Single Sign On

CIS 2235 Adv System Administration

Overview

Once upon a time, all users shared the same machine ("time-shared" computing)

Now, every user has their own machine

- some services run locally on the desktop
- others are accessed remotely
- many require some form of authentication

The problem?

- too many user names and passwords
- ideally, would like to be able to authenticate once
- but how?

Answer? Single Sign On (SSO)

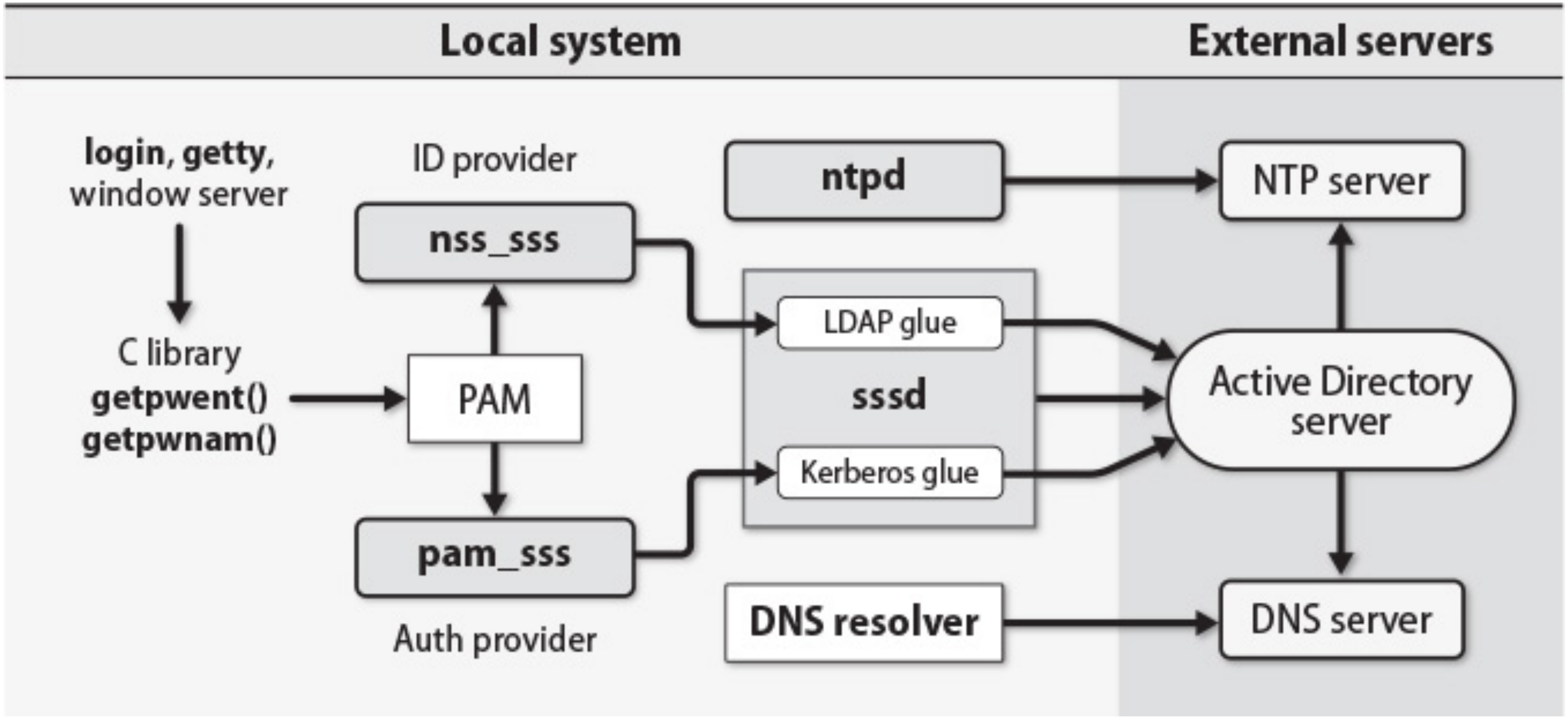
Identity, Authentication, and Authorization

