Building a modern LDAP based security framework

Andrea Barisani
Gentoo Infrastructure Team
<lcars@gentoo.org>

Introduction

DISCLAIMER:

All the scripts and/or commands and/or configurations provided in the presentation must be treated as examples, you should use them at your own risk. Please review all the code before using it in any environment.

What is LDAP?

- Lightweight Directory Access Protocol
- Simple protocol for updating and searching directory services
- Designed to be fast and reliable, atomic updates (no locking required), runs over TCP
- A directory is a database containing descriptive, attribute-based information
- We will cover OpenLDAP, the most widely used Open Source implementation. Other options are Red Hat/Fedora Directory Server, Active Directory, Oracle Internet Directory, iPlanet Directory Server...

LDAP directory entry

- An entry is a collection of attributes referenced with a unique *distinguished name* (DN)
- Directory entries are arranged in a hierarchical tree-like structure

```
dn: cn=Manager,dc=pacsec,dc=jp
objectClass: organizationalRole
objectClass: simpleSecurityObject
cn: Manager
userPassword: e320499feefewFEWFDSFDSFdfje4
```

- cn is *common name*, dc is *domain component*
- attributes are defined as part of an *object class*, objects and related attributes are grouped together in schemas

LDAP directory: how can we use it?

- user account storage:
  - UNIX account attributes (uidNumber, gidNumber, userPassword, ...)
  - Microsoft Windows account attributes (using samba schema)
  - Apache auth attributes (using mod_ldap), mail routing attributes
  - custom attributes (gpgKey, gpgFingerprint, location, ...)
  - ssh authorized keys (sshPublicKey)
- UNIX groups storage
- sudo configuration storage
- The final goal is cross-platform authentication, being able to manage users globally on the LDAP server, without performing any action on the server pool (scalability for “add/revoke a user to N servers” scenarios)

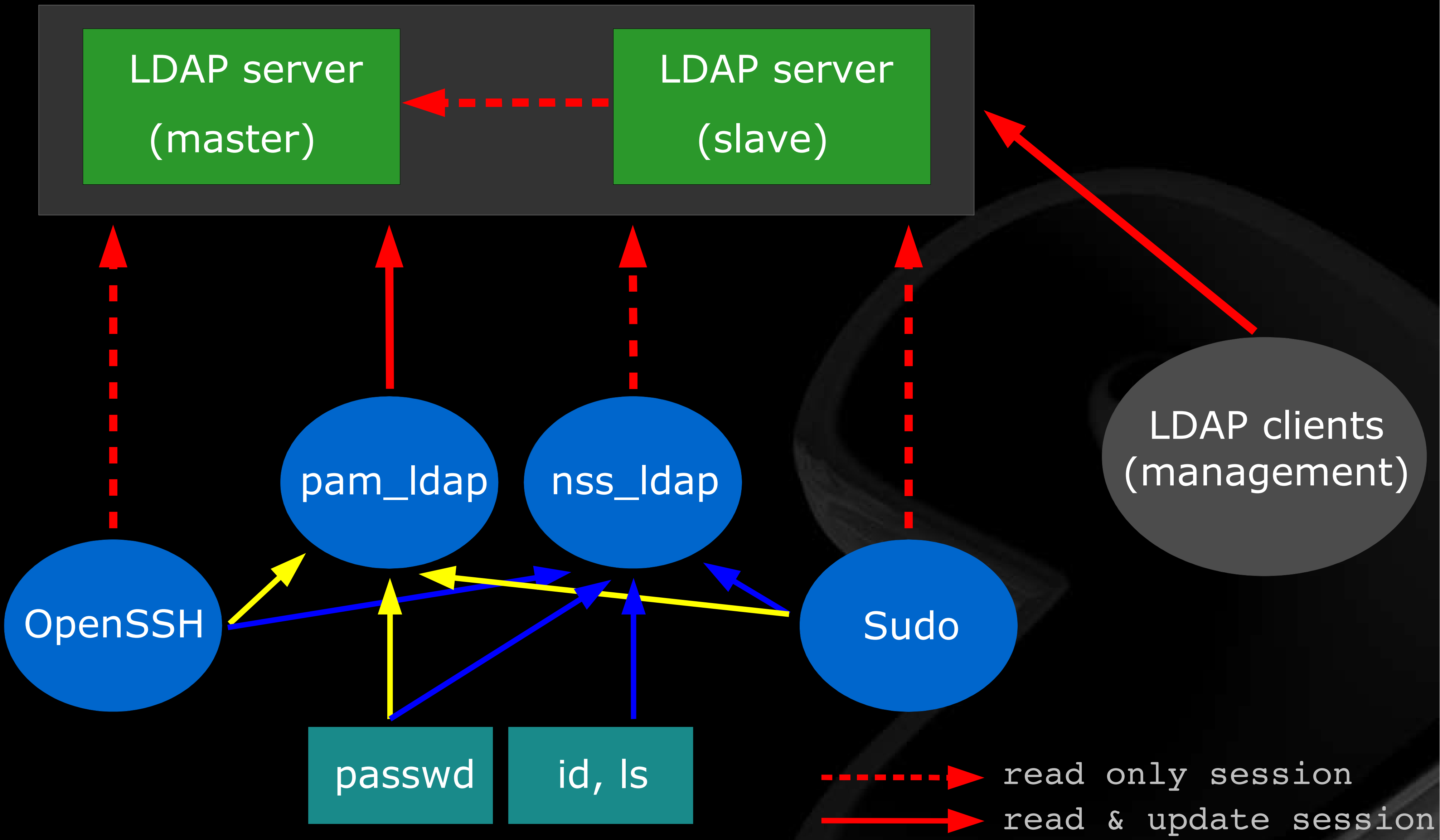
LDAP complexity: why do most admins find it hard to implement properly?

