ISBN 978-89-6211-679-3

MyProxy와 LDAP를 연동한 인증시스템 구축 기술 보고서

문종배 (jbmoon@kisti.re.kr)
남덕윤 (dynam@kisti.re.kr)

차세대연구환경개발실
한국과학기술정보연구원

ISBN 978-89-6211-679-3

1. 개요

본 기술문서는 KISTI 슈퍼컴퓨터 인증에서 사용하는 LDAP 기반 사용자 인증방법과 그리드 환경을 위한 MyProxy를 연동하여, 보다 손쉽고 유연한 인증시스템을 구축하는 방법에 대해 기술하고자 한다.

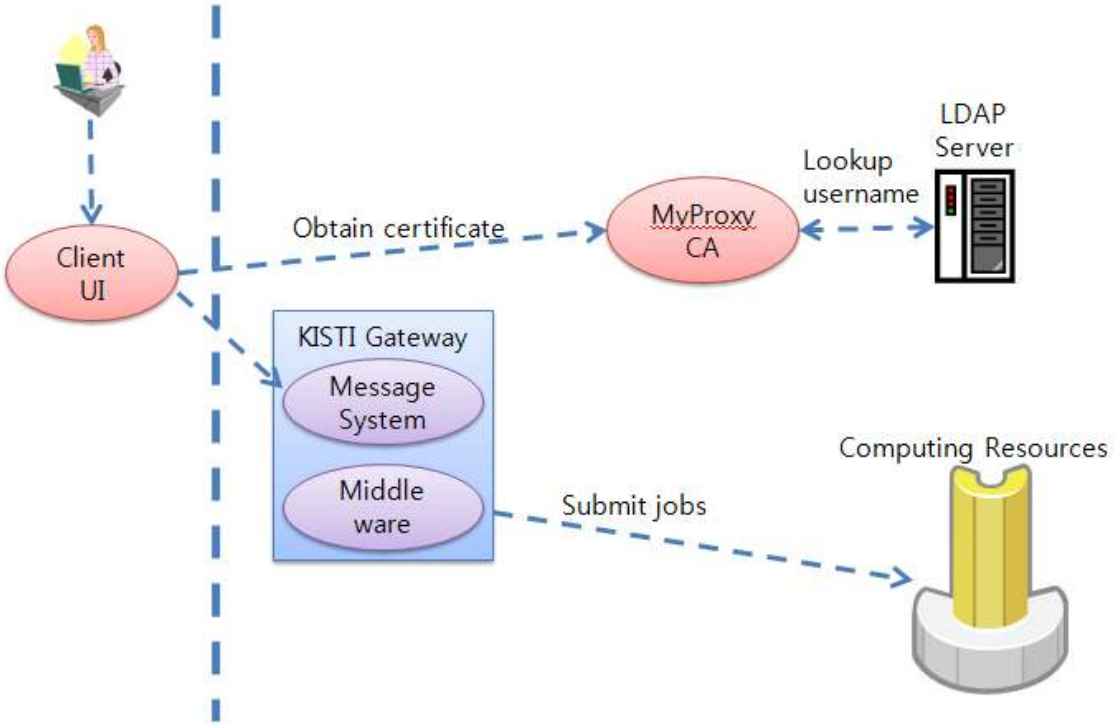


그림 1. MyProxy와 LDAP 연동 기반 작업제출기법 구성도

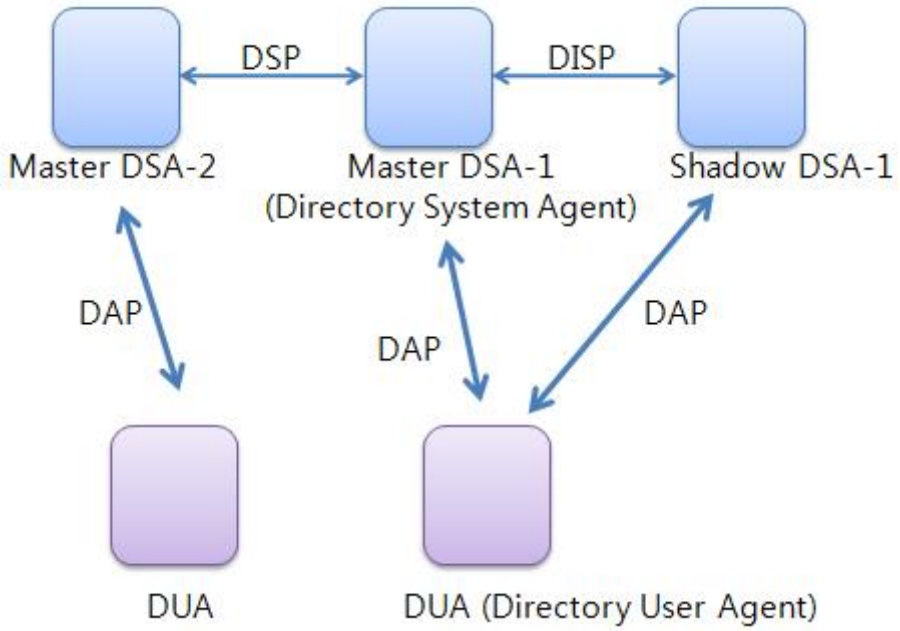
□ MyProxy

MyProxy 는 X.509 Public Key Infrastructure (PKI) 보안 증명서들(인증서들과 개인키)을 관리하는 오픈소스 도구로써, 사용자가 인증서를 언제 어디서든 안전하게 획득할 수 있도록 온라인 인증서 저장소와 온라인 인증서의 허가를 동시에 제공한다. 사용자는 myproxy-logon 명령어를 이용하여 검증된 인증서를 획득할 수 있다. MyProxy는 Certificate Authority (CA) 역할을 수행할 수 있으며, 기존 다른 인증시스템과 연동할 수 있다. 따라서 MyProxy와 패스워드를 연동할 수도 있으며 MyProxy와 LDAP와 연동하여 인증시스템을 구축 할 수도 있다.

ISBN 978-89-6211-679-3

□ LDAP (Lightweight Directory Access Protocol)

Lightweight Directory Access Protocol 이란 '경량의 디렉토리 액세스 프로토콜'이다. 디렉토리란 특별한 형태의 데이터베이스라고 할 수가 있다. 그리고 쓰기 작업보다 읽기 작업이 더 많을 뿐 아니라 어떤 것을 찾는 작업이 많은 곳에 더더욱 적합한 서비스라고 할 수가 있다. 1980년대 말에 특정분야의 디렉토리 서비스의 이용과 개발 요구가 높아감에 따라 CCITT(International Telegraph and Telephone Consultative Committee, 현재는 ITU이다)와 ISO(International Organization for Standardization) 두 단체가 함께X.500이라는 디렉토리 서비스 표준을 만들기 시작하였다. 결국 1990년에 CCITT가 표준을 발표했고 1993년, 1997년 몇 번의 수정작업을 거쳐 현재에 이르렀다. 이 X.500은 최초의 일반적인 목적의 디렉토리 시스템이었고 다양한 쿼리를 사용하는 강력한 검색기능을 제공하였을 뿐만 아니라 서버와 데이터의 분산이용이했고 그리고 무엇보다도 특정 운영체제나 특정 네트워크와 특정 응용프로그램에 구애받지 않고 사용될 수 있는 표준이라는 점이 눈길을 끌 수 있었다.