- Concepts
  - User Identity - an abstract representation of an individual.
    - for example, a username represents a single user
  - Authentication - proving that you are the person represented by the abstract identity
    - How do we know that you are you in a safe, secure way?
  - Authorization - determining level of access permitted for a given identity

Core elements for SSO

- Centralized directory store of user identity and authorization information and method of accessing it
  - Windows Active Directory (AD)
  - LDAP
- Tool for managing information in the directory
  - phpLDAPadmin or Apache Directory Studio for LDAP
  - appropriate Windows tool for AD
- A way to authenticate user identities
  - via PAM to either LDAP or Kerberos (AD)
- Tools that will use centralized identity lookup methods to authenticate

SSO flow


Local system
External servers
login, getty,
window server
C library
getpwent()
getpwnam()
ID provider
nss_sss
PAM
pam_sss
Auth provider
ntpd
LDAP glue
sssd
Kerberos glue
DNS resolver
NTP server
Active Directory server
DNS server

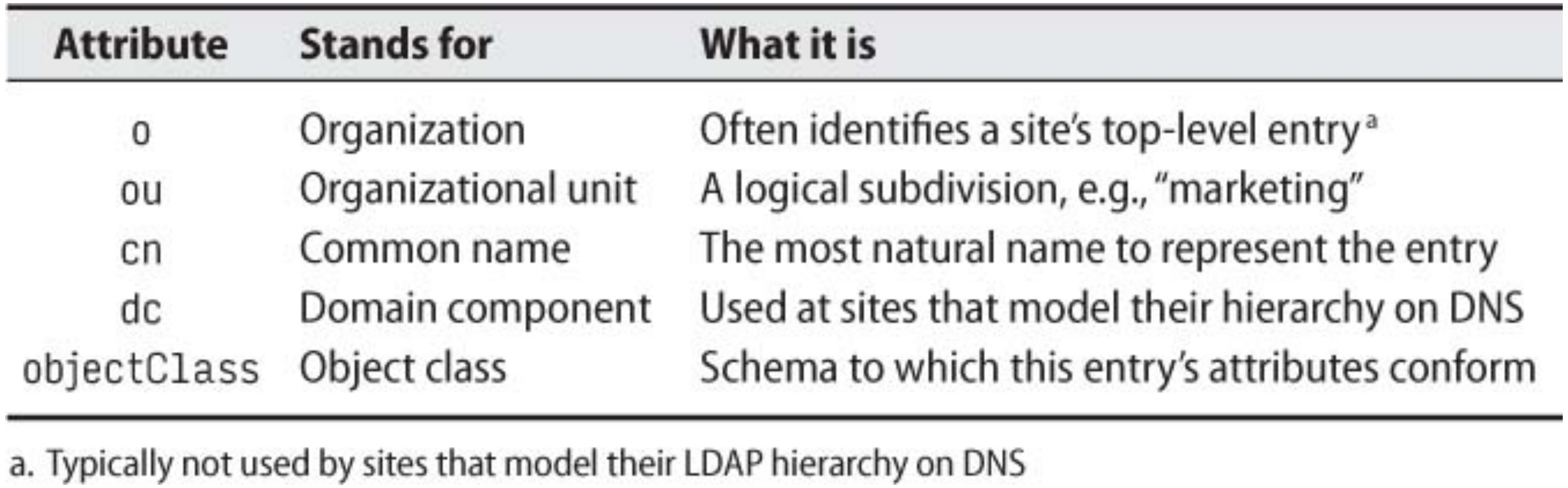
Directory Store

- A directory store is just a place to information
  - usernames, phone numbers, addresses, etc
  - X.500 was a definition of such a directory store
  - MS Active Directory is a directory store
- LDAP
  - “Lightweight” Directory Access Protocol
  - originally just a description of how to talk to an X.500 directory
  - X.500 is obsolete, but LDAP still persists
  - can be used to interact with AD

