
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 or graphical windows
 - ◆ Access to special devices (smart-cards, biometric readers, etc.)

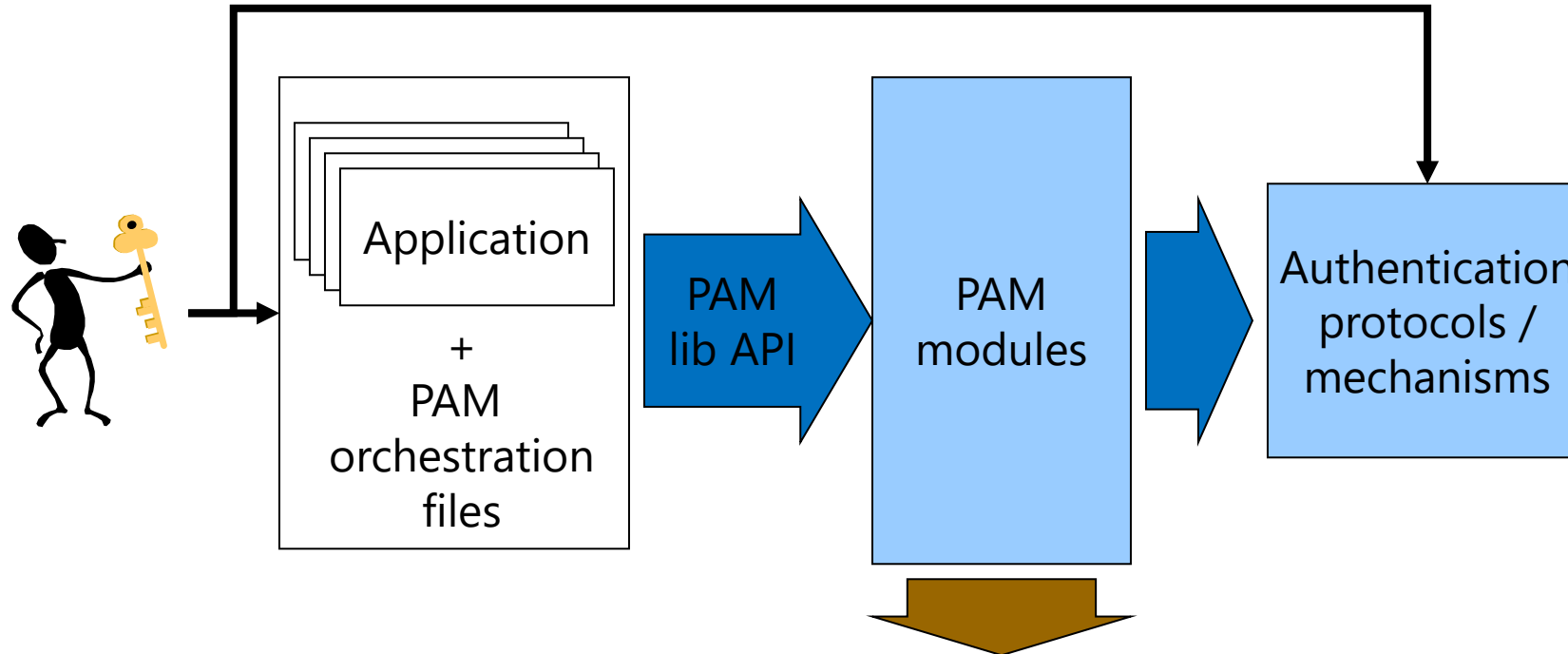
PAM: features

- ▷ 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



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, ...

