Computer System Design and Administration

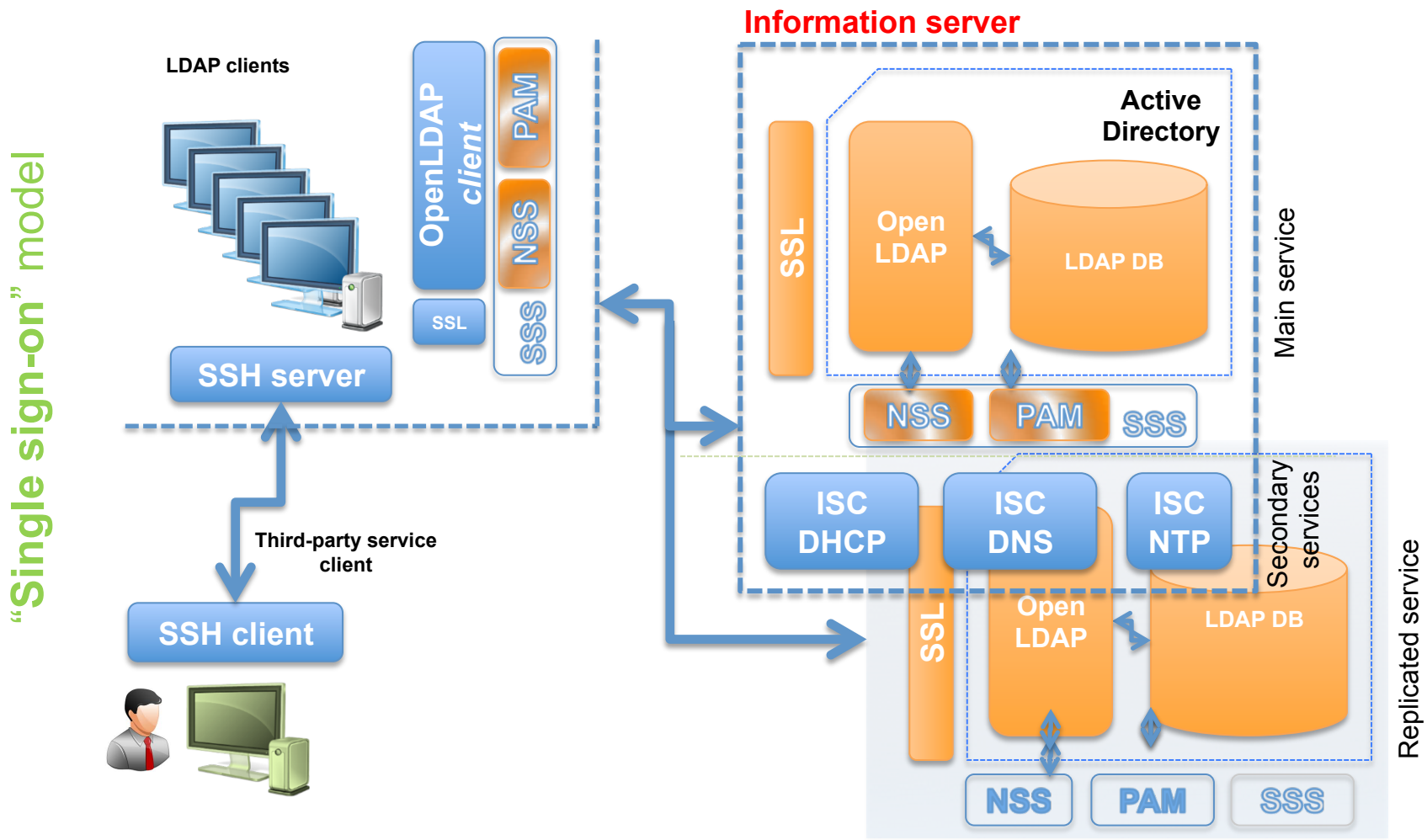
Topic 3. Active directory integration mechanisms: PAM + NSS (SSSd)