Organizational units

- LDAP compliant directories
  - have entries
    - just property lists
  - organized in a hierarchy
    - based on organizational units and distinguished names

```
dn: uid=ldamon,dc=cis,dc=vtc,dc=vsc,dc=edu
```



Attribute	Stands for	What it is
o	Organization	Often identifies a site's top-level entry[a]
ou	Organizational unit	A logical subdivision, e.g., "marketing"
cn	Common name	The most natural name to represent the entry
dc	Domain component	Used at sites that model their hierarchy on DNS
objectClass	Object class	Schema to which this entry's attributes conform

[a] Typically not used by sites that model their LDAP hierarchy on DNS

LDAP choices

- In addition to AD as a directory store, there are several LDAP native choices
  - OpenLDAP - traditional server
  - 389 Directory Server - Fedora maintained server
    - better overall documentation
    - more active community
  - Both are from the same original code base, so they are administered much the same way
- Configuring LDAP is beyond the scope of what we are doing in this course, but look on Canvas for some additional resources

Using directory services

Once you have a directory service, you have to configure services to use it

First, setup sssd

- System Security Service Daemon
- centralized daemon for authentication and account management
- not strictly required, but provides additional features and centralized administration
- supports authentication through both LDAP and AD (via Kerberos)

Using directory services

Lastly, let’s switch to use ssd

/etc/nsswitch.conf

- the name service switch
- lists types of lookups and sources

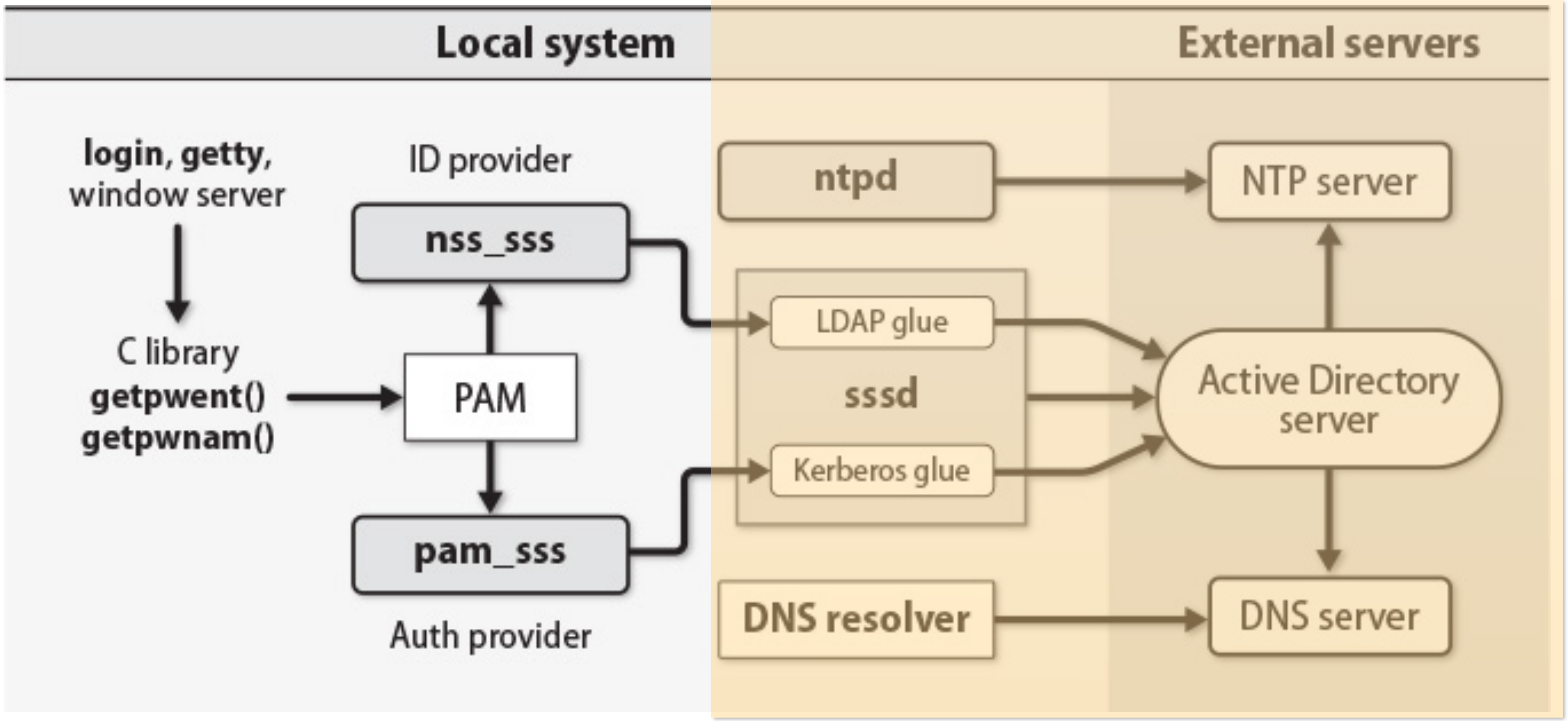
```
passwd:         compat sss
group:          compat sss
shadow:         compat sss
gshadow:        files

hosts:          files dns
networks:       files

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

netgroup:       nis
```