[그림 2] X.500 디렉토리 서비스 구성

ISBN 978-89-6211-679-3

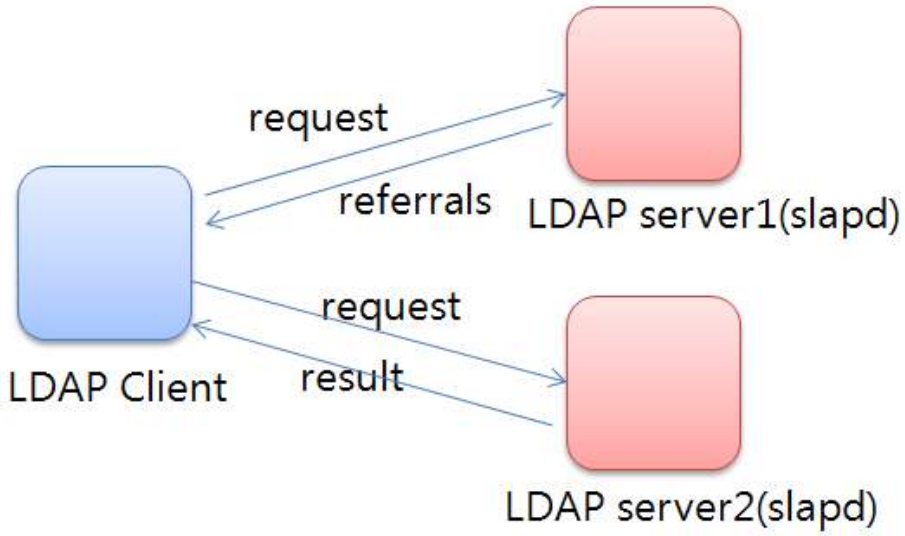


그림 3. LDAP 서비스 구조

하지만, X.500 개발자들은 DAP(X.500의 directory client access protocol)가 너무 방대한데다 복잡하고 구현하기 어렵다는 점 때문에 그 당시의 일반 PC급에서는 적용해서 사용하기가 힘들다는 걸 알았고 이의 해결책을 모색하기 시작했고 그렇게 해서 나온 것이 LDAP이다. LDAP는 DAP의 기능을 거의 다 지원을 했고 복잡했던 부분이나 잘 쓰이지 않았던 부분은 단순화하거나 없애버렸다. 그리고 대부분의 데이터 형식에 있어서 단순한 문자열을 사용함으로써 구현을 단순화하고 퍼포먼스를 늘릴 수가 있었다. 이렇게 LDAP는 처음에 X.500 디렉토리 서비스의 프론트엔드로 사용되었다. 그 후 최초이면서 많이 알려진 미시건대학의 LDAP(U-M LDAP)가 나오게 되었고 현재 많은 상용 또는 오픈소스의 LDAP제품들이 나와 있다.

□ OpenLDAP (Open Source Lightweight Directory Access Protocol)

OpenLDAP는 LDAP의 전신이라고 할수 있는 Umich(미시건대학) LDAP 3.3을 기반으로 새로이 만든 LDAP프로젝트의 산물이다. 오픈소스 정책을 따르면서도 상용 LDAP서버 못지않은 응용프로그램들과 서버툴들을 제공하겠다는 목적아래 인터넷의 많은 개발자들이 자원해서 이 프로젝트를 돕고 있다.

ISBN 978-89-6211-679-3

2. MyProxy와 LDAP을 연동한 인증 시스템 구축

2.1 MyProxy 설치

MyProxy 는 GT(Globus Toolkit) 에 포함되어 배포되고 있다. 따라서 GT4.x 또는 GT5.x 를 다운로드 받아야 한다. 다운로드 위치는 다음과 같다.
http://www.globus.org/toolkit/downloads/

GT의 기능 중 MyProxy만 설치하여 사용한다. MyProxy GPT Package를 다운로드 한다. 설치에 맞는 MyProxy version을 선택하는데 있어서 호환성을 검토하여 선택한다.

데몬으로써 myproxy-server를 실행한다면 진행 전에 수행중인 서버를 정지한 후 다음의 작업을 수행해야 한다. 예를 들어, myproxy init.d script를 설치한다면, myproxy-server를 다음 명령어로 정지시킨다.

```
/etc/rc.d/init.d/myproxy stop
```

다음으로 MyProxy GPT Package를 설정한다. Globus Toolkit를 설치하려는 디렉토리를 GLOBUS_LOCATION 환경 변수에 지정한다. 또한 GPT_LOCATION 환경 변수도 지정한다. GPT_LOCATION와 GLOBUS_LOCATION은 같을 수 있다. GPT 소프트웨어는 GLOBUS_LOCATION 환경 변수에 의해 설정된 위치에 설치되어야만 한다. 또한 다수의 GPT 패키지들을 설치할 예정이라면, 모든 패키지들이 동일한 위치에 설치되어야 한다. MyProxy 설치를 위한 GLOBUS 관련 내용들은 다음 command로 다음의 내용들을 확인해 볼 수 있다.

```
$ $GPT_LOCATION/sbin/gpt-query globus_gssapi_gsi
4 packages were found in /usr/local/globus that matched your query:


packages found that matched your query
globus_gssapi_gsi-gcc32dbg-dev ver: 3.13 cmp id: 3.13.0
globus_gssapi_gsi-gcc32dbg-rtl ver: 3.13 cmp id: 3.13.0
globus_gssapi_gsi-gcc32dbgpthr-dev ver: 3.13 cmp id: 3.13.0
globus_gssapi_gsi-gcc32dbgpthr-rtl ver: 3.13 cmp id: 3.13.0
```

MyProxy 패키지를 제거하고 싶을 때는 다음 command를 수행한다.

```
$GPT_LOCATION/sbin/gpt-uninstall myproxy
```

최종적으로 MyProxy 패키지는 다음 command를 실행하여 설치한다.
Finally, install the MyProxy package with the following command:

```
$GPT_LOCATION/sbin/gpt-build -force -verbose myproxy*.tar.gz <flavor>
```

<flavor> 태그에서는 선택한 Globus flavor를 입력한다.
그리고 다음 command로 시작한다.

```
/etc/rc.d/init.d/myproxy start
```

이제 MyProxy client 툴들을 모두 설치했다. MyProxy client를 이용하기 위해, Globus 환경을 설정해야 한다. MyProxy 설치 위치를 GLOBUS_LOCATION 환경 변수로 설정한다. 그리고 다음 명령어로 환경 설정을 반영한다.

```
source $GLOBUS_LOCATION/etc/globus-user-env.csh
```

MyProxy client를 위한 추가 문서는 MyProxy 사용자 가이드에서 확인할 수 있으며, MyProxy 서버의 설정을 위해서는 Server Installation 가이드를 확인한다. MyProxy와 관련한 버그가 있다면 Bugzilla에 보고하면 Troubleshooting Guide를 통해 안내 받을 수 있다.