- defining documentation as “messy” is an understatement:
  - lack of proper documentation; users, developers and security wise
  - existing documentation is sometimes incomplete and/or incorrect
  - usually key features can only be enabled with undocumented features (unless you consider source code a form of documentation)
- many different components/software layers are involved in a complete framework, this renders debugging more difficult than expected
- the “background noise” of common questions/errors on mailing lists and forums related to LDAP software is considerable
- generally the awareness about *what* could be done with LDAP and *how* is scarce, this sometimes leads to insecure implementations

LDAP complexity: then why are we supposed to use it?

- it's not *that* bad (consider that I probably felt the need to scare you more than necessary), once it works it's a robust enterprise-grade framework
- despite all flaws related to the documentation it's a reliable and secure system for central account management (and more)
- a properly implemented LDAP framework can effectively increase the security of users management (*especially* when migrating from NIS/YP)
- highly scalable
- access restriction (acl, ip-based, socket-based), TLS, utf-8 support, custom database backends, replication
- awareness is growing (openssh-lpk and sudo being examples), expect many more LDAP'ized apps and better documentation in the future

The framework



Software versions

- openldap-2.2.28
- nss_ldap-239 / pam_ldap-180
- openssh-4.1p1 | openssh-4.0p1 | openssh-3.9p1 + openssh-lpk patch

  http://www.opendarwin.org/en/projects/openssh-lpk
- sudo-1.6.8p9
- many outstanding bugs have been fixed in gentoo versions because of our implementation, these problems will affect any serious production environment:
  - referral chase security fix (GLSA 200507-13 | CAN-2005-2069)
  - working failover in sudo

    http://dev.gentoo.org/~lcars/misc/sudo-ldap_timelimit.diff
  - many fixes in openssh-lpk (including proper failover), I became co-maintainer of the project in the process

Security patch: CAN-2005-2069

- affects TLS (LDAP over SSL considered deprecated, TLS becoming default choice)
- on a master + slave setup writes are handled by the the master but clients sometimes connect to a slave first and are "referred" to a new URI (for the master) when trying to update entries
- should TLS be started on the referred connection ? (no way to tell from the URI)
- pam_ldap/nss_ldap made no effort to do so, but then it's not mentioned in OpenLDAP documentation that they should (bug 1)
- OpenLDAP wouldn't have allowed it anyway, a bug prevents you from starting TLS on anything other than the initial connection (bug 2)
- cross application bug
- the result is password being sent in the clear when chasing referrals

Security patch: CAN-2005-2069

- why didn't anyone notice?
- current status (October 1st 2005):
  - most vendors shipped fixes for pam_ldap and OpenLDAP fairly quickly (although many haven't fixed nss_ldap)
  - pam_ldap/nss_ldap maintainer took nearly a month to fix it (patch was already available along with the first warning)
  - OpenLDAP still haven't released a fix (although it is in CVS now)
  - the bug was ignored for more than a month and a half
  - why didn't upstream care?