Recap


Local system
External servers
login, getty,
window server
C library
getpwent()
getpwnam()
ID provider
nss_sss
PAM
pam_sss
Auth provider
ntpd
LDAP glue
sssd
Kerberos glue
DNS resolver
NTP server
Active Directory
server
DNS server

Alternatives

As described, SSO is a big hammer worthy of being used for big problems

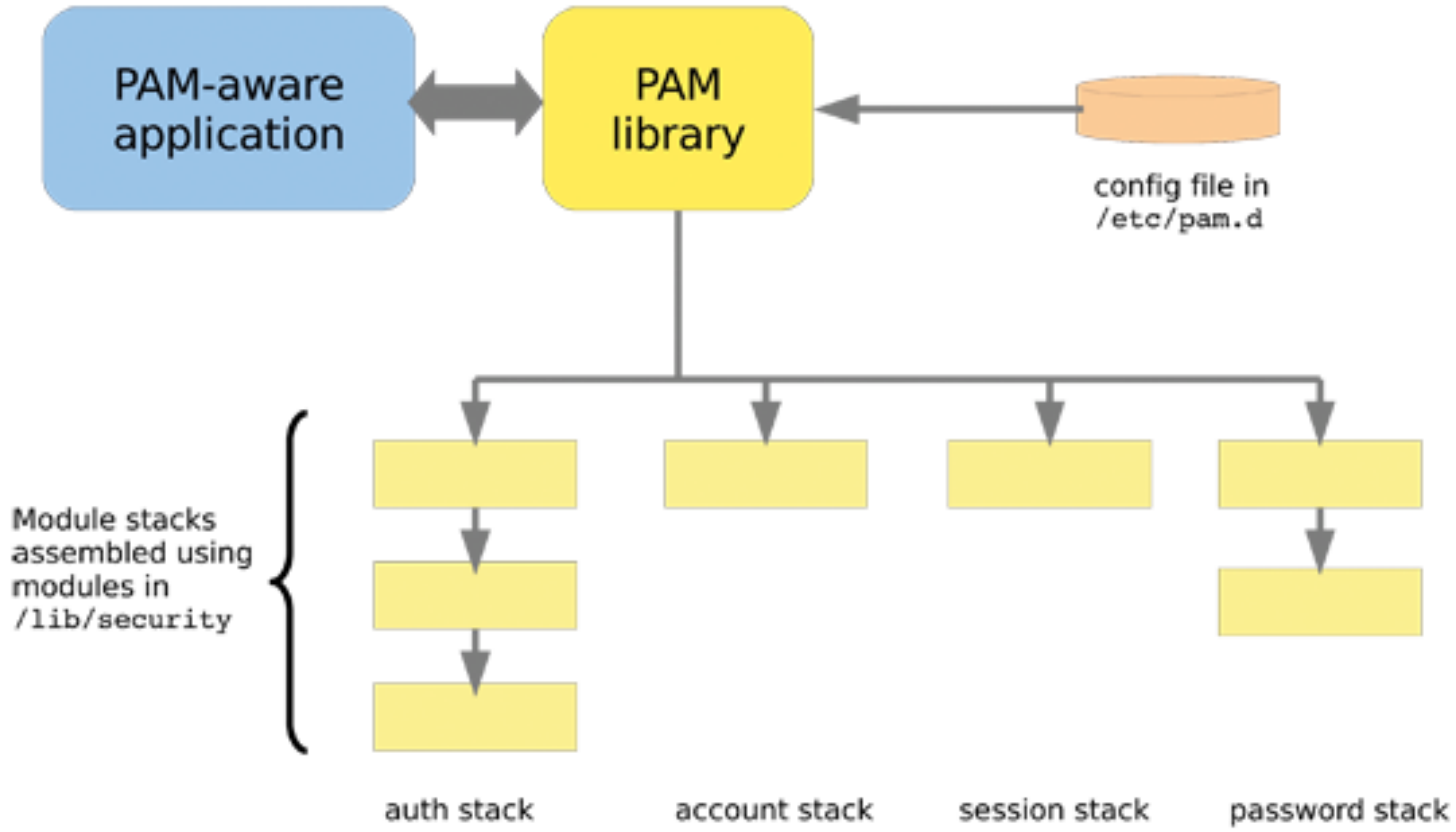
- Smaller organizations have smaller problems
- might not be worth the effort
- can instead do a “roll your own” syncing of files using tools like rsync or modern configuration management tools like ansible to copy files between systems.

