PAM
(Pluggable Authentication Modules)

Motivation

- Users
  - Unification of authentication mechanisms for different applications
- Manufacturers
  - Authenticated access to services independent of authentication mechanisms
- Administrators
  - Easy orchestration of authentication mechanisms different services requiring client authentication
  - Flexibility to configure specific authentication mechanisms for each host
- Manufacturers and Administrators
  - Flexible and modular approach for integrating novel authentication mechanisms

PAM: features

- Independent authentication protocols / mechanisms
  - Linux password, S/Key, smartcards, biometrics, etc.
  - One module per protocol / mechanism
- Orchestration of protocols / mechanisms
  - Alone or combined
  - AND and OR combinations
  - Application-independent
- Several interface approaches
  - Input from text consoles of graphical windows
  - Access to special devices (smart-cards, biometric readers, etc.)

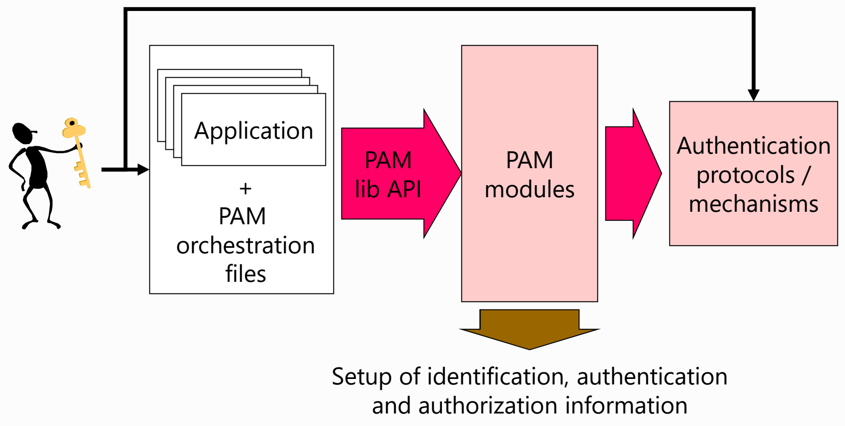
PAM: features

- Modular and extensible architecture
  - Dynamic loading of required modules
  - Handling of several actions besides authentication
    - Password management
    - Accounting management
    - Session management
- Default orchestration per host
  - Defined by the administrator
    - Username/password, biometrics, smart-cards, etc.
- Application-specific orchestrations
  - Each application can use a unique orchestration

Classic Unix authentication

- ▷ Requested input: username + password
- ▷ Validation
  - Active account for username
    - Entry with the username in the /etc/passwd file
  - Transformed password for that username
    - Entry with the username in the /etc/shadow file
  - Transformation of the provided password with the function and the salt used for that username
  - Comparison with the stored transformation
- ▷ Obtained credentials
  - UID + GID [+ list of secondary GIDs]
  - New process descriptor (login shell)

PAM: Architecture


Application
+
PAM orchestration files
PAM lib API
PAM modules
Authentication protocols / mechanisms
Setup of identification, authentication and authorization information

PAM: Actions

- Authentication (auth)
  - Identity verification
- Account Management (account)
  - Enforcement of access policies based on account properties
- Password Management (password)
  - Management of authentication credentials
- Session Management (session)
  - Verification of operational parameters
  - Setup of session parameters
    - max memory, max file descriptions, graphical interface configuration, ...

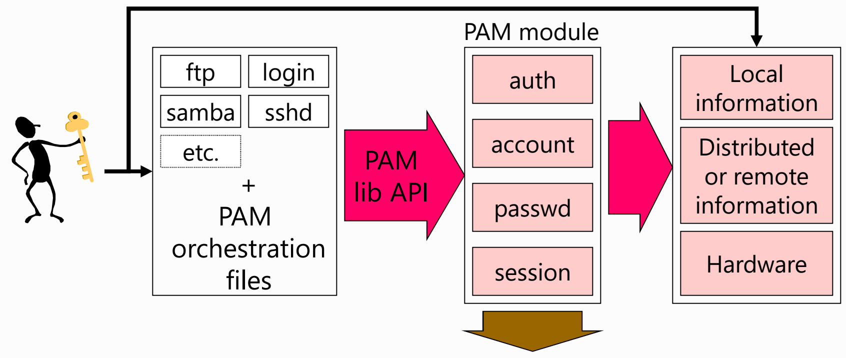
PAM: Modules

- Dynamically loaded (*shared libraries*)
  - /lib/security/pam_*.so
  - /lib/x86_64-linux-gnu/security/pam_*.so
- Standard API
  - Functions provided by the modules that are used
    - C interfaces
  - Decision provided on returned code
    - PAM_SUCCESS
    - PAM_AUTH_ERR, PAM_AUTHINFO_UNAVAIL, etc...
  - Not all functions need to be implemented
    - A module does not need to implement all 4 actions

PAM: orchestration files

- ▷ Typically, one per PAM client application
  - e.g. /etc/pam.d/ftp or /etc/pam.d/ssh
  - Can use shared files: /etc/pam.d/common-auth
- ▷ Specify how the actions should be applied
  - Their mechanisms (modules)
  - Their parameters
  - Their termination, with or without success
- ▷ Each module uses a particular set of resources
  - Local files
    - /etc/passwd, /etc/shadow, /etc/groups, etc.
  - Distributed information or located in remote servers
    - NIS, Kerberos, LDAP, etc.