http://dev.gentoo.org/~lcars/ldap/nss_ldap-239-tls-security-bug.patch

http://dev.gentoo.org/~lcars/ldap/pam_ldap-176-fix-referral-tls.patch

http://dev.gentoo.org/~lcars/ldap/openldap-2.2.26-tls-fix-connection-test.patch

OpenLDAP configuration: extending the schema

- obtain a unique *Object Identifier* (OID): 1.3.6.1.4.1.2242.1.1.1
  - attributes: 1.3.6.1.4.1.2242.1.1.*
  - objectclasses: 1.3.6.1.4.1.2242.1.2.*

```
attributetype ( 1.3.6.1.4.1.22242.1.1.4
 NAME 'accessLevel'
 DESC 'user access level'
 EQUALITY caseIgnoreMatch
 SUBSTR caseIgnoreSubstringsMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

custom.schema

- use attributes of existing schemas if possible

```
objectclass ( 1.3.6.1.4.1.22242.1.2.1
 NAME 'pacsecUser'
 DESC 'pacsec user'
 AUXILIARY
 MUST ( accessLevel $ ... )
 MAY  ( gpgkey $ gpgfingerprint $ ... )
```

http://www.alvestrand.no/objectid/1.3.6.1.4.1.1466.115.121.1.html

http://www.iana.org/assignments/enterprise-numbers

OpenLDAP, slapd configuration

- /etc/openldap/slapd.conf is the configuration file for the OpenLDAP daemon process serving LDAP requests (slapd)
- we have to include the additional schemas we are going to use

```
include /etc/openldap/schema/core.schema
include /etc/openldap/schema/cosine.schema
include /etc/openldap/schema/inetorgperson.schema
include /etc/openldap/schema/nis.schema
include /etc/openldap/schema/custom.schema
include /etc/openldap/schema/sudo.schema
include /etc/openldap/schema/openssh-lpk.schema
```

- most likely verbose logging will be useful for initial testing, when in production it should be disabled or set to a reasonable value

```
loglevel 256
#loglevel 0
```

OpenLDAP, slapd configuration: Transport Layer Security (TLS)

- we don't allow unencrypted traffic, all connections are protected by TLS

```
security tls=1
```

- we need certificate files (readable only by root and/or slapd user) signed by a Certification Authority (CA)

```
TLSCertificateFile     /etc/openldap/ssl/cert.pem
TLSCertificateKeyFile /etc/openldap/ssl/req.pem
TLSCACertificateFile  /etc/openldap/ssl/ca.pem
```

- in addition to transport protection we also want to authenticate our clients since we don't ever want to rely on the network layer for authentication

```
TLSVerifyClient demand
```

- we can allow password hashing server side, this is used for the so called Password Modify Extended Operations, it needs to be enabled client side

```
password-hash {MD5}
```

OpenLDAP, slapd configuration: Access Lists (ACL)

- ACL syntax is far from being perfect and it could be confusing at first but it's very powerful and reasonably flexible, let's look at some examples

- protected world-readable attribute, granting selective write access:

```
access to dn.subtree="ou=users,dc=pacsec,dc=jp" attrs="accessLevel"
 by dn.subtree="ou=admin,ou=users,dc=pacsec,dc=jp" \
    peername.regex="10\.1\.7\.1" write
 by dn.base="uid=ldap_admin,ou=infra,dc=pacsec,dc=jp" \
    sockurl.exact="ldapi://%2var%2run%2openldap%2slapd.sock" write
 by * read
```

- protected attribute, readable only by authenticated users

```
access to dn.subtree"dc=pacsec,dc=jp" attrs="userPassword"
 by dn.subtree="ou=admin,ou=users,dc=pacsec,dc=jp" \
    peername.regex="10\.1\.7\.1" write
 by dn.base="cn=syncrepl,dc=pacsec,dc=jp" \
    peername.regex="10\.1\.7\.2" read
 by self write
 by anonymous auth
```

OpenLDAP, slapd configuration: Access Lists (ACL)

- protected world-readable attribute, granting write access to users:

```
access to dn.subtree=”dc=pacsec,dc=jp” attrs=”sshPublicKey,gpgkey”
 by dn.subtree=”ou=admin,ou=users,dc=pacsec,dc=jp” \
    peername.regex=”10\.1\.7\.1” write
 by self write
 by * read
```

- policy for everything else, put after all other acl entries (order matters!)