José Ángel Herrero Velasco

Department of Computer and
Electrical Engineering

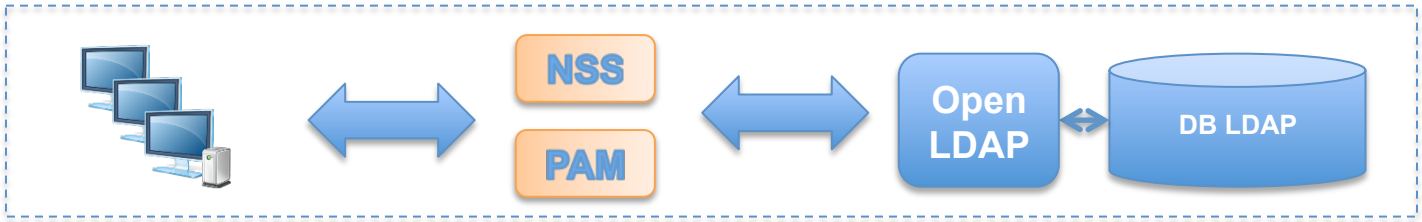
This work is published under a License:
Creative Commons BY-NC-SA 4.0

Secure information service: Puzzle



Target: The Single Sing-On mechanisms

- Implementation and development of a secure and **centralized** system for the management of ***account and computational information*** in an enterprise (corporative) environment, using **LDAP** protocol.
  - **Single Sing-On → INTEGRATION mechanisms:**
    - **LDAP active directory ←→**



    - Centralized **identification** and **authentication** aware versions of C-library routines
      - **Identity → NSS.**
      - **Authenticate→ PAM.**

→ SSSd (all in one).