PAM - Pluggable Authentication Modules

“PAM is a framework that assists applications in performing … "authentication-related activities.” The core pieces of PAM are a library (libpam) and a collection of PAM modules, which are dynamically linked libraries (.so) files in the folder /lib/security.

Each module performs one specific task, and a "PAM-aware" application typically uses a stack of several modules to accomplish its goals.

PAM - Pluggable Authentication Modules



PAM - types of modules

- auth
  - proving who you are by providing appropriate credentials
    - username/password, biometric, etc
- account
  - are you allowed to log in? Time of day restrictions, for example
- password
  - password updates (as discussed last class)
- session
  - resources needed for this session. Mount of home directory, for example

PAM

Configured as a stack
Example:

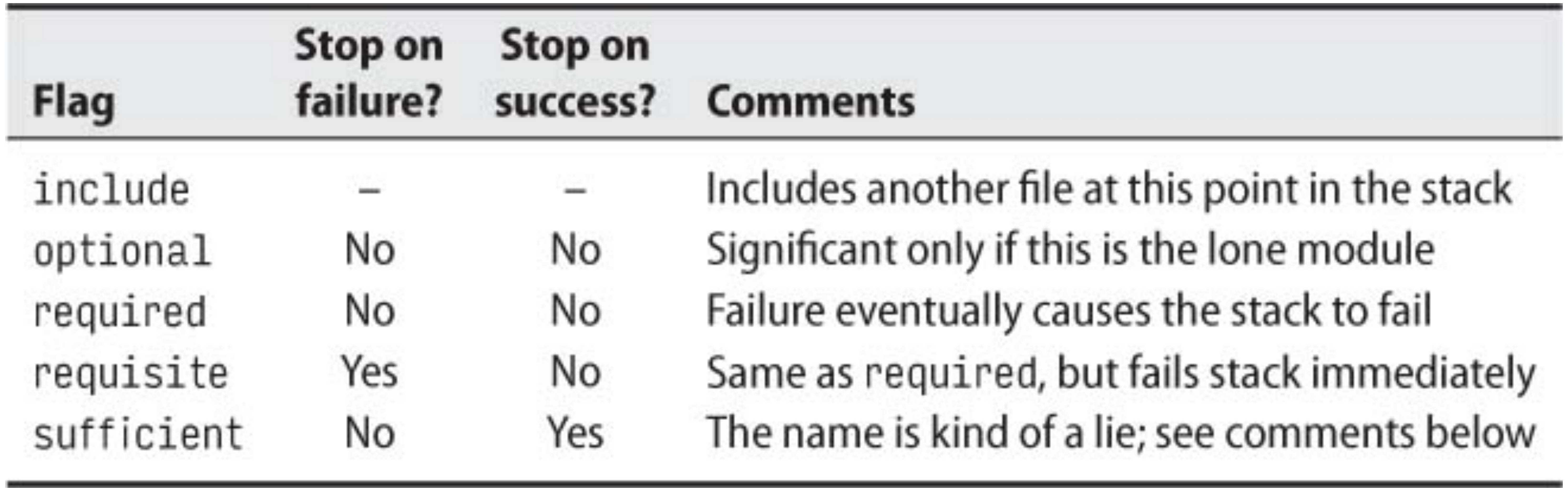
```
auth       required   pam_securetty.so
auth       required   pam_unix.so nullok
auth       required   pam_nologin.so
```