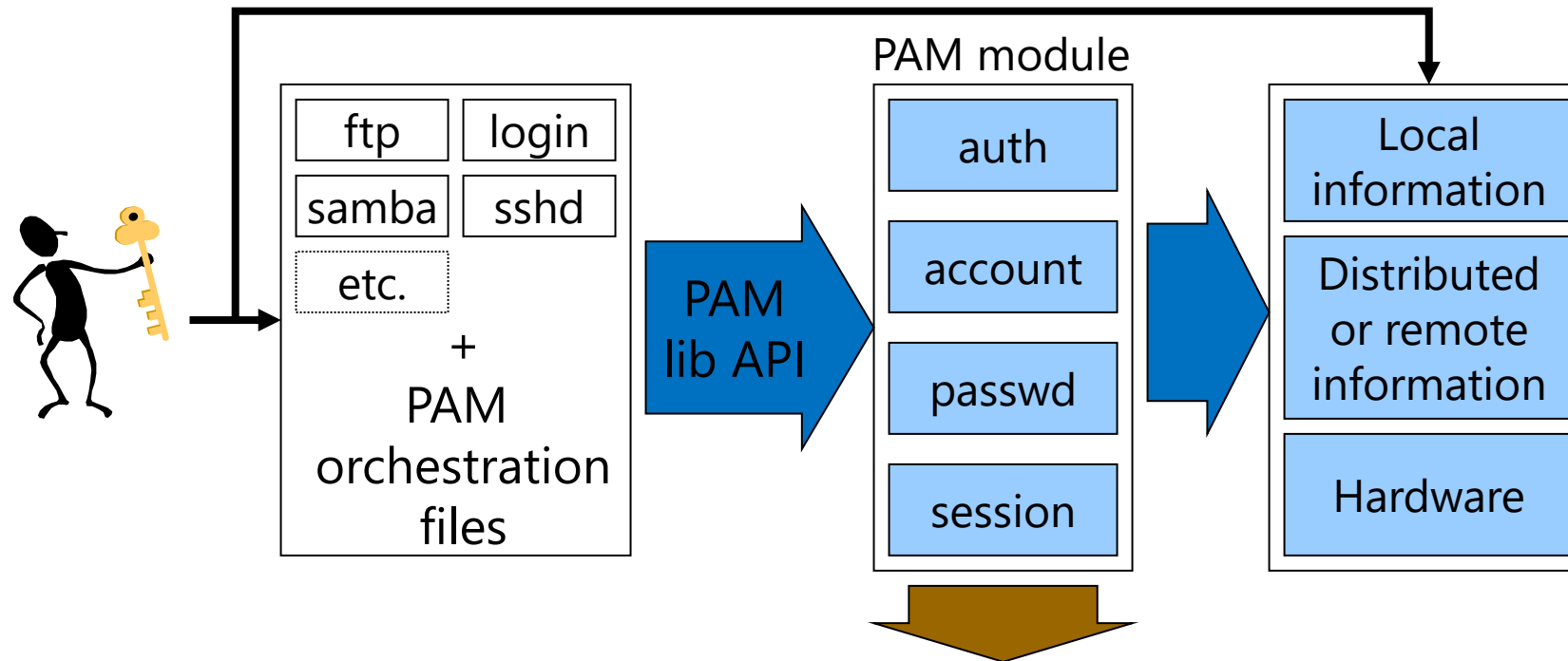
PAM: Modules

- ▷ 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
 - Python wrapper exists
 - ◆ 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 (used by applications)

- ▶ 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' API

- ▷ "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**
 - required**
 - sufficient**
 - optional**

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

PAM: Module invocation

- ▶ **required:** The module result must be successful for authentication to continue.
 - ◆ If the test fails at this point, the user is not notified until the results of all module tests that reference that interface are complete.
- ▶ **requisite:** The module result must be successful for authentication to continue.
 - ◆ However, if a test fails at this point, the user is notified immediately with a message reflecting the first failed required or requisite module test.
- ▶ **sufficient:** The module result is ignored if it fails.
 - ◆ If the result of a module flagged sufficient is successful and no previous modules flagged required have failed, then no other results are required and the user is authenticated

PAM: Module invocation

- ▷ **optional**: The module result is ignored.
 - ◆ A module flagged as optional only becomes necessary for successful authentication when no other modules reference the interface.
- ▷ **include**: Unlike the other controls, this does not relate to how the module result is handled.
 - ◆ This flag pulls in all lines in the configuration file which match the given parameter and appends them as an argument to the module.