MyProxy 서버 설정은 myproxy-server.conf 파일에 다음과 같이 설정한다.

```
authorized_retrievers "*"
pam "sufficient"
certificate_issuer_cert /root/.globus/simpleCA/cacert.pem
certificate_issuer_key /root/.globus/simpleCA/private/cakey.pem
certificate_issuer_key_passphrase "xxxxxx"      # Password
certificate_serialfile /root/.globus/simpleCA/serial
certificate_out_dir /root/.globus/simpleCA/newcerts
certificate_mapfile /etc/grid-security/grid-mapfile
```

2.2 OpenLDAP 서버 설치

본 문서에서는 LDAP의 설치를 OpenLDAP를 사용하여 구축한다.

○ 설치환경

일반적인 리눅스 시스템 또는 유닉스 시스템이면 된다. 윈도우에서 설치하시려는 분은 다음의 URL을 참고하시기 바란다.

- http://www.openldap.org/faq/data/cache/99.html
- http://www.openldap.org/faq/data/cache/115.html

○ BackEnd DB에 대해서

OpenLDAP는 그 자체의 저장구조를 가지고 있지 않고 다양한 저장구조를 붙여서 사용을 할 수가 있다. 버클리DB, GDBM, NDBM, Shell, Passwd, 심지어 SQL BackEnd도 지원을 한다. 다양한 BackEnd DB를 지원하지만 오직 최고의 성능을 내기위해 OpenLDAP측에서 권장하는 BackEnd DB는 버클리DB이다. 이것도 버전에 따라서 지원하는 것이 다른데, OpenLDAP 1.2.x버전의 경우는 버클리DB 2.7.7버전을 (즉DB2를) 사용하기를 권하고 있고 OpenLDAP 2.x버전의 경우에는 버클리DB 3.1버전을 (즉 DB3를) 사용하기를 권장하고 있다. 여러분이 어떤 것을 설치하느냐에 따라서 달라진다는 것을 유의하기 바란다. 다음의 사이트에서 OpenLDAP와 DB2/3를 구할 수 있다.

http://www.openldap.org
http://sleepycat.com

○ Berkeley DB 컴파일

소스 db-2.7.7 또는 db-3.1을 /usr/local/src 디렉토리에 풀었으며 모든건 root로 작업한다.

```
# cd /usr/local/src/db-2.7.7        (소스 디렉토리로 이동)
# cd dist                           (컴파일은 이 디렉토리에서 한다)
# ./configure --prefix=/usr/local/db2  (prefix는 DB가 설치될 디렉토리이다)
# make                              (Berkeley DB library 생성)
# make install                      (Berkeley DB library 설치)
```

○ 추가적인 옵션이 필요하다면 ./configure --help 명령으로 찾아본다.

○ OpenLDAP 컴파일

openldap-2.0.7 또는 openldap-2.3.7 소스는 /usr/local/src 디렉토리에 풀었으며 root로 작업한다.

```
# cd /usr/local/src/openldap-2.3.7          (소스 디렉토리로 이동)
# ./configure --prefix=/usr/local/ldap (/usr/local/ldap를 prefix로 주고 컴파
일)
# make depend                          (Build dependencies)
# make                                 (Build the system)
# cd tests                             (Test the standalone system)
# make
# cd ..
# make install                    (install the binaries and man pages)
```

세세한 옵션들이 많이 있으므로 configure --help로 자신에게 맞게 옵션들을 골라 써주면 된다. 그리고 참고로 만일 버클리DB와 GDBM이 같이 깔려있을 때 OpenLDAP는 우선적으로 버클리DB부터 찾아보게 되는데 만일 자신이 GDBM을 BackEnd DB로 설정하여 컴파일 하고 싶다면 --with-ldbm-api=gdbm 과 같은 옵션을 명시해 주면 된다. (하지만 GDBM이 버클리DB에 비해 성능이 많이 떨어진다.)

□ 환경 설정

○ 위에서 /usr/local/ldap를 prefix로 주었기 때문에 /usr/local/ldap 디렉토리 이하에 모든 파일이 존재한다. 우리는 LDAP 데몬을 띄우기 위해 /usr/local/ldap/etc/openldap/slapd.conf 파일을 편집해야 한다.

○ 설정의 기본 방식

다음은 slapd.conf의 기본 설정 방식을 보여준다.

```
# 전체에 영향을 미치는 설정
<global config directives>
스키마 include라든지 전역적 ACL설정등이 여기에 온다.

# first database definition & config directives
database <typeA>
<database-specific directives>
관리자 dn, 패스워드, 데이터 디렉토리, 지역ACL설정등이 여기에 온다.

# second database definition & config directives
database <typeB>
<database-specific directives>
관리자 dn, 패스워드, 데이터 디렉토리, 지역ACL설정등이 여기에 온다.
```

위와 같이 두 개이상 의 database지시자가 올수가 있어서 한 개의 서버에 다중 데이터베이스를 구축할 수도 있다. Ex) dc=database,dc=sarang,dc=net과 dc=plug,dc=sarang,dc=net을 함께 관리할 수가 있게 된다.

위의 간단한 전체 구조를 보았지만 사실 아직 잘 이해가 안가리라고 생각한다. 다음의 대략적인 전체 설정 파일을 보고 이해한 다음 다시 여기로 와서 본다면 무슨 뜻인 지 감이 잡힐 것이다.