```
access to *
 by dn.subtree=”ou=admin,ou=users,dc=pacsec,dc=jp” \
    peername.regex=”10\.1\.7\.1” write
 by * read
```

- we can allow selective access to entries and attributes to specific users or groups of users restricting based on IP address (strenghtened by underlying hostname matching TLS cert) or socket name (requires access to a specific box + matching filesystem permissions)

OpenLDAP, slapd configuration: backend db

- we can choose different backends for the db, each one has different data structures and options, the default choice is bdb (currently a required choice for replication with syncrepl)

```
database    bdb
suffix      “dc=pacsec,dc=jp”
directory   /var/lib/openldap-data
sessionlog 100 500
index       objectClass,uid,uidNumber,gidNumber,accessLevel pres,eq
index       entryUUID pres,eq
cachesize   10000
sizelimit   1000
```

- rootdn/rootpw can be used *temporarily* for initial db creation, it must be removed when deploying since it bypasses all acl

```
rootdn      “cn=Manager,dc=pacsec,dc=jp”
rootpw      <password>
```

OpenLDAP, slapd configuration: slave server

- slurpd was considered the standard choice but it's a push based system (master updates the slaves), it's not scalable and it doesn't handle network problems very well
- syncrepl provides a better alternative, it's pull based (slaves fetch updates from the master) and it has better connection control
- we'll use a “dummy” rootdn without password for making syncrepl write to the slave db, connections trying to perform write operations on the slave will be referred to the master

```
updateref ldap://ldap1.pacsec.jp:389

database bdb
rootdn “cn=Replication,dc=pacsec,dc=jp”
...
```

OpenLDAP, slapd configuration: slave server

- rid matches master slapd.conf sessionlog id
- the essential (but undocumented in OpenLDAP 2.2) retry feature specifies reconnection times (60 seconds the first 10 times, 300 seconds for following connections, + means undefinetly)

```
syncrepl rid=100
    provider=ldap://ldap1.pacsec.jp:389
    type=refreshOnly
    interval=00:00:00:60
    retry="60 10 300 +"
    timelimit=10
    searchbase="dc=pacsec,dc=jp"
    updatedn="cn=Replication,dc=pacsec,dc=jp
    binddn="cn=syncrepl,dc=pacsec,dc=jp"
    bindmethod=simple
    credentials=<password>
    startssl=critical
```

- grant access on master acls

```
by dn.base="cn=syncrepl,dc=pacsec,dc=jp" \
    peername.regex="10\.1\.7\.2" read
```

OpenLDAP client library configuration and directory access

- /etc/openldap/ldap.conf is the configuration for the client library

```
BASE          dc=pacsec, dc=jp
URI           ldap://ldap1.pacsec.jp ldap://ldap2.pacsec.jp
TLS_REQCERT   demand
TLS_CACERT    /etc/openldap/ssl/ca.pem
TIMELIMIT     5
```

- we require server validation, again we don't rely on the network layer
- we need client certificates specification for root in /root/.ldaprc (we'll discuss later certs for other users), they should not be world readable

```
TLS_CERT   /etc/openldap/ssl/cert.pem
TLS_KEY    /etc/openldap/ssl/req.pem
```

- initial directory can be configured with slapadd|slapmodify (direct access to the backend) or ldapadd|ldapmodify

LDAP initialization: organizational units (OU)

- init.ldif

```
dn: dc=pacsec,dc=jp
objectClass: organization
objectClass: dcObject
o: pacsec.jp
dc: pacsec

dn: ou=users,dc=pacsec,dc=jp
objectClass: organizationalUnit
ou: devs

dn: ou=groups,dc=pacsec,dc=jp
objectClass: organizationalUnit
ou: groups

dn: ou=SUDOers,dc=pacsec,dc=jp
objectClass: organizationalUnit
ou: SUDOers

dn: ou=admin,ou=users,dc=pacsec,dc=jp
objectClass: organizationalUnit
ou: infra
```

```
dn: cn=syncrepl,dc=pacsec,dc=jp
objectClass: organizationalRole
objectClass: simpleSecurityObject
cn: syncrepl
userPassword: {SSHA}s83JkijBCAEE3409...
structuralObjectClass: organizationalRole
```

- we initialize our directory tree by creating the needed organizational units and syncrepl dn entry

```
slapadd -p -w -l init.ldif
```

```
ldapadd -Z -W \
 -D "cn=Manager,dc=pacsec,dc=jp" \
 -f init.ldif
```

LDAP initialization: user entry