PAM: Definition & features

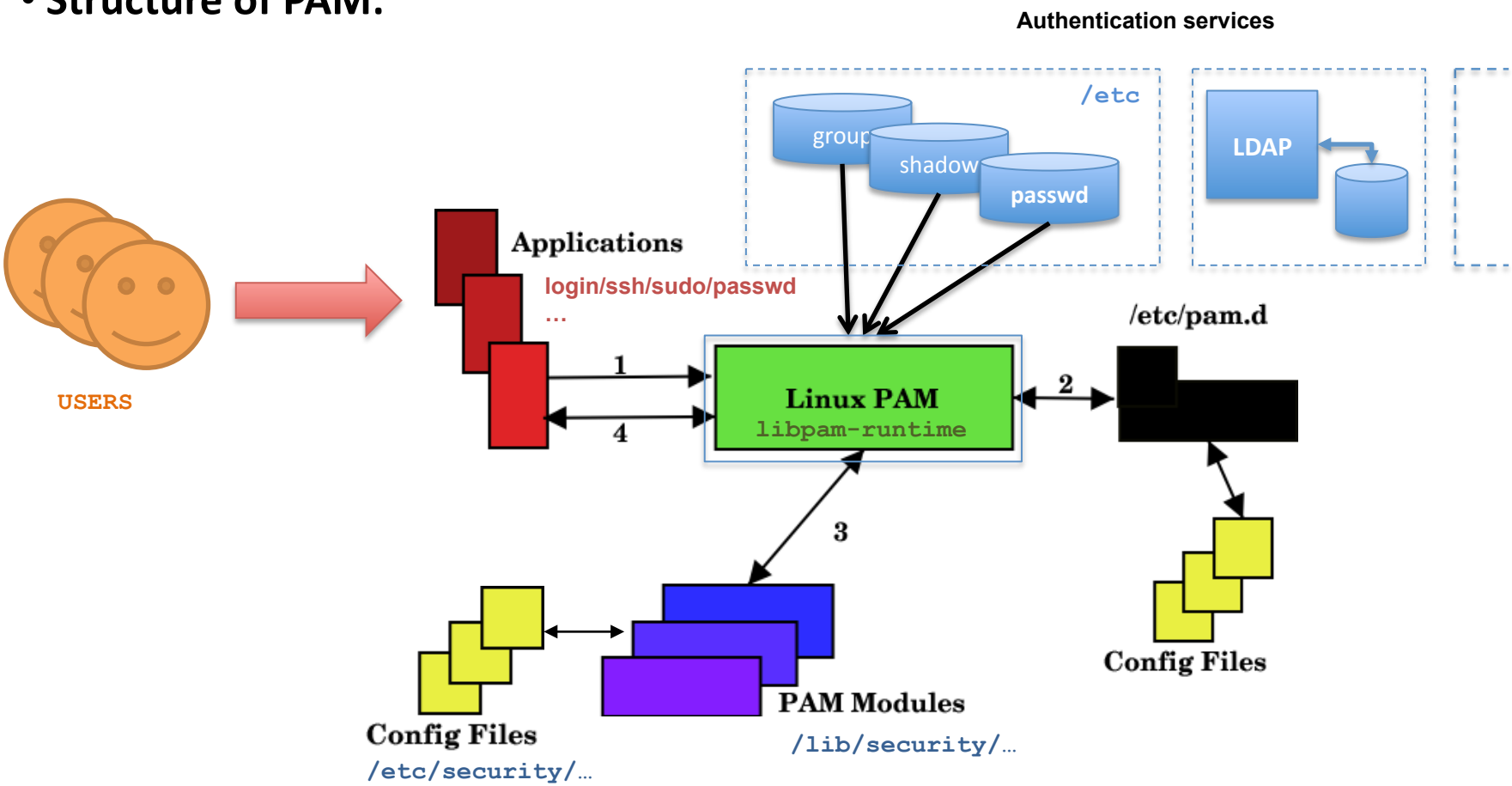
- **P.A.M:** ***P****luggable* ***A****uthentication* ***M****odules:*
  - Initially developed by Sun Microsystems (Oracle) from 1995 (RFC 86.0).
- **Definition:**
  - It's a **suite of shared libraries** that:
    - Enables **integrating** *multiple* ***authentication*** *schemas* into an API which is used by applications:
      - **PAM-aware applications, through this API, are able to select the authentication mechanism(s) that it uses:**
        - » Currently, by default.
    - Sysadmins may entirely **change** the *authentication system* without "*altering*" the applications.
    - Programmers relieve themselves of the chore of implementing authentication systems for their software.
- **Main features:**
  - It makes it easy to integrate new advances in authentication mechanisms:
    - `{/etc/passwd,/etc/shadow,/etc/group}` (*default*).
    - `LDAP authentication module.`
    - `Kerberos.`
    - Biometric system.
    - One-time passwords.
  - It not only enables:
    - **Checking** that a user is ok (1):
      - **… He is who he says he is.**
    - **Validating** a user for using a specific object from the system (2).
  - But also:
    - It adds a mark-up:
      - **Management of authentication data.**
      - **Execute *pre* and *post* authentication tasks.**
      - **Sessions management.**
      - **…**

PAM: Definition & features

- **All in aall:**
  - It allows:
    - The applications to **become independent** from authentication mechanisms:
      - **PAM is located between applications/services and authentication mechanisms (wrapper):**
        - » In the past, applications had to include these mechanisms themselves *(binary).*
    - Defining **security policies** and customizing them for each application/service:
      - **It can establish default policies.**
      - **It can provide a wide range of authentication mechanisms.**
  - It provides:
    - A common scheme (API) for authentication tasks:
      - **Easing the app programmer work.**
  - It has:
    - A **MODULAR** architecture:
      - **PAM Run-time (core): libpam-runtime.**
      - **Set of dynamic load modules or libraries: `pam_*.so`.**
      - **+ module configuration files (sometimes).**
      - **+ PAM configuration files about applications/services.**

