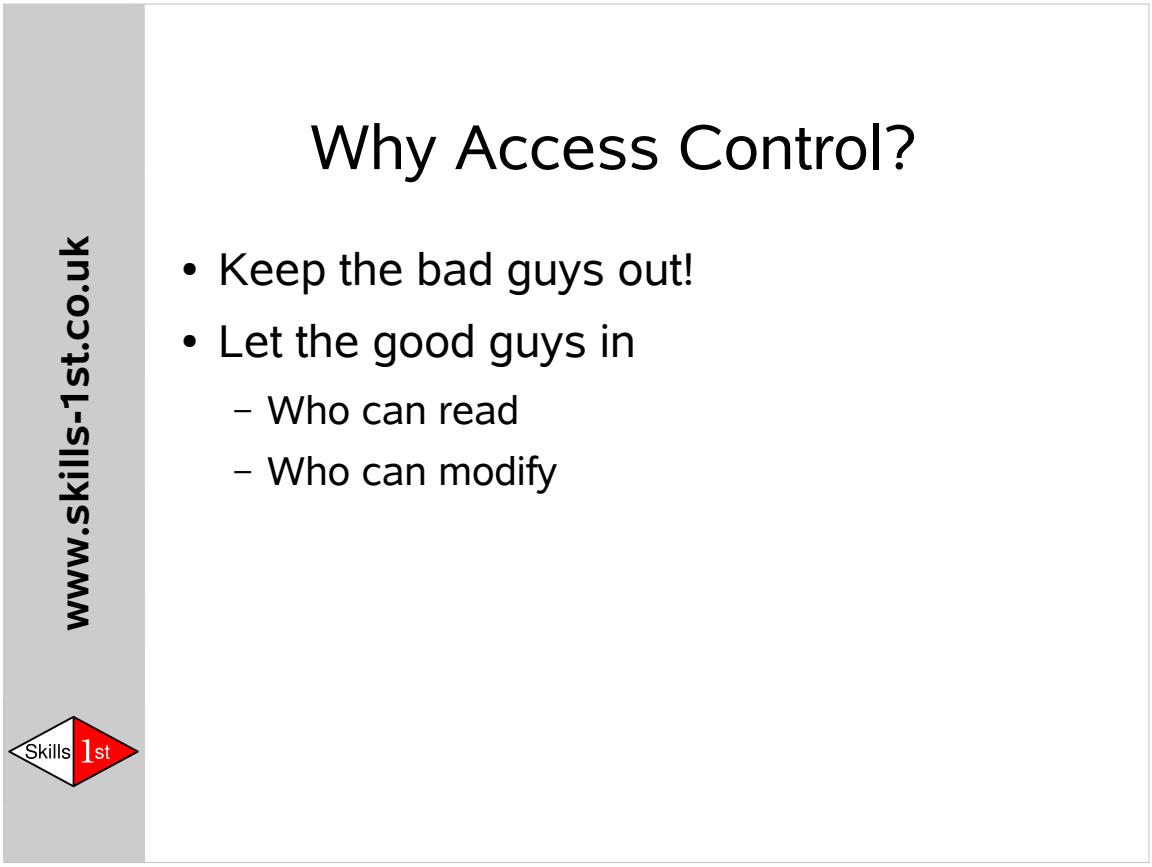
www.skills-1st.co.uk

Access Control Policies for LDAP

Andrew Findlay
Skills 1st Ltd

andrew.findlay@skills-1st.co.uk

January 2009



When the subject of access control is first raised in an LDAP project, people usually start by talking about who should be kept *out*. The designer should turn the questions around, as it is far more useful to ask who should be *allowed* to see a given item.