○ 대략적인 전체 설정 파일

slapd.conf파일의 내용을 간단히 살펴보자. 여기에는 빠진 옵션들이 많다.

```
#
# See slapd.conf(5) fordetails on configuration options.
# This file should NOT be world readable.
#

include         /usr/local/ldap/etc/openldap/slapd.at.conf
# objectclass에 대한 attribute를 정의해 놓은 파일을 include한다.

include         /usr/local/ldap/etc/openldap/slapd.oc.conf
```

```
    # objectclass를 정의해 놓은 파일을 include한다.

    schemacheck       off
    # LDAP add,modify등을 수행할 때 입력되는 스키마 데이터가 올바른지
점검을 한다.
    # off라 두면 점검 과정이 빠지므로 그만큼 수행속도는 빠르게 되지만
그 만큼 입력되는 데이터가 신뢰성을 뒷받침해주어야 할 것이다.

    pidfile           /usr/local/ldap/var/slapd.pid
    # slapd 프로세스 id number

    argsfile          /usr/local/ldap/var/slapd.args
    # slapd argument
    ######################################################
    # ldbm database definitions
    ######################################################

database          ldbm

    # suffix란  ldbm에서의 최상위 Base Dn을 설정하는 작업으로 실제로
suffix에 해당하는 엔트리 데이터를 나중에 입력해준다.
    suffix            "dc=database, dc=sarang, dc=net"

    # rootdn : 관리자의 아이디를 이렇게 dn형식으로 입력한다.
    rootdn            "cn=DsnManager, dc=database, dc=sarang, dc=net"

    # rootpw : 관리자 패스워드를 설정한다.
    rootpw            dkagh

    #
    # DB 데이터가 저장되는 디렉토리
    # 만일 기본설치 디렉토리외에 자신이 지정하고 싶은 디렉토리를 설정한
다면 반드시 mkdir로 디렉토리를 직접 만들어 준 후 slapd를 구동하여야 한
다. 여기에 그냥 디렉토리 설정을 해놓는다고 해서 slapd가 뜰 때 자동으로
존재하지 않는 디렉토리를 만들어 주진 않는다.
    directory         /usr/local/ldap/var/openldap-ldbm
```

ISBN 978-89-6211-679-3

○ slapd.conf 의 중요한 설정

OpenLDAP의 설정파일인 slapd.conf는 의외로 간단한 구조를 가졌으면서도 관리자의 설정 여하에 따라서 매우 복잡한 설정을 가질수 있고 이에 따라 매우 유연하고 강력한 설정을 해줄 수도 있게 된다. 이러한 slapd.conf의 몇 가지 자세히 알아보아야 할 설정을 이제부터 살펴보기로 한다.

1) ACL(Access Control List)

ACL이란 마치 우리가 파일시스템에서 파일과 디렉토리에 소유자와 권한을 설정해 주는 것과 같이 각각의 엔트리에 사용자마다의 접근권한을 설정해 주는 일을 한다. 그리고 이러한 ACL설정으로 LDAP내 각각의 entry,attribute의 접근에 대해 권한을 설정할 수가 있으며 접근할 수 있는 호스트에 대한 설정도 할 수가 있다. 그러므로 LDAP관리자는 반드시 데이터의 중요도에 따라서 ACL을 설정해야하며 이러한 설정이 올바르게 적용되고 있는지 확인도 해야 할 것이다.

ACL의 기본형식은 다음과 같다.

```
access to <권한제한을 설정할 것>
        by <누구에게> <엑세스 권한>
```

그럼 일반적인 설정 예제를 한번 보자.

```
defaultaccess none
access to *
   by self write
   by dn=".+" read
   by dn="^$$" read
   by * none
```

위의 내용을 해석하면 다음과 같다. 우선 기본 접근권한을 none으로 설정한다. (defaultaccess none) 그리고 'access to *' 부분은 모든 엔트리(*)에 대한 접근권한을 정의하겠다는 뜻이다. 정규표현식을 써서 'access to dn=".+,dc=database,dc=sarang,dc=net" '이라고 하면 dc=database,dc=sarang,dc=net을 dn의 한 부분으로 가지는 모든 엔트리에 대해 접근권한을 정의하겠다는 뜻이다. 이제 그 아래에서부터 모든 엔트리에 대

한 접근권한 정의가 시작된다. ‘by self write’란 엔트리자신(self)이 자기 자신의 엔트리에 대해 쓰기권한을 가질 수 있다는 것이다. 좀 혼란스러운 내용일수가 있는데 만일 cn=홍길동,dc=example,dc=com 이라는 person 엔트리가 LDAP에 저장되어 있다면 이 엔트리 자신은 즉 홍길동이란 사람은 자신의 dn을 제시하고 userPassword를 제시함으로써 자기 자신의 정보에 쓰기권한을 가질 수가 있다는 말이 된다. 그러므로 홍길동이란 사람은 cn=짱구,dc=example,dc=com이라는 엔트리의 내용에는 쓰기 접근을 할 수가 없다. 물론 관리자 dn과 패스워드로 접근한다면 어떤 엔트리에도 쓰기작업이 허용된다. ‘by dn=".+" read’ 이 말은 dn이 존재하는 즉 dn이 존재하고 패스워드를 제시해서 인증된 사용자에게만 read권한을 주겠다는 뜻이 된다. (OpenLDAP 2.x에서라면 이 ‘인증된 사용자’라는 것을 dn=".+"에서 users로 바꿀 수가 있다.) ‘by dn="^$$" read’ 이것은 dn과 패스워드를 제시하지 않은 사용자 즉 anonymous 사용자에게 read권한을 주겠다는 뜻이다.(OpenLDAP 2.x에서라면 dn="^$$"부분을 anonymous로 바꿀수가 있다.) ‘by * none’란 그 외의 모든 상황에 대해서는 권한을 주지 않겠다는 뜻이다.

이제 특정 attribute에 대해서 설정을 해보자.

```
access to attr=userpassword
  by self write
  by * none
```

이라고 설정을 한다면 userpassword속성에 대해서만 접근권한을 정의하게 된다. 그리고 이때 권한을 설정하고자 하는 것이 복수개라면 콤마를 구분자로 다음과 같이 쓴다.

```
access to attrs=userpassword,uid,sn
```

attr에서 복수개일 때 attrs로 바뀐 것에도 주의해야 한다. 좀 더 강력한 사용 예를 살펴보자.

```
access to dn=".*,dc=(.*),dc=(.*),dc=net" attrs=children,entry,uid
  by dn="cn=Administrator,dc=$1,dc=$2" write
```

```
        by dn=".+" read
        by * none
```

위에서 보는 바와 같이 정규표현식의 그룹핑(grouping)도 쓸 수가 있다. 또한 호스트에 대해서도 설정을 해줄 수가 있다.

```
    defaultaccess none
    access to dn=".*,dc=database,dc=sarang,dc=net"
       by domain="localhost" write
       by addr="150.183.xxx.xxx" read
```

위 예제(OpenLDAP 1.2.x기준)는 도메인 방식으로 로컬호스트에 대해서는 쓰기권한을 부여하고, 아이피 지정방식으로 해당 아이피에 대해서는 읽기권한을 부여하고 있다.