pam_securetty.so – only allow root logins from secure terminals listed in /etc/securitytty

pam_unix.so – standard unix module. Does usual unix account login checks – account not expired, etc. nullok says it is okay to login without a password (gulp!)

pam_nologin.so - if the file /etc/nologin exists, and the user is not root, login will fail.

PAM control flags



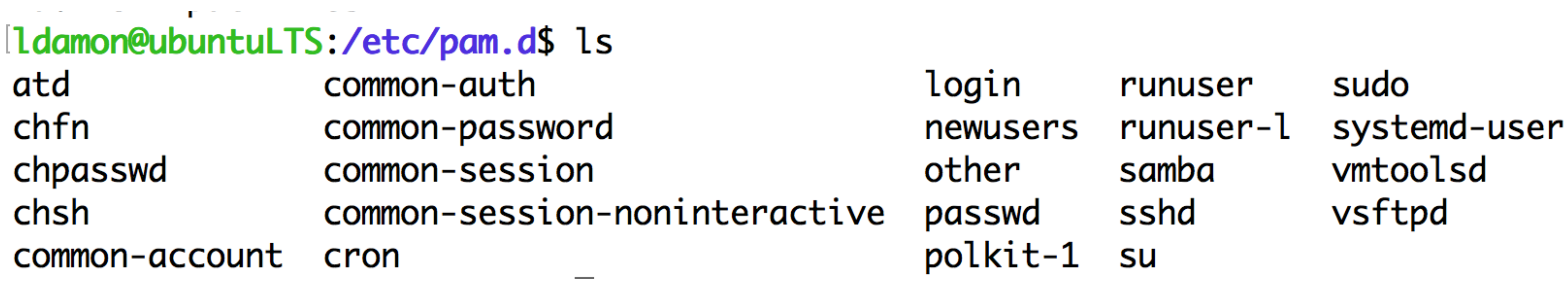
Flag	Stop on failure?	Stop on success?	Comments
`include`	–	–	Includes another file at this point in the stack
`optional`	No	No	Significant only if this is the lone module
`required`	No	No	Failure eventually causes the stack to fail
`requisite`	Yes	No	Same as `required`, but fails stack immediately
`sufficient`	No	Yes	The name is kind of a lie; see comments below

- Control flags tell PAM what to do if a module returns false
  - can be used to hide elements of the stack from a probing application
- The success of a sufficient module stops processing, but not always with a success. It doesn’t override a previous failure.

Configuring PAM

PAM is configured via files /etc/pam.d

- Each PAM-aware app places its own file in the directory
  - login uses /etc/pam.d/login, for example



```
ldamon@ubuntuLTS:/etc/pam.d$ ls
atd              common-auth                     login     runuser    sudo
chfn             common-password                 newusers  runuser-l  systemd-user
chpasswd         common-session                  other     samba      vmtoolsd
chsh             common-session-noninteractive   passwd    sshd       vsftpd
common-account   cron                            polkit-1  su
```

- Additional configuration in /etc/security/*.conf files