Concepts

- Subject – *who*
- Object – *what*
- Access – *permitted actions*
  - Read / Add / Delete entries
  - Read / Search / Modify attributes
- ACI – Access Control Item
- ACL – Access Control List

Access control rules are usually defined as lists, where each item says something about *who* should have what *access* to some set of *objects*.

www.skills-1st.co.uk

Design Process

- Define the requirements
  - Subjects: define groups
  - Objects: define categories
  - Allow for data management
  - Verify application requirements
  - Refine with examples
- Build a test suite
- Write ACLs

Policies should refer to groups or classes of user even if some groups are empty or only have one member. Groups might be “personnel administrators”, “departmental secretaries”, “all authenticated users”, “the mail system” etc.

Objects (or targets) are entries or individual attributes. They might be “all entries describing people”, “entries describing groups”, “public attributes”, “sensitive attributes” etc.

Directory data has to be loaded and maintained. The policy must allow for this: don't use the all-powerful “root” user for routine work.

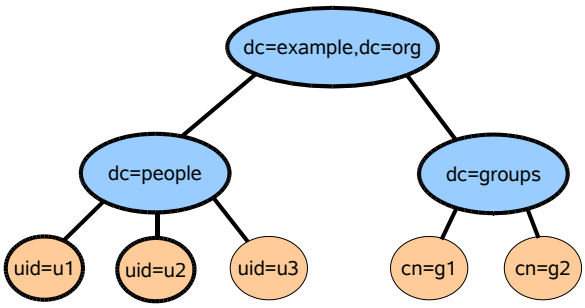
Most directory access comes from automatic processes like mail systems: each needs an account with suitable access.

Draw the proposed DIT and use it to discuss examples.

Every example should lead to a test.

www.skills-1st.co.uk

Simple Policies



- Read only
- Data admin
- Admin group

- Read-only: “everyone in the world may read every attribute of every entry (except for passwords)”
- Data admin: “as above, but *this* named entry can modify everything”
- Admin group: “Anyone in *this* named group can modify any user entry”

www.skills-1st.co.uk

Design Principles

- ACLs are *programs*
- Have few ACLs
- Avoid routine ops involving ACLs
- Use attributes to trigger ACIs
- Write the tests *first*
- Don't mix grants and denys
- Give access to groups, not individuals

ACLs are hard to get right, and hard to check comprehensively. Don't allow them to proliferate, and avoid having routine processes touch them.

Add local attributes to the schema so that data in entries can trigger global access rules.

Test-driven design works well.

More Principles

- Use DIT Content Rules
- Make DNs opaque
  - uniqeIdentifier=A674EC43
- Avoid spaces and punctuation in RDNs

Access control is not just ACLs.
DIT Content Rules give useful control over new and modified entries.
Don't put useful information into DNs: it is very hard to protect it, and you don't ever want to change the DN of an entry once created.
Keep the DNs simple: they don't mean anything now, so make it easy to write filters to match them.

www.skills-1st.co.uk

Server capabilities

- IBM TDS
  - ACLs in DIT, at the control point
  - Filters
- Sun / Netscape / Red Hat / Fedora
  - ACLs in DIT, anywhere above control point
  - Filters, macros
- OpenLDAP
  - ACLs outside DIT, program-like
  - Filters, regular expressions, sets...

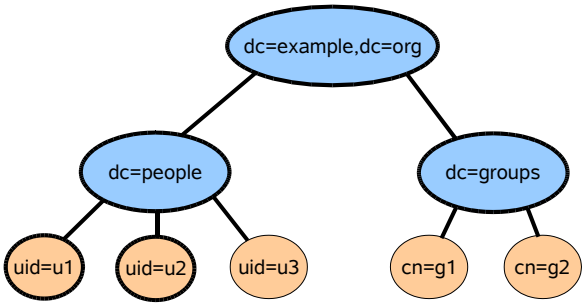
Thee LDAP servers with a common UMich heritage, but very different access-control languages.
Oracle Internet Directory has similar capabilities to Sun / Netscape, but a different syntax.
Apache DS is following X.500(1993)

www.skills-1st.co.uk

Skills 1st

Example: user registry

- To authenticate users: ID and password
- Public read
- Users can change their own passwords
- Passwords not readable by anyone