PAM: Detailed Architecture



Setup of identification, authentication and authorization information

PAM APIs: PAM lib

- Start/end of the PAM lib
  - pam_start(service, user name, callback, &pam_handle)
  - pam_end(pam_handle, status)
- Module specific data
  - pam_get_data(), pam_set_data()
  - pam_get_item(), pam_set_item()
- “auth” action
  - pam_authenticate(pam_handle, flags)
  - pam_setcred(pam_handle, flags)
- “account” action
  - pam_acct_mgmt(pam_handle, flags)
- “passwd” action
  - pam_chauthtok(pam_handle, flags)
- “session” action
  - pam_open_session(pam_handle, flags)
  - pam_close_session(pam_handle, flags)

Orchestration of PAM actions

- Sequence of module invocations per action
  - By default, modules are executed sequentially
  - Each module has its own parameters and calling semantic
    - Required, requisite, sufficient, optional
    - […]
  - Execution proceeds until the end, or failure
    - To better hide the source of a failure, module execution can either abort immediately or delay the failure upon executing the entire sequence
  - Applications can recover from failures

PAM APIs: PAM modules

- "auth" action
  pam_sm_authenticate(pam_handle, flags)
  pam_sm_setcred(pam_handle, flags)
- "account" action
  pam_sm_acct_mgmt(pam_handle, flags)
- "passwd" action
  pam_sm_chauthtok(pam_handle, flags)
- "session" action
  pam_sm_open_session(pam_handle, flags)
  pam_sm_close_session(pam_handle, flags)

PAM: Module invocation

- Syntax: action control module [parameters]
- Control is specified for each action and module

  requisite
  - If the module fails, the result is returned immediately

  required
  - If the module fails, the result is set but the next modules are invoked

  sufficient
  - If module fails the result is ignored
  - Otherwise, returns success if all previous "required" modules also were successful

  optional
  - Result is ignored
  - EXCEPT: if this is the only module in the action

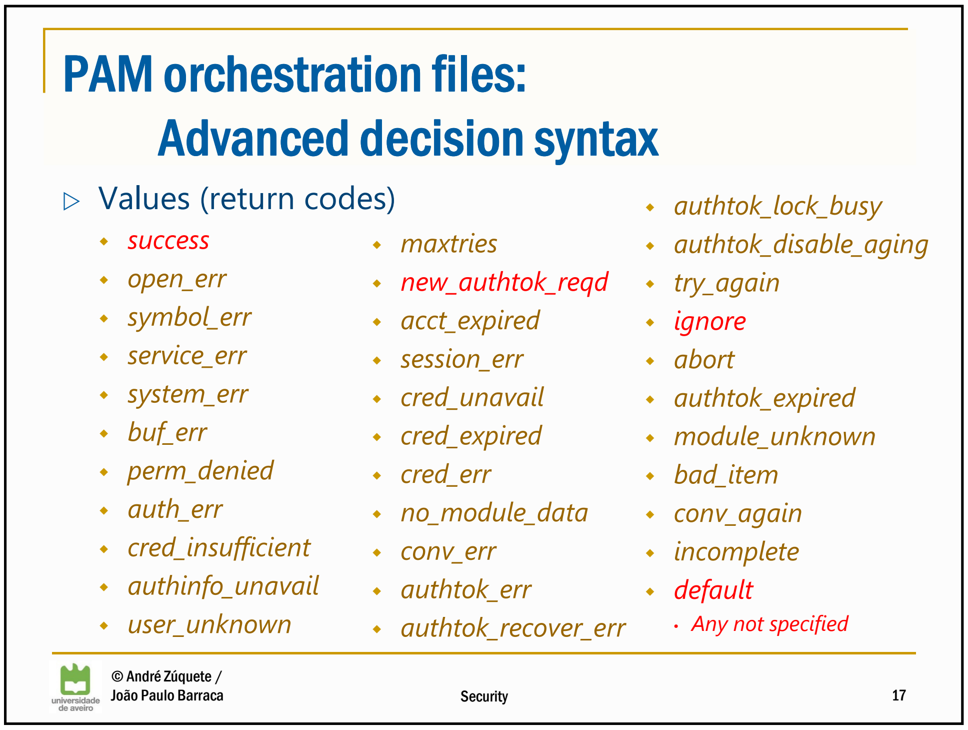
  [success=ok/number default=ignore/die/bad ...]