PAM: Architecture

- **Structure of PAM:**



PAM: Operation scheme

- Every **PAM-aware application/service** is configured in a particular file:
  - `/etc/pam.d/`
  - **Where we set up...**
    - **Rules** about authentication process:
      - **Rule groups.**
    - Required **modules** (PAM libraries).
  - The applications/services will remain oblivious to these "halfway steps".
- The applications/services should support PAM (be PAM-aware):
  - These should be implemented around PAM capabilities.
  - When an app source code is compiled, it should be dynamically linked with PAM `libpam.so` library:
    - Check (instance).
    - `$ ldd /usr/sbin/sshd`
- **Operation:**
  - PAM configuration files define a series of **rules** (one per line):
    - Each of which names (runs) a particular PAM module.

`Module-type control-flag module arguments` `(module)`

PAM: Installation

- Actually all UNIX/Linux systems support PAM:
  - In every Linux *distro* (general-purpose), **PAM is integrated into the system:**
    - It is installed when installing the O.S.
    - It doesn't need to install anything!!!
- PAM is composed of a "**core**" + **modules** or dynamic libraries:
  - The PAM "**core**" is implemented in **libpam-runtime** package:
    - `$ dpkg -L libpam-runtime`
    - It provides:
      - **The basic functionality of PAM.**
      - **Documentation manuals (`$man`).**
      - **Initial directory tree.**
  - The rest of the **modules/libraries** will be installed on demand:
    - `$ apt-get install` **`libpam`**`-xxx`
    - They provide the **required functionality** to:
      - **Add new connectors for different authentication mechanisms.**
      - **Authentication items (data) management.**
      - **Session management.**
      - **...**

Debian

PAM: Configuration

- 2 configuration models in PAM:
  - ***Initial* model (UNIX):**
    - Based on a single file: ← … deprecated!!!
      - `/etc/pam.conf`
  - ***Current* model (Linux):**
    - **Modular configuration** (many configuration files)
      → in **`/etc/pam.d/`** directory:
      - **One PAM configuration file for each application/service:**
        - » Instance: `/etc/pam.d/`**`sshd`**.
      - **4 PAM common configuration files *(by default):***
        - → *They are used by each and every application/service* ***without specific configuration file.***
        - → *They have a* ***generic scope:***
          - »`/etc/pam.d/common-`**`auth`**.
          - »`/etc/pam.d/common-`**`account`**.
          - »`/etc/pam.d/common-`**`sessión`**.
          - »`/etc/pam.d/common-`**`password`**.

PAM: Configuration

- In every application/service PAM configuration file, a couple of **rules** are defined (one per line):
  - **Grouped** by each type of used module:
    - Set of rules for **auth.**
    - Set of rules for **account.**
    - Set of rules for **session.**
    - Set of rules for **password.**

Instance:

```
auth     required    pam_securetty.so
auth     required    pam_nologin.so
auth     required    pam_pwdb.so shadow md5

account  required    pam_pwdb.so

session  required    pam_pwdb.so
session  optional    pam_lastlog.so

password required    pam_cracklib.so retry=3
password required    pam_pwdb.so shadow md5
```

  - The order **of the rules** in this file is very **IMPORTANT:**
    - Should be considered !!!
    - Descending order.
  - Format:

```
Module-type   control-flag   module arguments  (module)
```

PAM: Configuration

- **Module-type:**
  → It shows **the module type** used by the rule:

module-type	control-flag	module	arguments (module)
auth	required	pam_ldap.so	minimum_uid=900

  - **Auth:** authentication and credentials:
    - Sets up the **authentication** mechanism for a particular application:
      - **Local system (`passwd/shadow/group`).**
      - **LDAP.**
      - **Kerberos…**
    - It grants to the user the required *credentials* for carrying out any operation.
  - **Account:** user account validation:
    - Sets up the **validation** tasks about users' rights in the system:
      - **Can the user access/execute the requested resource (service/app)?:**
        - » Uses according to date, resource availability, location.
    - Account expiration…
  - **Session:** session management (start/end session):
    - Sets up the tasks which are done at the **beginning/end** of sessions:
      - **Creation home directories…**
      - **Operation run at the beginning…**
      - ***Prolog* and *epilog*…**
  - **Password:** update authentication items (passwords):
    - Sets up the tasks to keep **authentication items** correctly updated:
      - Passwords & credentials.