This very simple policy generated about 20 tests...

www.skills-1st.co.uk

ACLs for TDS

```
dn: dc=example,dc=org
changetype: modify
add: ibm-filterAclEntry
ibm-filterAclEntry: group:CN=ANYBODY:
 (objectclass=*):normal:rsc
ibm-filterAclEntry: access-id:cn=this:
 (objectclass=*):at.userPassword:grant:w
```

This LDIF places two ACIs at the top of the DIT. Both are filtered entries, but the filter matches all entries.

The first ACI gives read access to all “normal” attributes for all users including anonymous ones. “Normal” attributes are things like *cn, sn, mail* – not *userPassword*.

The second ACI allows users to change their own passwords.

ACLs for Sun / Netscape

```
dn: dc=example,dc=org
changetype: modify
add: aci
aci: (targetattr != "userPassword")
  (version 3.0; acl "Make public objects visible";
     allow (read, compare, search)
     (userdn = "ldap:///anyone") ;)
aci: (targetattr = "userPassword")
  ( version 3.0; acl "Users change own passwords";
     allow (write)
     (userdn = "ldap:///self") ;)
```

This LDIF places two ACIs at the top of the DIT. Both take effect on all entries.

The first ACI makes every attribute except *userPassword* visible to all users.

The second ACI allows users to change their own passwords.

ACLs for OpenLDAP

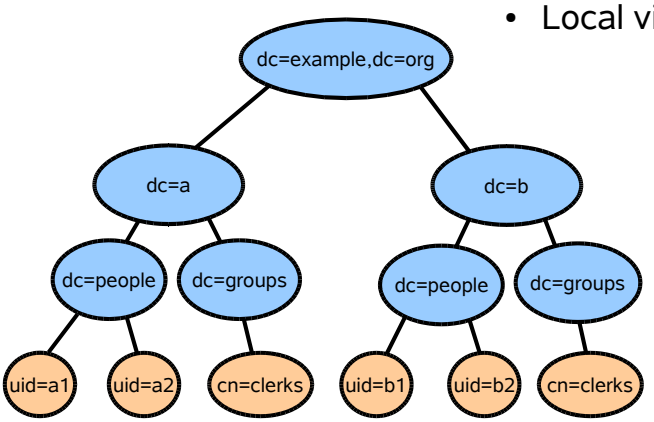
```
access to attrs="userPassword"
       by self =w
       by * auth

access to *
       by * read
```

OpenLDAP ACLs are stored outside the DIT that they control.
The first directive allows users to change their own passwords and allows any user (in practice the anonymous user) to authenticate.
The second directive allows every other attribute to be read by everyone.

www.skills-1st.co.uk

Example: Local Visibility

- Visibility attribute
- Local visibility by default



The world may see any entry that is marked with
exampleVisibility=public
Other entries can only be seen by users in the same department.

www.skills-1st.co.uk

ACLs for TDS

- Needs an ACL in each department entry
- Identify local users with a dynamic group

```
dn: cn=users,dc=groups,dc=a,dc=example,dc=org
changetype: add
objectclass: groupOfURLs
objectclass: ibm-dynamicGroup
cn: users
memberURL: ldap:///dc=people,dc=a,dc=example,dc=org??
  sub?(objectclass=*)

dn: dc=a,dc=example,dc=org
changetype: modify
replace: ibm-filterAclEntry
ibm-filterAclEntry:
  group:cn=users,dc=groups,dc=a,dc=example,dc=org:
  (objectclass=*):normal:rsc
```

The dynamic group is inside Department A. It selects all subjects with DNs in Department A's *people* arc.
The ACL for Department A grants read access to members of the group.

