an optional mono-value attribute.

2.2.9 "relation"

To be able to describe relations between entries, we have created a simple langage. You can use functions, get attribute values, use either the DN or the RDN of either the author or the target, and a litteral string. When severeals choices are available, take successively each of them.

List of available functions :

- *and* :
- *or* : binary OR function
- *egal* : compare DN equality
- *search* : LDAP search operation with LDAP_SCOPE_SUBTREE
- *list* : LDAP search operation with LDAP_SCOPE_ONELEVEL
- *sup* : if the second argument is a positive non nul number, remove n elements to the first argument which is attented to be a DN. If the first number is 0, each superior level is return. It stops reaching the suffix of the backend.

All these functions accept two operands. The functions are used in the following way :
$function(a, b)$ or $function.(a, b)$
where function can be replaced be any available function, and a and b are expressions.
List of available variables :

- $authorDN$:
- $authorRDN$:
- $targetDN$:
- $targetRDN$:

All these variables can be used with the following syntax :

$"\$variable"$

and finally you can get the value(s) of an attribute :

$(expression).attribute$

where attribute can be any of the available attributes on the destination LDAP server

So just give a look at the following examples :

1) $list("ou = People, dc = upmc, dc = fr", "(\&(uid = \$authorRDN)(uid = \$targetRDN))")$

Check if the author and the target are the same person.

2) $search.(sup.(search.("ou = AdministrativeRegistration, dc = upmc, dc = fr", "uid = \$targetRDN"), 2), "uid = \$authorRDN")$

Check if the author and the target are in the same subtree "two level up".

This is an mandatory mono-value attribute.

2.3 AACLS Schema

The private OID subtree (13129) is dedicated to "Université Pierre et Marie Curie - Paris VI"

The attributeType :

```
attributeType ( 1.3.6.1.4.1.13129.1.2.9 NAME 'attribute'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

attributeType ( 1.3.6.1.4.1.13129.1.2.10 NAME 'targetBase'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

attributeType ( 1.3.6.1.4.1.13129.1.2.11 NAME 'targetFilter'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

attributeType ( 1.3.6.1.4.1.13129.1.2.12 NAME 'authorBase'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

attributeType ( 1.3.6.1.4.1.13129.1.2.13 NAME 'authorFilter'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

```
attributeType ( 1.3.6.1.4.1.13129.1.2.14 NAME 'relation'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

attributeType ( 1.3.6.1.4.1.13129.1.2.15 NAME 'rights'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{10}
        SINGLE-VALUE )
```

And the objectClass :

```
objectClass ( 1.3.6.1.4.1.13129.1.1.6 NAME 'aacls'
        SUP top
        MUST ( cn $ attribute $ rights $ targetBase )
        MAY ( description $ targetFilter $ authorBase $ authorFilter $ relation ) )
```

2.4 AACLS Examples

2.4.1 A simple example

Let's look at a simple example

```
dn: cn=5,ou=ACL2,dc=upmc,dc=fr
cn: 5
objectClass: aacls
objectClass: top
attribute: cn
attribute: mail
attribute: redList
attribute: photoPublication
attribute: uid
rights: r
relation: search.("ou=Personnes,dc=upmc,dc=fr", "(&(uid=$targetRDN)(uid=$authorRDN))")
targetBase: ou=People, dc=upmc, dc=fr
description: Enable a person to read part of his own information (cn, mail, redList, photoPublication and uid)
```

This complete entry describes an AACL.

2.4.2 A complex example

This example is based on the DIT of the UPMC directory. So just some explanations : students are stored in severeals trees. The self relative information are stored in "ou=People,dc=upmc,dc=fr".

The information which are relative to a registration is stored in "ou=Administrative Registration,dc=upmc,dc=fr". In this tree, there's three levels : family of diplomas, diplomas and modules of teaching. In the general case, a student can see part of the other students' information, i.e. read cn, mail, employeeType attributes of students which are in the same diplomas. In the DIT, this means that they are in the same subtree with a depth of 2.

And now the AACL :

```
# 11, ACL2, upmc, fr
dn: cn=11,ou=ACL2,dc=upmc,dc=fr
objectClass: aacls
objectClass: top
attribute: cn
attribute: mail
attribute: employeeType
rights: r
targetBase: dc=upmc, dc=fr
description: Enable a student to see other relative students
cn: 11
relation: search.(sup.(search.("ou=Administrative Registration, dc=upmc, dc=fr","uid=$targetRDN"),2),"uid=$authorRDN")
```

Chapter 3

AACLS module configuration

To be able to configure the AACLS gateway you need the following information :

- URI : the LDAP url of the original server
- BINDDN and BINDPW : the bind DN and password to use to connect to the original server (at this time only the LDAP_SIMPLE_BIND method is used)
- SUFFIX : the suffix used
- BASE : the base DN where the AACLS gateway can find the ACL inside the LDAP tree

The type of backend to notice is "aacls". Here follows an example :

```
# This is the main ldapd configuration file. See slapd.conf(5) for more
# info on the configuration options.

# Schema and objectClass definitions
include         /etc/ldap/schema/core.schema
include         /etc/ldap/schema/cosine.schema
include         /etc/ldap/schema/nis.schema
include         /etc/ldap/schema/inetorgperson.schema
include         /etc/ldap/schema/aacls.schema

schemacheck     off
pidfile         /var/run/slapd.pid
argsfile        /var/run/slapd.args

database        aacls
suffix          "dc=upmc, dc=fr"
binddn          "cn=admin, dc=upmc, dc=fr"
bindpw          "xxxxxxxx"
base            "ou=ACL2, dc=upmc, dc=fr"
uri             ldap://ldap-orig:389/
```

Chapter 4

Licensing and trademarks

This software is released under GPL v2 license.

This backend is designed specifically for OpenLDAP Softare but
is not a product of the OpenLDAP Project <http://www.openldap.org/>.
OpenLDAP is a registered trademark of the OpenLDAP Foundation.