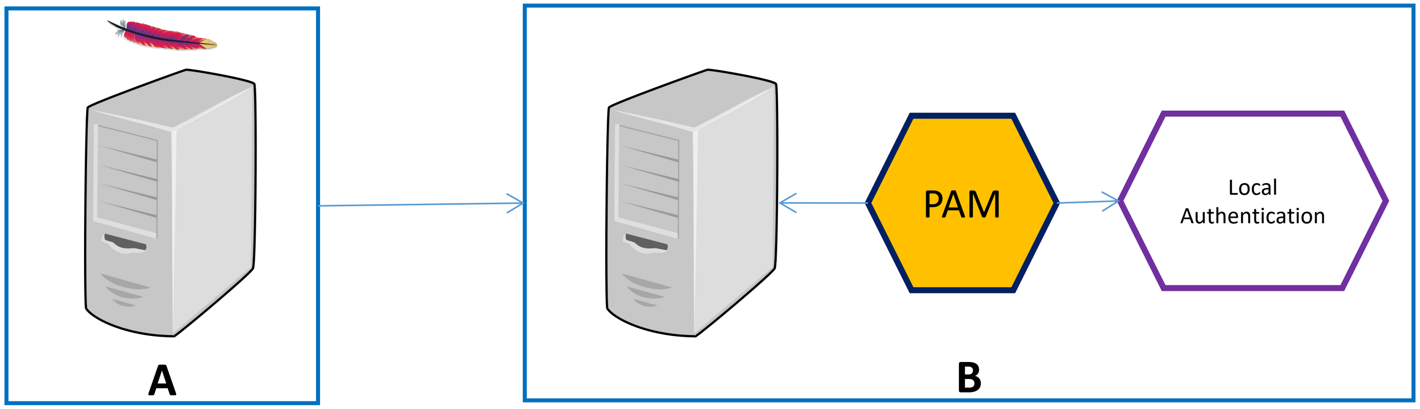
# Operating System Security #4 PAM

Herman Kabetta

## What is PAM?

- What is PAM?

**Pluggable Authentication Modules** provide dynamic authentication support that sits between Linux application and the Linux native authentication system.