PAM: Configuration

- **Control-flag:**
  - → It shows **the operation that PAM** should do if the module run successfully or not:

module-type	control-flag	module	arguments (module)
auth	**required**	pam_ldap.so	minimum_uid=900

  - **Simplified** syntax: 4 possible values (action: notification only):
    - *Requisite* → Must be verified:
      - **If module fails, PAM immediately notifies a failure to the app and STOP!**
      - **If it doesn't, PAM will run the rest of rules (descending order).**
    - *Required* → Must be verified:
      - **If module fails, PAM notifies a failure once the remaining rules are completed.**
      - **If it doesn't, PAM will run the rest of rules (descending order).**
    - *Sufficient* → Can be verified or not:
      - **If module fails, PAM will ignore it.**
      - **If it doesn't, PAM will immediately notify "OK" without trying any other modules:**
        - » Unless there is a previous failure *(required)*.
    - *Optional* → Can be verified or not:
      - **Its result is considered only if it is the only rule in the stack.**
  - **Extended** syntax:
    - [value=operation, value=operation…]:
      - **Value: it's the value returned by the module when it's run.**
      - **Operation: action carried out depending on the value...**

Returned values:
`success,`
`abort,`
`authok_err,`
`default,`
`user_unknown …`

Operations:
`n° entero k,`
`ingone,`
`ok,`
`done,`
`reset, die …`

PAM: Configuration

- **Module:**

  `module-type  control-flag  module  arguments` (module)

  - Name of the **dynamic load library** which is ran by the rule:
    - It contains the rule functionality:
      - **For instance:**
        - » Use of a particular authentication method by the application.
    - On Debian Linux, **relative `PATH`** to the PAM library:

    → **`/lib/security/pam_*.so`**…

- **Arguments:**
  - **Parameters** which are passed to the module:
    - `auth [success=3 default=ignore] pam_ldap.so` **`minimum_uid=900`**.
  - If an "*invalid*" argument is passed, it's ignored:
    - PAM records an error event in *syslog.*

PAM: Modules

- They are **dynamic load libraries** which are executed for each rule:
  - Located on `/lib/security/…` `(/lib/x86_64-linux-gnu/security/…)`.
- In same cases, there may be a **configuration file** *associated with* the module:
  - Located on `/etc/security/…`
- Instances:
  - **`pam_unix.so`:**
    - **Type:** *auth, session, account, password*
    - Implements integration of the **classic UNIX authentication mechanism:**
      - → `passwd/shadow`.
  - **`pam_ldap.so`:**
    - **Type**: *auth, session, account, password.*
    - Implements integration of the **LDAP authentication mechanism** (openLDAP).
    - This module can be configured in `/etc/pam_ldap.conf`.
  - **`pam_krb5.so`:**
    - **Type:** *auth, session, account, password.*
    - Implements integration of the **Kerberos5 authentication mechanism.**