www.skills-1st.co.uk

ACLs for TDS

- Global ACL: passwords and public entries

```
dn: dc=example,dc=org
ibm-filterAclEntry: access-id:cn=this:
 (objectclass=*):at.userPassword:grant:w
ibm-filterAclEntry: group:CN=ANYBODY:
 (exampleVisibility=public):normal:rsc
```

All users can change their own passwords.
Entries with exampleVisibility=public are visible to everyone.

www.skills-1st.co.uk

ACLs for Sun / Netscape

- Macro selects same-department users

```
dn: dc=example,dc=org
aci: (target="ldap:///($dn),dc=example,dc=org")
  (targetattr != "userPassword")
  (version 3.0; acl
    "Users see entries in their own department";
    allow (read, compare, search)
    (userdn =
"ldap:///dc=people,[$dn],dc=example,dc=org??sub?")
  ;)
```

A single global ACL works for all departments because of the macro facility.

The DN value matched in the target clause is substituted into the userdn clause. Square brackets cause ever-shorter versions to be tried until a match is found

www.skills-1st.co.uk

ACLs for Sun / Netscape

- Filter selects public entries

```
dn: dc=example,dc=org
aci: (targetfilter =
         "(exampleVisibility=public)")
  (targetattr != "userPassword")
  (version 3.0;
   acl "Make public objects visible to all";
   allow (read, compare, search)
  (userdn = "ldap:///anyone")
 ;)
```

The target filter selects public entries.

www.skills-1st.co.uk

ACLs for OpenLDAP

```
access to dn.subtree="dc=example,dc=org"
       attrs="userPassword"
       by self =w
       by * auth

access to filter="(exampleVisibility=public)"
       by * read

access to dn.regex="(dc=[^,]+,dc=example,dc=org)$"
       by dn.subtree,expand="dc=people,$1" read
       by * break

access to * by * none
```

The first directive deals with the userPassword attribute
The second directive handles public entries
The third directive recognises access by members of the same department. The department part of the DN of the target is saved and substituted into the *by who* clause.
The final directive denies all other access.

www.skills-1st.co.uk

Controlling DIT Content

- For delegated administration
- ACLs should only allow write for correct object type
  - OpenLDAP, Netscape OK. TDS fails.
- Need to control auxiliary classes: DIT Content Rule

```
ditcontentrule ( 2.16.840.1.113730.3.2.2
        NAME 'dcrPerson'
        DESC 'Control inetOrgPerson entries'
        AUX strongAuthenticationUser
        )
```

Most LDAP servers have very lax control on new entries.
Often the best you can do is to grant add permission only when the new entry has an appropriate object class. Even that does not help in TDS.
It is very hard to control auxiliary object classes using ACLs, but DIT Content Rules do it perfectly. OpenLDAP implements these but the other servers listed do not.

www.skills-1st.co.uk

Attribute sets for OpenLDAP

- Use object class to define set
- Remember to give access to “entry”

```
objectclass ( 1.2.826.0.1.3458854.666.3.1
   NAME 'attrsetAnonVisible'
   DESC 'Attributes visible to anon users'
   AUXILIARY
   MAY ( objectclass $ cn $ sn $ displayname $
         mail $ uniqueIdentifier )
   )

access to filter="(objectclass=person)"
        attrs="entry,@attrsetAnonVisible"
        by * +rsc break
```

Use object classes to define sets of attributes for use in ACLs.

Gotchas

- Hard to hide entries entirely
  - Detection by error message
  - OpenLDAP can protect leaf entries
  - Others have no protection
- Hard to control content of new entries
  - OpenLDAP can do it
  - Sun / Netscape has some control
  - TDS has none

If a user does a search on a base object that they cannot see, they get an error if it does not exist but no error (and no result) if it does.

www.skills-1st.co.uk

Summary

- Access control needs care
- Difficulty can rise fast with policy size
- Test-driven development
- Design patterns
- Read the paper

Andrew Findlay

Andrew.Findlay@skills-1st.co.uk