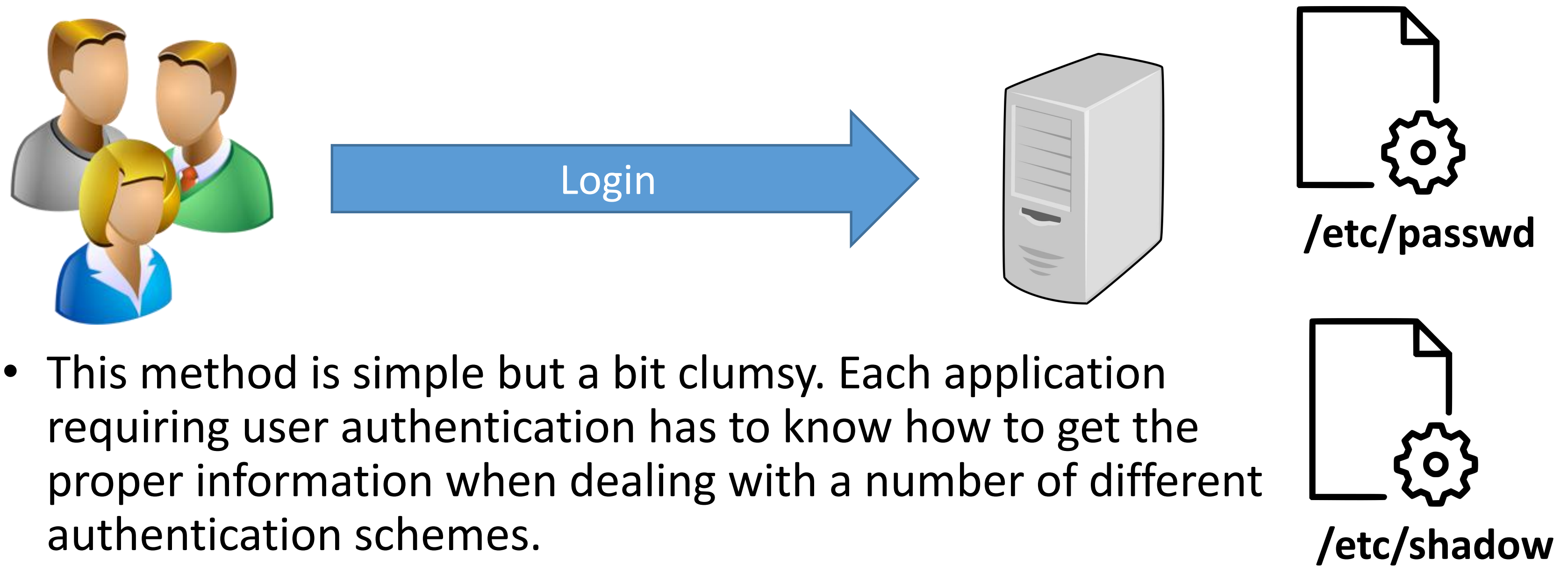


- The main purpose of PAM is to allow system administrators to integrate services or programs with different authentication mechanism without changing the code for the service
- There are many programs in your local system that use PAM Modules, e.g. SU, password, SSH, logging in FTP, TELNET, etc.

## What is PAM?

- The configuration of Linux-PAM can be done in two ways. You can either put everything in one single file as **/etc/pam.conf** or split the configuration by service in the directory with **/etc/pam.d** = *(better for each individual services)*
- Keep in mind that Linux-PAM will ignore **/etc/pam.conf** if the **/etc/pam.d** directory exists
- If a service or program does not have a config file then it will consult **/etc/pam.d/other** config file
- IMPORTANT: If PAM is wrongly configured then you will not be able to login. Therefore, its recomended that you take a snapshot or backup of your system
- PAM sends all its activity information to
  - **/var/log/messages**
  - **/var/log/secure**