Linux x86_64

PAM: Modules

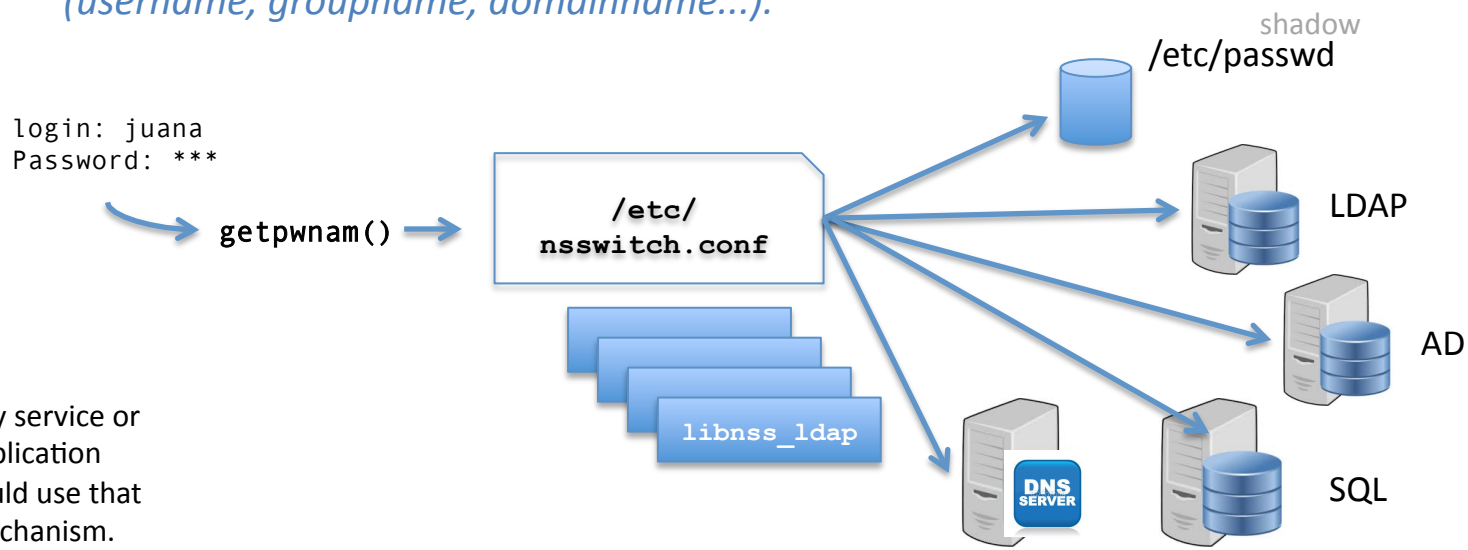
- `pam_cracklib.so:`
  - **Type:** *password.*
  - It's used when the user **changes** its password:
    - **It checks the strength of the passwords.**
    - **It can use dictionaries:**
      - » `/etc/lib/cracklib_dict.`
- `pam_env.so:`
  - **Type:** *auth.*
  - It enables setting **environment variables.**
- `pam_limits.so:`
  - **Type:** *session.*
  - It enables setting **"shell" limits** for using any system resources:
    - **Session time, "*core*" file sizes, cpu time, number of cpus…**
  - This module can be configured in `/etc/security/limits.conf.`
- `pam_mkhomedir.so:`
  - **Type:** *session.*
  - When a session starts, it creates the **user home directory**, if it doesn't exist.

PAM: Modules

- `pam_access.so`:
  - **Type:** *account.*
  - It checks if user **can open a session** from a remote host (IP).
  - This module can be configured in /`etc/security/access`.
- `pam_deny.so`:
  - **Type:** *auth, session, account, password.*
  - It always returns a **failure:**
    - **It's useful for ordering the *stack* of rules.**
- `pam_permit.so`:
  - **Type:** *auth, session, account, password.*
  - It always returns an **"ok":**
    - **It's useful for ordering the *stack* of rules too.**

NSS: Definition & features

- **N.S.S:** ***N**ame **S**ervice **S**witch.*
- Complementary mechanism to PAM:
  - Is part of the OS (default): **"C library".**
- **NSS** establishes what **identification method** will be used by the users (and other system entities) in the system[1]:
  - It unifies the **translation/mapping** between numeric IDs *(uid, gid, ip...)* and names *(username, groupname, domainname...).*



[1] Any service or application could use that mechanism.