Group방식 ACL도 있다.
http://www.openldap.org/faq/index.cgi?_highlightWords=acl&file=52

OpenLDAP 2.x 버전의 ACL포맷은 다음과 같다.

```
<access directive> ::= access to <what>
                [by <who> <access> <control>]+
        <what> ::= * | [ dn[.<target style>]=<regex>]
                [filter=<ldapfilter>] [attrs=<attrlist>]
        <target style> ::= regex | base | one | subtree |
children
        <attrlist> ::= <attr> | <attr> , <attrlist>
        <attr> ::= <attrname> | entry | children
        <who> ::= [* | anonymous | users | self |
                dn[.<subject style>]=<regex>]
                [dnattr=<attrname> ]
                [group[/<objectclass>[/<attrname>][.<basic
style>]]=<regex> ]
```

```
                [peername[.<basic style>]=<regex>]
                [sockname[.<basic style>]=<regex>]
                [domain[.<basic style>]=<regex>]
                [sockurl[.<basic style>]=<regex>]
                [set=<setspec>]
                [aci=<attrname>]
        <subject style> ::= regex | exact | base | one | subtree
| children
        <basic style> ::= regex | exact
        <access> ::= [self]{<level>|<priv>}
        <level> ::= none | auth | compare | search | read |
write
        <priv> ::= {=|+|-}{w|r|s|c|x}+
        <control> ::= [stop | continue | break]
```

ACL에 대해서 간단히 알아보았다. 다음과 같은 ACL에 대한 주의점을 유의해야 한다.

- 위의 ACL들은 하나 이상 여러 개의 리스트를 설정하여 쓸 수도 있다. 그리고 local.acl.conf같은 파일 이름으로 ACL을 설정하고 slapd.conf에서 include지시자를 사용하여 쓸 수도 있다.

- 만일 동일한 액세스 항목을 두개 이상 설정하게 되는 실수를 저지른다면 첫 번째 설정 값만 실행하게 되고 두 번째 설정은 적용되지 않게 된다.

- ACL정의내의 entry,attribute내용을 작성할 때 콤마(,)앞뒤로 공백문자가 있어서는 안 된다. dn=".*, dc=example, dc=com"과 같이 콤마앞뒤로 하나라도 공백문자가 있는 날에는 여러분이 설정한 ACL이 전혀 작동하지 않는 수가 있다. slapd를 띄울 때 아무 에러메시지가 없다고 해서 ACL이 잘 작동하는 것이 아니라는 뜻이다.

- ACL리스트의 순서가 문제가 될때가 있다. ‘access to *’와 ‘access to dn=".*,dc=example,dc=com"’이 설정되어 있다면 좀더 특정 ACL리스트라

고 할수 있는 'access to dn=.*,dc=example,dc=com"'부분이 보다 앞에 위치하여야 한다. 앞의 설정을 포함할 수 있는 좀 더 넓은 범위의 ACL을 뒤로 놓는 것이 안전하다.

- ACL이 복잡하면 할수록 ldapsearch에 드는 비용은 비싸기 마련이다. 특히 group ACL은 더욱 그렇다. 그러므로 최소한의 필요한 최적의ACL만을 설정하는 것으로 좀 더 나은 검색속도를 가질 수가 있게 된다. 결국 ACL이 간단하면 간단할수록 정규표현식이 적으면 적을수록 퍼포먼스 향상에 많은 도움이 된다.

- ACL이 올바로 뜨는지 검색 시 어떤 영향을 미치는지 알고 싶다면 다음과 같이 slapd를 띄울 때 './libexec/slapd -f etc/openldap/slapd.conf -d 133' 으로 띄우면 ACL이 올바로 뜨는지를 디버깅할 수 있고 틀린 문법을 올바로 잡아서 확인이 가능하다.

2) 패스워드 인증방식 설명

OpenLDAP는 RFC 2307 패스워드 방식을 지원한다. 이 방식에는 {CRYPT}, {SHA}, {SSHA}, {MD5}, {SMD5} 등의 방식이 있다. 이러한 패스워드 방식은 slapd.conf내의 rootpw와 attribute인 userPassword에 사용될 수 있다. RFC 2307 패스워드 방식에서 {SSHA}방법이 일반적으로 널리 사용되고 crack하기가 어렵다. 반면 {CRYPT}는 특성상 운영체제 의존적인 문제가 있고 (그러므로 항상 slapd가 떠있는 시스템에서 crypt하여 생성해야 함) crack되기가 쉬워 그리 권장할만한 방식이 아니다.

(1) plain text

가장 단순한 방식이며 LDAP데이터의 외부노출에 대해 크게 신경쓰지 않아도 되는 곳이나 보안에 대해 크게 신경을 쓰지 않아도 되는 곳이라면 이 방식을 쓰면 된다. 다음은 입력되는 데이터의 예를 보여준다.

```
rootpw  secret    #   slapd.conf의 관리자 패스워드 설정부분이다.
userPassword: secret
```

(2) crypt

위에서 말한 것처럼 이 방법은 그리 권장하는 방법은 아니지만 손쉽게 쓸 수가 있다. 그리고 꼭 주의해야 할 점은 이 CRYPT방식은 운영체제 의존적이라 항상 slapd 데몬이 떠있는 시스템에서 패스워드 생성을 해야 한다. /etc/passwd(또는 /etc/shadow)에 있는 패스워드를 긁어서 userPassword 또는 rootpw에 '{CRYPT}encrypted_password' 와 같이 입력을 한다. 그런 다음 테스트를 해보면 잘 될 것이다. 또는 여러분이 직접 스크립트를 작성하여 패스워드를 생성해도 좋다. perl을 사용하여 다음과 같이 해본다.

```
perl -e 'print("{CRYPT}" . crypt("secret","salt") . "₩n");'
```

secret는 패스워드를 말하는 것이고 다음 인자는 salt값을 말하는 것이다. 대부분의 시스템에서 잘 작동할 것이지만 다음과 같이 해야 되는 경우도 있을 것이다.

```
perl -e 'use Crypt::PasswdMD5;
print("{crypt}" . unix_md5_crypt("secret","salt") . "₩n");'
```

위의 생성 예는perl 모듈 Crypt::PasswdMD5를 필요로 하고 MD5가 설치되어 있어야 한다. 그리고 위의 예가 MD5를 포함하고 있지만 여전히 {crypt}라는 것을 기억해야 한다.