```
dn: uid=lcars,ou=admin,ou=users,dc=pacsec,dc=jp
cn: Andrea Barisani
givenName: Andrea
sn: Barisani
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: pacsecUser
objectClass: ldapPublicKey
userPassword: {crypt}$1$2f93D3A30fBCAEE34r3rf
loginShell: /bin/bash
gidNumber: 100
uidNumber: 660
uid: lcars
gecos: Andrea Barisani,,,
gpgkey: 0x864C9B9E
gpgfingerprint: 0A76 074A 02CD E989 CE7F AC3F DA47 578E 864C 9B9E
description: developer, Trieste - Italy
telephoneNumber: (555) 593 342 430
accessLevel: server1.pacsec.jp
accessLevel: server2.pacsec.jp
sshPublicKey: ssh-dss AAAAB3NZdjoie293t4tjfdklofj997438o9t5ru43ioyf8439Dr333...
```

nss_ldap / pam_ldap configuration

- /etc/ldap.conf is the shared file for nss_ldap and pam_ldap configuration, it must be world readable, /etc/nsswitch.conf specifies nss db search order
- /etc/ldap.conf file is completely unrelated to /etc/openldap/ldap.conf
- typos and errors are not going to raise an error because syntax checking is considered too expensive
- nss_ldap is a C library extension used by the Name Service Switch code, it provides transparent access to the LDAP directory for standard C library functions related to users and groups (getpwent, getgrent, ...)
- pam_ldap is the PAM module linked by applications that need authentication against the LDAP directory
- we filter against an accessLevel attribute for selectively enabling/disabling a user client-side preventing user enumeration in case of client compromise, some fine grained acls on slapd are necessary

nss_ldap / pam_ldap configuration

- /etc/ldap.conf

```
ldap_version          3
scope                 sub
timelimit             3
bind_timelimit        3
bind_policy           hard
idle_timelimit        3600
pam_login_attribute   uid
pam_member_attribute  gid
pam_password          md5
#pam_password         exop
pam_filter            accessLevel=server1.pacsec.jp
uri                   ldap://ldap1.pacsec.jp ldap://ldap2.pacsec.jp
suffix                “dc=pacsec,dc=jp”
base                  ou=users,dc=pacsec,dc=jp?sub
nss_base_passwd       ou=users,dc=pacsec,dc=jp?sub?accessLevel=server1.pacsec.jp
nss_base_shadow       ou=users,dc=pacsec,dc=jp?sub?accessLevel=server1.pacsec.jp
nss_base_group        ou=users,dc=pacsec,dc=jp?sub?accessLevel=server1.pacsec.jp
ssl                   start_tls
tls_checkpeer         yes
tls_cacertfile        /etc/openldap/ssl/ca.pem
tls_cert              /etc/openldap/ssl/cert.pem
tls_key               /etc/openldap/ssl/req.pem
```

- again we enforce tls and cert validation
- we filter against accessLevel client-side
- we specify client certificates for root
- `pam_password exop` enables slapd `password-hash`

Pluggable Authentication Module configuration

- /etc/pam.d/system-auth

```
auth      required   pam_env.so
auth      sufficient pam_ldap.so
auth      sufficient pam_unix.so likeauth nullok nodelay use_first_pass
auth      required   pam_deny.so

account   sufficient pam_ldap.so
account   required   pam_unix.so

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

session   required   pam_limits.so
session   required   pam_unix.so
session   optional   pam_ldap.so
```

- /etc/pam.d/sshd (we create the home directory automatically if missing)

```
...
session required pam_mkhomedir.so skel=/etc/skel/ umask=0077
```

OpenSSH configuration

- OpenSSH with LPK patch (LdapPublicKey) looks up the sshPublicKey attribute (it can hold multiple values for multiple keys) and uses it as authorized_keys file (no need to manually create/copy the file)
- the physical authorized_keys file will still be used if no matching entry is found
- latest openssh-lpk patch is able to parse /etc/ldap.conf for its configuration (supported settings: uri, base, timelimit, bind_timelimit, ssl, start_tls)
- the information is public so the attribute can be world readable and conveniently modifiable by the user
- /etc/ssh/sshd_config

```
...