NSS: Name Service Switch

- Configuration files:

  `/etc/nsswitch.conf:`

  – Define the query **order** (From left to right) among the possible identification methods by **GNU C library:**

Types of lookup

```
passwd:         compat ldap sss
group:          compat ldap sss
shadow:         compat ldap sss

hosts:          files dns
networks:       files

protocols:      db files
services:       db files
ethers:         db files
rpc:            db files

netgroup:       nis
```

Sources list

Compatibility mode (compat)

/etc/passwd
shadow
group

Checking:

```
$ id <username>
$ getent passwd
…
$ ping <hostname>
```

NSS: Name Service Cache

- ***NSCD:** Name Service Cache Daemon.*
- This service is often used **to cache entity information** *(users, groups, hosts...):*
  - It is able to speed up **identification queries.**
- When a client or system requests this kind of information, the **NSCD** service **caches** it to solve the *following* requests more efficiently:
  - It maintains two different caches:
    - *Hits:* results found.
    - *Failures:* results not found.
- **Be careful!!!:**
  - Theses caches may not be updated:
    - In this case, restart the service:

    → `/etc/init.d/nscd restart.`
- Linux distributions don't usually install NSCD by default:
  - **Debian: `nscd`** is not installed by default.
  - **Red Hat: `nscd`** is installed by default, but it is unavailable.
  - **SUSE: `nscd`** is installed by default and it is available on boot.

NSS + LDAP: User/group identification

- To enable **LDAP user/group identification** in the system, integration between **NSS** and LDAP is necessary:
  - Sysadmin should install the `libnss-ldap` package:
    - NSS-LDAP **connector.**
    - Suite of **libraries** and **tools.**
  - So it allows us:
    - **Identification of users**, **groups**, hosts, services…
    - **Mapping** between numeric IDs *(**UID**, gid, ip...)* and names (***USERNAME**, groupname, domainname...)*.
      - `$ id <username>.`
      - `$ getent shadow.`

- **`libnss-ldap`** installation:

```
$ apt-get install libnss-ldap.
```

NSS + LDAP: User/group identification

- **NSS configuration** for LDAP (most important configuration items):
  `$ vi /etc/libnss-ldap.conf.`
- `host`: LDAP server hostname/IP:
  - (*) If you use SSL (slapd**s**), it's very important that this is the "`common name`" used in the SSL certificate.
- `base`: "distinguished name" (DN) or LDAP *suffix (root)* for corporative DIT.
- `port`: LDAP port.
- `URI` (**U**niform **R**esource **I**dentifier): it establishes the LDAP server network ID:
  - `uri ldap://server-01.localdomain.`
- `rootbinddn`: it sets the administrator "DN" for the LDAP DB:
  - `rootbinddn: cn=admin,dc=localdomain.`
- SSL/TLS options:
  - **`TLS_CACERT`: PATH to CA certificate.**
  - `TLS_CERT`: PATH to service certificate.
  - `TLS_KEY`: PATH to service certificate key → TLS_CERT.
  - `TLS_REQCERT`: it sets the mechanism to use for clenching the certificate:
    - Never, try, **demand.**

PAM + LDAP: User authentication

- To enable **LDAP authentication** in the system, integration between **PAM** and LDAP is necessary:
  - Sysadmin should install the `libpam-ldap` package:
    - PAM-LDAP **connector.**
    - Suite of **libraries** and **tools.**
  - It enables applications and services to use **LDAP** for user **authentication:**
    - Every service that uses PAM will be able to use LDAP as **authentication database:**
      - **sshd, login…**
  - … This is for adding (connecting) LDAP to PAM as authentication mechanism:
    - `pam_ldap.so.`

- **`libpam-ldap`** installation:

```
$ apt-get install libpam-ldap.
```

PAM + LDAP: User authentication

- **Configuration:**

  `$ vi /etc/pam_ldap.conf`

  - This file uses the same config items as **libnss-ldap:**
    - Both usually have the same content.