(3) md5, smd5

다음의 스크립트로 MD5방식의 패스워드를 생성할 수 있다.

```
#/usr/bin/perl
use MIME::Base64;
use MD5;
my $passwd = 'secret';
my $md5 = new MD5;
print "{MD5}" . encode_base64($md5->hash($passwd,''));
```

(4) sha, ssha

OpenLDAP v2.x 부터 slappasswd라는 커맨드라인 명령어가 포함되었다. 그래서 아주 간단히 {crypt}, {md5}, {smd5}, {sha}, {ssha} 패스워드를 만

들어낼 수가 있다. 사용방법은 다음과 같다.

```
# slappasswd -h {sha}
```

위와 같이 치면 패스워드를 물어보고 두 번 확인 입력을 하면 해싱(Hashing)되어 생성된 패스워드를 알려준다. 그냥 아무 인자 없이 slappasswd만 치게 되면 기본값으로 {ssha}가 넘겨진다. 이것을 복사하여 slapd.conf의 rootpw에 갖다 붙이거나 하면 된다. 좀 더 자세한 사항이 필요하다면 맨페이지를 참고면 된다.

[참고] TLS/SSL , Kerberos, SASL
다음의 링크를 따라가면 유용한 정보를 얻을 수 있다.

- How do I use TLS/SSL ?
http://www.openldap.org/faq/data/cache/185.html

- Using TLS/SSL with slapd, a quick guide
http://www.openldap.org/lists/openldap-devel/199908/msg00039.html

○ 기본적인 OpenLDAP의 콘솔명령 사용

pstree|grep slapd 로 slapd가 떠있는지를 확인했다면 이제 ldap를 가지고 테스트를 해볼 준비가 된 셈이다. 모든 콘솔명령어에는 공통적으로 들어가는 커맨드라인 옵션이 있다. 그 옵션들은 다음과 같다.

-h : 접속할 LDAP호스트를 설정한다. 설정하지 않으면 로컬호스트를 검색.
-p : 접속할 LDAP호스트의 포트번호를 명시한다. 설정하지 않을 경우의 디폴트값은 389번 포트이다.
-D : 접속자의 bind dn이다. 마치 우리가RDBMS에 접속할 때 아이디와 비밀번호가 필요한 것처럼 -D에 자신이 알고 있는 엔트리 dn을 제시한다.
-W 또는 -w password : 비밀번호를 제시한다. -W와 -w의 차이점은 -w뒤에는 비밀번호를 적어줘야 하지만 -W뒤에는 비밀번호를 적어주지 않고 프로그램이 오퍼레이션을 시작하

기 전에 사용자에게 물어보도록 한다.
-d debuglevel : 디버깅을 해야 할 필요가 있을 때 필요하다.

1. ldapadd
이제 처음으로 띄운 LDAP 서버에 데이터를 한번 넣어보도록 하자. 커맨드라인 명령을 사용할경우 두가지 방법으로 데이터 입력이 가능하다.

(1) 커맨드라인상에서 바로 입력
커맨드라인 상의 입력은 대개 단순히 하나의 데이터를 입력할 때 편하다.

```
$ ldapadd -D "cn=DsnManager,dc=database,dc=sarang,dc=net" -W << EOF
> dn: dc=database,dc=sarang,dc=net
> dc: database
> objectclass: dcobject
> EOF                          입력의 끝에 EOF를 입력하여 끝임을 알린다.
Enter LDAP Password: dkagh              여기에 암호를 입력한다.
adding new entry dc=database,dc=sarang,dc=net
```

위의 작업은 DIT의 최고 상위 엔트리를 입력하는 작업이다. 이제부터 앞으로 입력하는 엔트리는 모두 이 엔트리의 아래에 위치하게 된다. 그리고 dcobject objectclass의 스키마를 보고 요구사항에 틀리지 않도록(1.2.x버전이라면 etc/openldap/slapd.oc.conf 파일에 있고 2.0.x대 버전의 경우 etc/openldap/schema/core.schema 파일에 정의되어 있다.) 해주어야 한다. 그렇지 않을 경우 입려과정에서 에러가 난다. 만일 문법에 틀리는 데도 입력이 잘된다면 slapd.conf에 schemacheck 부분이 off로 되어 있지 않나 보기 바란다. (하지만 입력된 데이터의 옳고 그름을 점검하는 과정도 부하라고 할 수가 있으므로 신뢰할만한 관리자들만이 입력,수정을 할 경우에는 off로 해놓는 것이 더 나을 수 있다. 그리고 1.2.x의 slapd.conf의schemacheck은 디폴트가 off이지만 2.x의 schemacheck은 디폴트가 on이라는 점도 참고할 필요가 있다.)

(2) LDIF파일 포맷으로 입력

다음 섹션에서 좀더 자세히 설명하겠지만 LDIF(LDAP Data Interchange Format)란 LDAP의 데이터를 텍스트형식으로 표현하는 데이터 포맷이다. LDIF포맷의 파일로 입력하면 똑 같은 타이핑 작업을 피할 수 있을 뿐 아니라 백업된 데이터의 복구에도 이 방식이 쓰일 수 있으므로 유용하다고 할 수 있다. Vi와 같은 편집기를 열어 위의 내용을 dsnroot.ldif와 같은 파일이름으로 저장을 한다.

```
$ ldapadd -D "cn=DsnManager,dc=database,dc=kisti,dc=re,dc=kr" -W -f dsnroot.ldif
Enter LDAP Password: xxxx
adding new entry dc=database,dc=kisti,dc=re,dc=kr
```

우리가 입력한 데이터가 잘 입력이 되었는지 확인 작업으로 넘어가기 전에 이 LDIF로 입력할 때의 주의사항이 한 가지 있다. 흔히 실수할 수 있는 일인데 문제는 항상 LDIF포맷 파일에 입력할 엔트리가 두 개 이상일 경우에 발생한다. 두개 이상의 엔트리일 경우 엔트리간의 간격은 빈줄 한 줄로 꼭 유지하여야 한다. 꼭 한 줄이어야 한다. 두 줄 이상이 되어버리면 첫 엔트리만이 입력이 되고 두 번째 엔트리 데이터부터는 입력이 안 된다. 그리고 입력되는 엔트리 순서에도 신경을 써주어야 한다.

2. ldapsearch

LDAP를 검색할 때 우리가 SQL을 사용하여 일반적으로 관계형 데이터베이스를 검색하는 것과 같이 검색조건을 설정하고 검색을 하는데, 다음과 같은 네 가지 주목할 만한 검색조건에 대해서 말을 해보자. (여기서 언급하지 않은 사항들에 대해서는 ldapsearch의 맨페이지를 참조하면 될 것이다. 그리고 이 내용은 ldapseach명령에만 해당되는 것이 아니라 LDAP의 검색에 관련된 모든 함수들이나 명령들에도 이 사항들은 공통적으로 해당이 된다.)