UsePAM yes
UseLPK yes
LpkLdapConf /etc/ldap.conf
```

Sudo configuration

- Sudo can look up sudoers settings in the LDAP directory
- no physical files to manage, we can manage and query sudo profiles centrally
- the physical sudoers file will still be used if no matching entry is found (can be overriden with the ignore_local_sudoers attribute but that's not recommended, it's safe to keep a physical failsafe entry)
- it's recommended to enable a separate configuration file at compile time (`--with-ldap-conf-file=/etc/ldap.conf.sudo`) in order to restrict sudo attributes visibility to superuser only (like standard /etc/sudoers permissions)
- we create a separate LDAP profile with authentication for accessing sudo entries

Sudo configuration

- we create a new ou for sudo entries and a new user for protecting the ou via acls

```
access to dn.subtree="ou=sudoers,dc=pacsec,dc=jp"
 by dn.base="cn=sudoers,dc=pacsec,dc=jp" read
 by dn.subtree="ou=admin,ou=users,dc=pacsec,dc=jp" \
    peername.regex="10\.1\.7\.1" write
 by dn.base="cn=syncrepl,dc=pacsec,dc=jp" \
    peername.regex="10\.1\.7\.2" read
```

```
dn: ou=sudoers,dc=pacsec,dc=jp
objectClass: organizationalUnit
ou: SUDOers
```

```
dn: cn=sudoers,dc=pacsec,dc=jp
objectClass: organizationalRole
objectClass: simpleSecurityObject
cn: sudoers
userPassword: {SSHA}i38fdaf8923prfWE...
structuralObjectClass: organizationalRole
```

```
dn:cn=admin,ou=SUDOers,dc=pacsec,dc=jp
cn: admin
objectClass: top
objectClass: sudoRole
sudoUser: lcars
sudoHost: cvs.pacsec.jp
sudoCommand: ALL
sudoOption: authenticate
```

```
dn:cn=mail,ou=SUDOers,dc=pacsec,dc=jp
cn: mail
objectClass: top
objectClass: sudoRole
sudoUser: foo
sudoRunAs: mail
sudoHost: mail.pacsec.jp
sudoCommand: /usr/bin/newaliases
sudoOption: !authenticate
```

Sudo configuration

- /etc/ldap.conf.sudo

```
ldap_version          3
timelimit             3
bind_timelimit        3
uri                   ldap://ldap1.pacsec.jp ldap://ldap2.pacsec.jp
ssl                   start_tls
tls_checkpeer         yes
tls_cacertfile        /etc/openldap/ssl/ca.pem
tls_cert              /etc/openldap/ssl/cert.pem
tls_key               /etc/openldap/ssl/req.pem
binddn                cn=sudoers,dc=pacsec,dc=jp
bindpw                <password>
sudoers_base          ou=SUDOers,dc=pacsec,dc=jp
sudoers_debug         2
```

- sudo attributes are only visible when binding with binddn, bindpw
- /etc/ldap.conf.sudo should match /etc/sudoers permissions, not world readable (unlike /etc/ldap.conf)

Name Service Cache Daemon (nscd) and client certificates

- every time the system performs getpwnam(3), getpwuid(3) and similiar libc functions nss_ldap queries the LDAP server, on busy servers this could affect performance considerably
- nscd provides a cache for such requests
- positive and negative queries are cached (TTL can be set in /etc/nscd.conf)
- when using nscd, expect delays in new account lookup and when enabling/disabling users (nscd can be evil if you forget about its presence)
- `/etc/init.d/nscd restart` or nscd `--invalidate` are useful
- authentication data is *not* cached
- with nscd we can avoid issuing per user certificates and keep only certificates for root and pam aware apps, users are not going to execute nss_ldap code directly since nscd acts as a transparent proxy cache

Failover scenario

- always use the closest slapd server as the first one in uri specification
- if network connectivity is down all servers will be tried sequentially, it's unadvisable to have more than 2 slave servers
- sshd LoginGraceTime should be set accordingly (at least 120 seconds when using 3 LDAP servers) to prevent login phase timeout in case all LDAP servers are not reachable
- 3 seconds is a reasonable timelimit/bind_timelimit setting considering that sudo and openssh make at least 2 LDAP bindings each time
- the worst case scenario is total LDAP server loss without any TCP/IP and/or ICMP rejection from the network (like dumb sysadmin messing with local firewall configuration, so it's not that unlikely to happen)
- wheel accounts should be kept both in LDAP and locally in case of LDAP problems

The end

Questions?

:-)