Configuration files: /etc/pam.d/login

```
auth optional pam_faildelay.so delay=3000000
auth [success=ok new_authtok_reqd=ok ignore=ignore user_unknown=bad default=die] pam_securetty.so
auth requisite pam_nologin.so

session [success=ok ignore=ignore module_unknown=ignore default=bad] pam_selinux.so close
session required pam_loginuid.so
session [success=ok ignore=ignore module_unknown=ignore default=bad] pam_selinux.so open
session required pam_env.so readenv=1
session required pam_env.so readenv=1 envfile=/etc/default/locale

@include common-auth
auth optional pam_group.so

session required pam_limits.so
session optional pam_lastlog.so
session optional pam_motd.so motd=/run/motd.dynamic
session optional pam_motd.so noupdate
session optional pam_mail.so standard
session optional pam_keyinit.so force revoke

@include common-account
@include common-session
@include common-password
```

PAM orchestration files: Advanced decision syntax

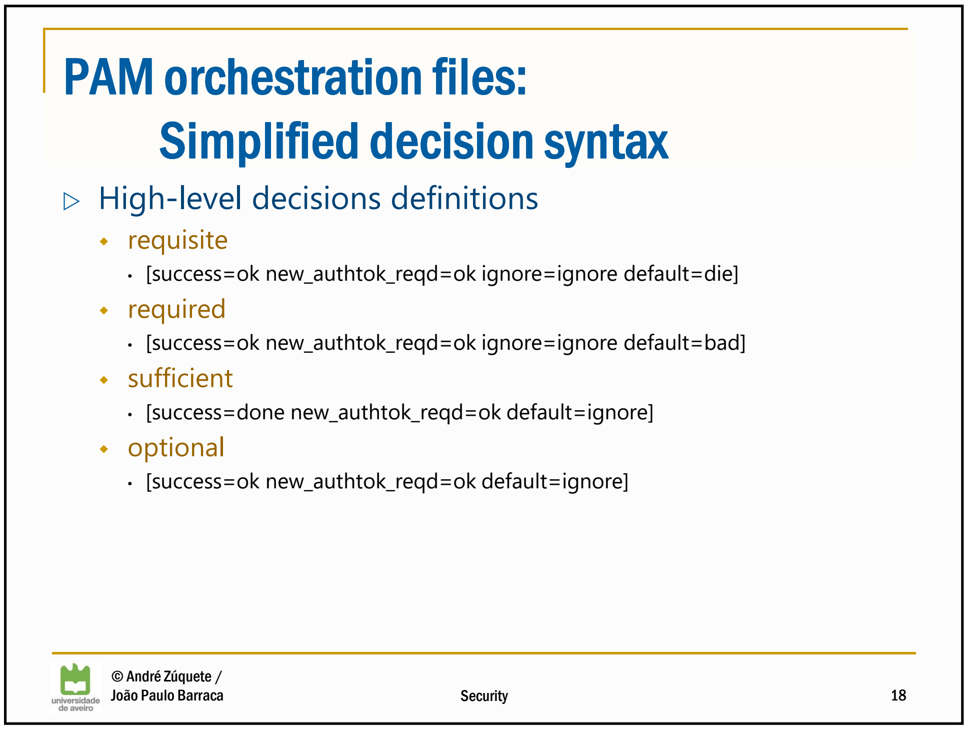
- [value=action value=action ...]
- Actions:
  - ignore: take no decision
  - bad: continue, but the final decision will be a failure
  - die: terminate immediately with failure
  - ok: continue, so far the decision is success
  - done: terminate immediately with success
  - reset: clear the entire state and continue
  - N (unsigned integer): same as ok + jump over N lines



PAM orchestration files: Advanced decision syntax

▷ Values (return codes)

- *success*
- *open_err*
- *symbol_err*
- *service_err*
- *system_err*
- *buf_err*
- *perm_denied*
- *auth_err*
- *cred_insufficient*
- *authinfo_unavail*
- *user_unknown*
- *maxtries*
- *new_authtok_reqd*
- *acct_expired*
- *session_err*
- *cred_unavail*
- *cred_expired*
- *cred_err*
- *no_module_data*
- *conv_err*
- *authtok_err*
- *authtok_recover_err*
- *authtok_lock_busy*
- *authtok_disable_aging*
- *try_again*
- *ignore*
- *abort*
- *authtok_expired*
- *module_unknown*
- *bad_item*
- *conv_again*
- *incomplete*
- *default*
  - *Any not specified*



PAM orchestration files: Simplified decision syntax

▷ High-level decisions definitions

- requisite
  - [success=ok new_authtok_reqd=ok ignore=ignore default=die]
- required
  - [success=ok new_authtok_reqd=ok ignore=ignore default=bad]
- sufficient
  - [success=done new_authtok_reqd=ok default=ignore]
- optional
  - [success=ok new_authtok_reqd=ok default=ignore]