(1)search scope

LDAP의 검색 영역을 지정하는 것으로 아래의 세개의 그림으로 충분히 이해가 가리라 생각이 된다. 그리고 base dn 이라는 말을 알아야 하는데 검색을 하는 기준점을 뜻하는 말이다. 무조건 base dn이하 엔트리들에서 검색을 하는데 그

종류가 아래와 같이 세 가지로 나뉜다는 것이다. base란 검색하는 base dn으로 제시한 그 엔트리 자체만을 검색한다는 뜻이다. onelevel이란base dn아래 한단계 하위 엔트리들까지 에서만 검색을 한다는 것이다. subtree란 base dn 아래의 모든 엔트리에서 검색을 한다는 뜻이다.

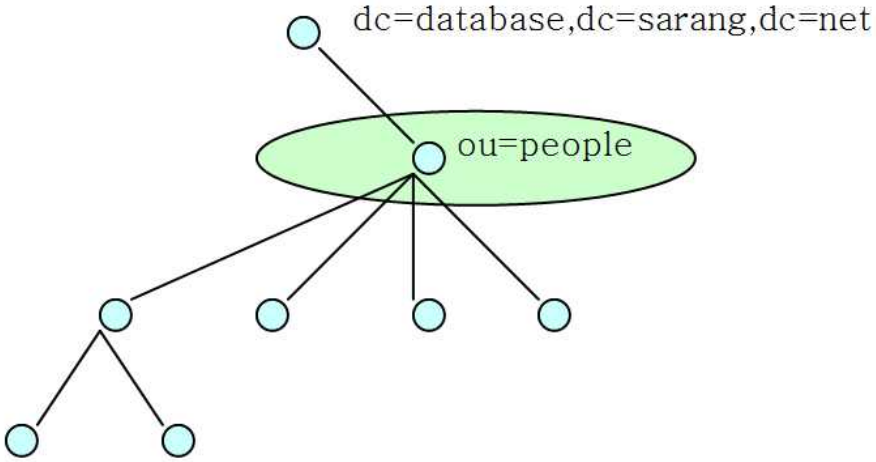


그림 8 base scope

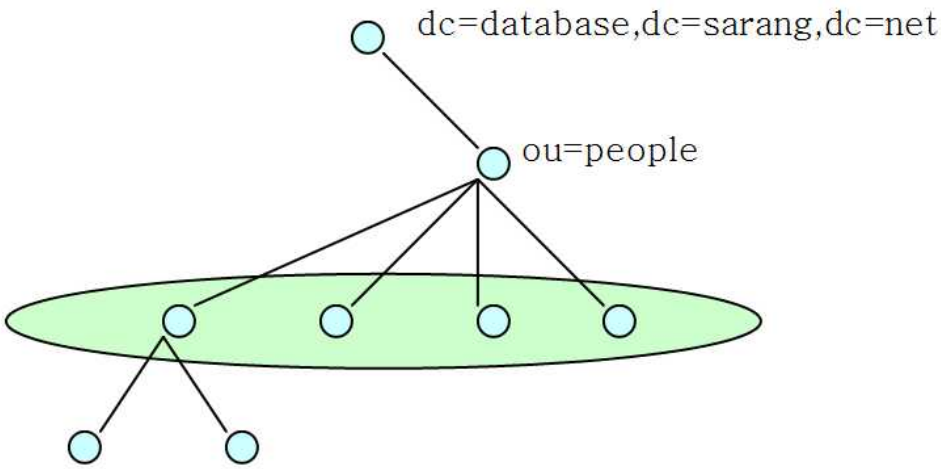


그림 9 one level scope

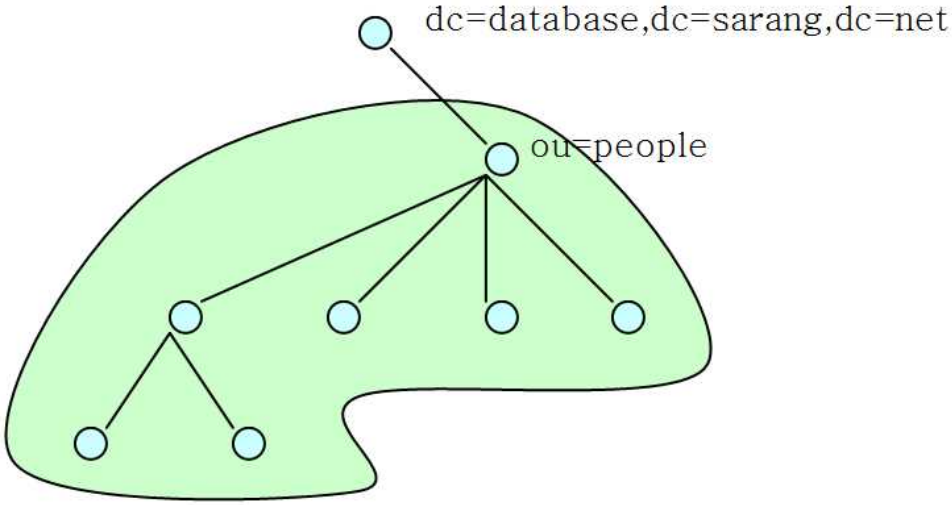


그림 10 subtree scope

(2)search filter

LDAP의 다양한 검색 필터에 대해서 알아보자. 대충 여덟가지 정도의 검색 필터를 사용할 수가 있다.

- equality : '(cn=홍길동)' 과 같이 완전 매칭이 이루어지는 검색 필터
- substring : '(cn=*길동)' '(cn=홍길*)' '(cn=*길*)' 과 같이 부분 문자열이 같은 것을 검색하는 필터
- approximate : 말뜻 그대로 유사한 단어를 검색하는데 쓰이는데 한글은 되지 않고 영어의 경우 'pipe'를 검색하면 piping 이라든지 유사단어를 검색
- less than,greater than : '(cn>=abc)' 단어순서상의 더 크고 더 작은 조건을 만족하는 검색 결과를 얻을 수 있게 해 주는 검색필터
- presence : '(cn=*)' 와 같이 cn attribute가 있고 없고를 따지는 검색필터
- and : (&(cn=dynam)(telephoneNumber=*1374*)) 과 같이 필터간의 AND 조건 검색을 만들어 준다.
- or : (|(cn=Nam)(telephoneNumber=010*)) 와 같이 필터간의 OR조건 검색을 만들어 준다.
- not : (!(cn=Nam*)) 과 같이 cn이 'Nam'으로 시작하지 않는 결과를 돌려받는데 쓰이는 검색 필터

이제 이러한 필터들을 조합해서 좀더 복잡한 조건을 가지는 필터들을 만들어서 검색할 수도 있다.