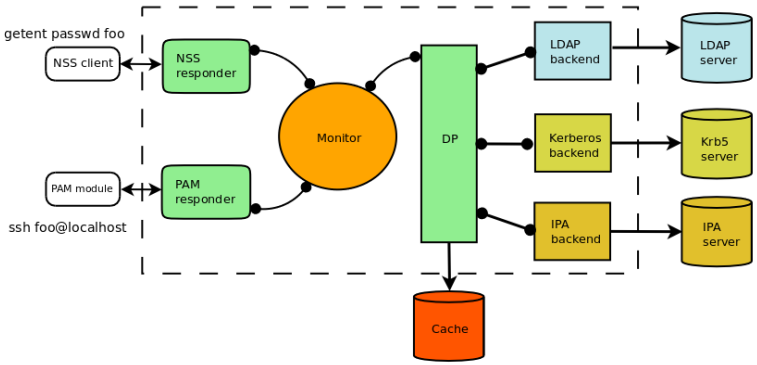
- Use `dpkg-reconfigure` to configure both **`libnss-ldap`** and **`libpam-ldap`**:

```
$ dpkg-reconfigure libnss-ldap.
$ dpkg-reconfigure libpam-ldap.
```

SSSD: NSS + PAM integration

- **SSSD: S**ystem **S**ecurity **S**ervices **D**aemon [Red Hat – Fedora]:
  - It provides:
    - A set of **daemons** to manage access to remote directories and authentication mechanisms.
    - An **NSS and PAM interface** toward the system and a pluggable backend system to connect to multiple and different account sources as well as D-Bus interface.
    - Offline authentication:
      - **Cache of user identities and credentials.**
    - A more robust database to store local users as well as extended user data.
  - It acts as an **intermediary** between local clients (apps/services) and any back-end provider.
  - Roles:
    - Identification → **ldap**, ipa, krb5, ad, proxy.
    - Authentication→ **ldap**, ipa, krb5, ad, proxy.
    - Access control → permit, deny, **ldap**, ipa, ad, simple.
    - Password management → permit, deny, **ldap**, ipa, ad, simple.
  - PAM and NSS own modules (client):
    - `pam_sssd` y `nss_sssd`.
  - Main features:
    - It supports several *back-ends:*
      - **LDAP, Kerberos, IPA, AD, proxy.**
    - Only allows encrypted channel (ldaps…).
    - It uses **D-BUS** as inter-process communication.
    - It can manage several domains.
    - Cache *on-disk.*
    - There is support for many *3-party* applications/services.
- **Architecture:**
  - Monitor [**`sssd`**]:
    - Main process which manages client requests.
  - *Responders:*
    - NSS → identification [**`nss`**].
    - PAM → authentication [**`domain/ldap`**].
  - Data provider [**`DP`**].

SSS components



Source: DocPlayer.net.

SSSD: NSS + PAM integration

- **Installation:**
  - `$ apt-get install sssd`
- **Configuration:**
  - **A single configuration file**
    `/etc/sssd/sssd.conf`

Format (syntax) →

```
[section]
key = value
key2 = value2,value3
```

→ **Global** parameters:

`[sssd]` monitor configuration section:

`services` list of services started by `sssd`.

`domains` user databases. At least one of them must be started.

→ **Service** parameters:

`[nss]`, `[pam]`, `[sudo]`… services configuration sections.

→ **SSSD domain** parameters:

`[domain/NAME]` domain configuration sections.

`id_provider` provider ID for this domain. "`proxy`", "`local`", "`ldap`", "`ipa`" or "`ad`"

`ldap_uri` Uniform Resource Locators → protocol ("`ldap://`", "`ldaps://`" or "`ldapi://`") and LDAP server name.

`ldap_search_base` LDAP DN used to search users.

`cache_credentials` It establishes whether or not user credentials will be cached (`LDB cache`).

All together: …