## The Importance of PAM

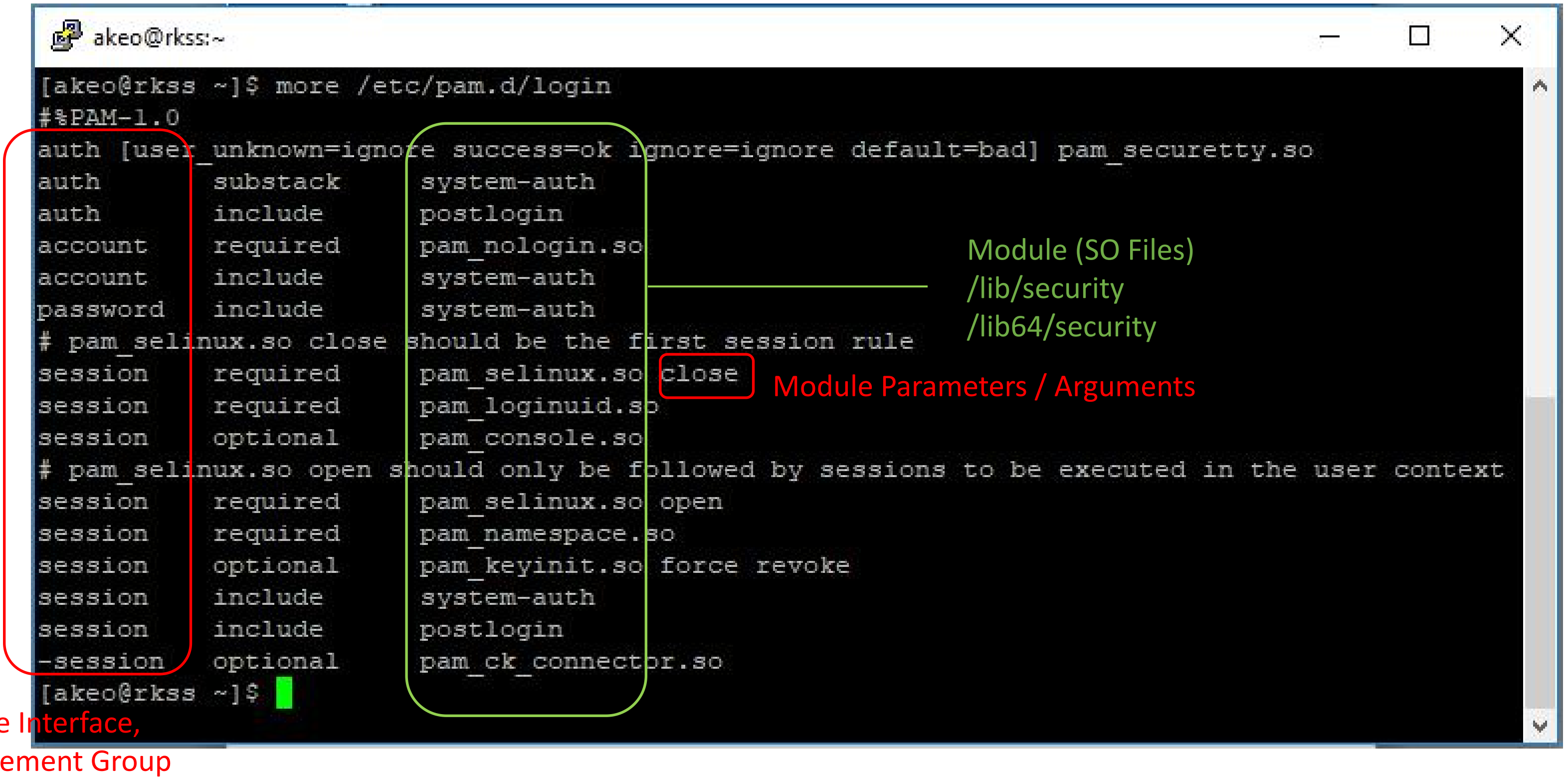


- This method is simple but a bit clumsy. Each application requiring user authentication has to know how to get the proper information when dealing with a number of different authentication schemes.
- As new authentication schemes is built the old ones become obsolete. In other words, if a system administrator wants to change the authentication scheme, the entire application must be recompiled.

  **Authentication examples: RAS, smart card, biometrics, etc.**
- Whenever new or updated authentication released all services (login, ftp, ssh, etc) had to be recompiled.
- PAM solved that problem

# The PAM File Configuration

- If you open any service file, you will see that the file is divided into three columns. the first column is management group, the second column is for control flags, and the third column is the module (SO file) used.



```
[akeo@rkss ~]$ more /etc/pam.d/login
#%PAM-1.0
auth [user_unknown=ignore success=ok ignore=ignore default=bad] pam_securetty.so
auth       substack     system-auth
auth       include      postlogin
account    required     pam_nologin.so
account    include      system-auth
password   include      system-auth
# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so close
session    required     pam_loginuid.so
session    optional     pam_console.so
# pam_selinux.so open should only be followed by sessions to be executed in the user context
session    required     pam_selinux.so open
session    required     pam_namespace.so
session    optional     pam_keyinit.so force revoke
session    include      system-auth
session    include      postlogin
-session   optional     pam_ck_connector.so
[akeo@rkss ~]$
```

# The PAM File Configuration - Module Interface

## FIRST COLUMN - Module Interface or Type

- Linux-PAM separates the tasks of authentication into four independent management groups :
  - **Authentication**: verify the user's identity, for example by requesting and checking a password or other secret.

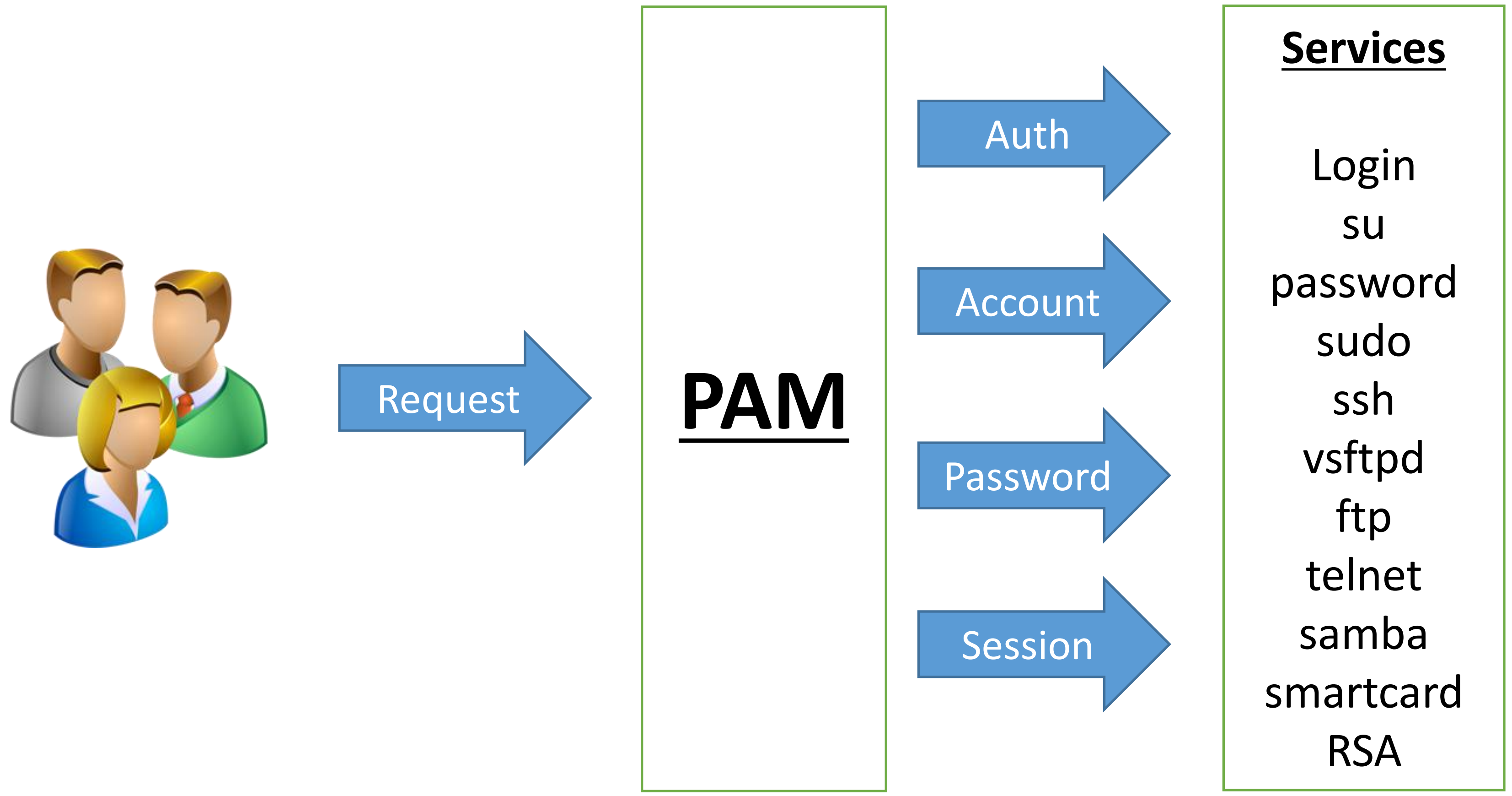
    e.g. Username/Password = OK
  - **Account**: check that the specified account is valid. This may include conditions like account expiration, time of day and that the user has access to the requested service.