(3) time limit
검색하는데 걸리는 시간에 대해서 제한을 걸어놓는다. 이것은 slapd.conf의 timelimit의 시간제한 범위 내에서만 설정가능하다. slapd.conf의 timelimit 보다 더 오랜 시간을 설정할 수는 없다.

(4) size limit
검색결과로 돌려받는 결과 데이터 크기에 제한을 한다. 이것도 위와 마찬가지로 slapd.conf의 sizelimit보다 큰 크기로 설정할 수는 없다.

3. ldapmodify
LDAP서버의 특정 엔트리에 수정을 하고 싶을 때 사용하는 명령이다.

4. ldapdelete
LDAP서버의 특정 엔트리를 지우고 싶을 때 이 ldapdelete명령을 사용하여 지운다. 쓰는 방법은 다음과 같다.

```
ldapdelete -v -D binddn -w passwd dn
ldapdelete -v -D binddn -W dn
```

binddn = "cn=DsnManager, dc=database, dc=sarang, dc=net"
dn = "지우고 싶은 dn" (leaf node만 삭제가 가능하다.)

위의 예제처럼 ldapdelete로는 항상 leaf node만이 삭제 가능하다. 하지만 leaf node가 아닌 경우에 서브트리전체를 지우고 싶을 때도 있을 것이다. OpenLDAP 2.x의 경우에 -r(recursive)옵션을 주므로써 간단히 주어진 dn이하의 서브트리를 모두 삭제할 수가 있지만 1.2.x버전인 경우에는 이것의 소스(ldapdelete.c)를 수정함으로써 우리는 이 프로그램에 -r(recursion) 옵션을 달 수가 있다. 그렇게 되면 트리의 어떤 entry라도 하부에 연결된 모든 entry들을 삭제할 수가 있게 된다. 이 패치는 다음의 URL에서 구할 수 있다.

http://www.openldap.org/lists/openldap-devel/199912/msg00014.html

5. ldapmodrdn

엔트리의 dn과 rdn을 수정하고 싶을 때에 사용한다. 수정하기 이전의 엔트리 내용은 다음과 같다.

```
dn: cn=someone,dc=database,dc=sarang,dc=net
objectclass: person
cn: someone
sn: dontknow
```

이 엔트리의 dn과rdn(cn=someone)을 수정해보기 위해서 다음과 같이 해본다.

```
ldapmodrdn -D "cn=DsnManager, dc=database, dc=sarang, dc=net" -W
"cn=someone, dc=database, dc=sarang, dc=net" "cn=kikibon"
```

이제 어떻게 변했는지 알아보자.

```
dn: cn=kikibon,dc=database,dc=sarang,dc=net
objectclass: person
cn: someone
cn: kikibon
sn: dontknow
```

위와 같이 바뀌었을 것이다. 이때 cn: someone 데이터까지 없애버리려고 한다면 ldapmodrdn에 -r 옵션을 주면 된다. 만일 파일로 저장한다음 -f 옵션으로 할 필요가 있다면 위의 두 바뀌어야 할 dn과 새로운 rdn을 차례로 적고 파일로 저장한 후 실행하면 될 것이다.

2.3 MyProxy와 OpenLDAP 연동

MyProxy와 OpenLDAP의 연동을 위해, OpenLDAP Client를 설치해 보자. openldap-client.rpm을 설치할 수도 있으며, 다음 방법은 OpenLDAP 패키지를

활용하여 설치하는 내용이다.

먼저 OpenLDAP version 2.3 이상을 다운로드 받는다. 가능하다면 설치하려는 시스템의 OS에 맞는 OpenLDAP 패키지를 활용한다. SSL/TLS 상에 LDAP을 활용할 예정이라면, TLS를 지원하는 OpenLDAP 라이브러리들이 있어야 한다.

```
./configure --with-tls
```

서버가 아닌 OpenLDAP 라이브러리들만 필요하다면 다음 명령어로 설정한다.

```
./configure --prefix=/usr/local/openldap-for-myproxy --disable-slapd
```

다음으로 Globus Toolkit을 설치한다. MyProxy와 동작하는 GT 2.4 이상의 버전을 활용한다. 그리고 다음의 작업 전에 X.509 user 및 host 인증서를 가지고 있어야 한다.

MyProxy 최신 버전의 압축을 풀고 다음을 수행한다.

```
./configure --with-flavor=gcc32dbg --with-openldap=/usr
```

openldap 설치 위치를 자신의 시스템에 맞게 수정한다. 또한 /lib 와 /include 디렉토리들을 맞게 수정한다. myproxy-server에서 LDAP 확장을 적용하기 위해 컴파일을 재수행 한다.

```
make install
```

마지막으로 MyProxy Administrator's Guide에 따라 MyProxy 서버를 설정한다. 특히 myproxy-server.config 파일에서 ca_ldap 옵션을 맞게 수정해야 한다.

○ MyProxy 서버가 LDAP를 참조하기 위해서는 /etc/pam.d/myproxy 라는 파일을 생성하여 다음과 같이 입력해 준다.

```
[eairs@cyber opt]$ more /etc/pam.d/myproxy
#auth    required          pam_unix.so
#account          required          pam_unix.so

auth      required          pam_ldap.so
account required          pam_ldap.so
```

○ LDAP 서버에서 사용자의 인증을 조회하기 위해서는 /etc/passwd 파일에 아래와 같이 설정해 준다.

```
[eairs@cyber opt]$ tail /etc/passwd
eairs:x:500:500:e-AIRS User:/home/eairs:/bin/bash
+@IBMAdm
+@IBMUser
+@SUNAdm
+@SUNUser

[eairs@cyber opt]$
```

○ LDAP과 PAM 연동을 위한 system-auth 설정

```
[eairs@cyber opt]$ more /etc/pam.d/system-auth-ac
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.

auth        required      /lib/security/$ISA/pam_env.so
auth        sufficient    /lib/security/$ISA/pam_unix.so likeauth nullok
auth        sufficient    /lib/security/$ISA/pam_ldap.so use_first_pass
auth        required      /lib/security/$ISA/pam_deny.so

account     required      /lib/security/$ISA/pam_unix.so broken_shadow
account     sufficient    /lib/security/$ISA/pam_succeed_if.so uid < 100 quiet
account                   [default=bad success=ok user_unknown=ignore]
/lib/security/$ISA/pam_ldap.so
account     required      /lib/security/$ISA/pam_permit.so

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

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

ISBN 978-89-6211-679-3

```
[eairs@cyber opt]$
```

○ grid-mapfile 생성

```
[eairs@cyber opt]$ more /etc/grid-security/grid-mapfile
"/C=kr/O=KISTI/CN=Dukyun Nam" dynam
[eairs@cyber opt]$
```