    eg. User Account = Enabled, Not Locked, Not Expired, Allowed to login at this time, has service access
  - **Password**: are responsible for updating passwords, and work together with authentication step. They may also be used to enforce strong passwords.

    e.g. Password Update = Only when password is changed, enforces password policies like password length, retires, etc.
  - **Session**: manage actions performed at the beginning of a session and end of a session

    e.g. Establish session, making sure hoe directory is created if needed, setting up user environment, etc.

# Account Access Through PAM



# The PAM Configuration File - Control Flags

## SECOND COLUMN - Control Flag

We have four control flags in services files

1. **Requisite**: The strongest flag. If a module Interface is flagged as requisite and it fails, PAM will return to the calling application and report the failure.
2. **Required**: In the case failure, execution is not stopped but continues to next module. If, after all of thmodeuls have been executed, one or more has failed, PAM will return failure to the calling application.
3. **Sufficient**: If a sufficient module returned OK, the processing of modules will stopped.
4. **Optional**: In the case of failure, the stack of modules continues execution and the return code is ignored.

In addition to the above there are two other valid control flags :

- **Include**: Include all lines of given type from the configuration file specified as an arguments to thiscontrol.
- **Substack**: Same as above.

## The PAM Configuration File - Modules (SO)

### THIRD COLUMN - Dynamically Loaded Modules (SO Files)

- PAM Loaded object files (the modules) are usually located in the following directories:

  **/lib/security** or **/lib64/security** depending on the architecture
- A module can provide mechanisms to authenticate users from any backend like a file **/etc/passwd** or database such as WinBind, AD, OpenLDAP, etc.
- Most of these modules are pre-built and comes pre-installed with Linux OS distribution. The programmers or developers can also write new modules based on their application requirement.
- The main module in any distribution is **pam_unix.so** which is responsible to verify authentication
- Each details can be pulled from man pages e.g. **man pam_unix**

## PAM Aware Services and Stacks

- How to Check a Program is PAM-aware

```
ldd usr/sbin/sshd | grep pam
ldd bin/su | grep pam
```

- **Modules Order**

The Linux-PAM modules in the stack are tried one by one.

The other matters because the effect of one module is required for the next module to work correctly.

A configuration like the following for login will work properly :

```
auth required pam_unix.so
auth optional pam_deny.so
```

But if you change the order like this :

```
auth optional pam_deny.so
auth required pam_unix.so
```

Then no one can log in, so the order matters

- PAM is powerful high-level API that allows programs that rely on authentication to authentic users to applications in a Linux system. It's powerful but very chalenging to understand and use