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 nouupdate
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]

Scenario 2 – LDAP auth with local backoff

```
auth      required    /lib/security/pam_env.so
auth      sufficient /lib/security/pam_unix.so likeauth nullok
auth      sufficient /lib/security/pam_ldap.so use_first_pass
auth      required    /lib/security/pam_deny.so

account   required    /lib/security/pam_unix.so
account   [default=bad success=ok user_unknown=ignore] pam_ldap.so

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

session   required    /lib/security/pam_limits.so
session   required    /lib/security/pam_unix.so
session   optional    /lib/security/pam_ldap.so
```

Try local accounts (for offline admin) and then LDAP

Allow changing the password of both local and remote accounts

Scenario 1 – Local authentication

```
auth      required    /lib/security/pam_env.so
auth      sufficient /lib/security/pam_unix.so likeauth nullok
auth      required    /lib/security/pam_deny.so

account   required    /lib/security/pam_unix.so

password  required    /lib/security/pam_cracklib.so retry=3
password  sufficient /lib/security/pam_unix.so nullok use_authok md5 shadow
password  required    /lib/security/pam_deny.so

session   required    /lib/security/pam_limits.so
session   required    /lib/security/pam_unix.so
```

Use system files

Prevent using weak passwords

Use md5 to store passwords in shadow

Scenario 2 – LDAP auth with local backoff

- ▷ LDAP server provides directory with users
 - ◆ Identifiers, shell, email, name
 - ◆ Group membership

- ▷ saslauthd: provides interface with remote directory
 - ◆ User identifiers and attributes
 - ◆ Group membership

Scenario 2 – LDAP auth with local backoff

```
ldap_servers: ldaps://dc1.DOMAIN.TLD
ldap_search_base: dc=DOMAIN,dc=TLD
ldap_bind_dn: cn=admin,ou=host,dc=DOMAIN,dc=TLD
ldap_bind_pw: Sup3rS3cr3TP4ssw0Rd
ldap_filter: (uid=%U)
ldap_scope: sub
#ldap_group_attr: memberUid
#ldap_group_match_method: filter
#ldap_group_filter: (memberUid=%u)
#ldap_group_search_base: ou=group,dc=DOMAIN,dc=TLD
#ldap_size_limit: 0
ldap_tls_check_peer: yes
ldap_tls_cacert_file: /etc/ldap/certs.txt
ldap_tls_cacert_dir: /etc/ssl/certs/
ldap_time_limit: 15
ldap_timeout: 15
ldap_version: 3
```

Specificies where server can be found

bind_dn is a system account to query the LDAP

Group membership mapping can be set

Scenario 2 – LDAP auth with local backoff

```
auth      required    /lib/security/pam_env.so
auth      sufficient /lib/security/pam_unix.so likeauth nullok
auth      sufficient /lib/security/pam_ldap.so use_first_pass
auth      required    /lib/security/pam_deny.so

account   required    /lib/security/pam_unix.so
account   [default=bad success=ok user_unknown=ignore] pam_ldap.so

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

session   required    /lib/security/pam_limits.so
session   required    /lib/security/pam_unix.so
session   optional    /lib/security/pam_ldap.so
```

Try local accounts (for offline admin) and then LDAP

Allow changing the password of both local and remote accounts

Scenario 3 – MS AD auth with local backoff

- ▷ MS AD server provides directory with users
 - ◆ Identifiers, shell, email, name
 - ◆ Group membership
- ▷ Machine must be enrolled into domain
 - ◆ Requires login using admin credentials
- ▷ sssd: System Security Services Daemon
 - ◆ Provides and caches information from remote AD system

Scenario 3 – MS AD auth with local backoff

```
[sssd]
domains = DOMAIN.TLD
config_file_version = 2
services = nss, pam
default_domain_suffix = DOMAIN.TLD

[domain/DOMAIN.TLD]
default_shell = /bin/bash
krb5_store_password_if_offline = True
cache_credentials = True
krb5_realm = DOMAIN.TLD
realmd_tags = manages-system joined-with-adcli
id_provider = ad
fallback_homedir = /home/%u@%d
ad_domain = DOMAIN.TLD
use_fully_qualified_names = True
ldap_id_mapping = True
access_provider = simple
simple_allow_groups = group-admins
```

**Typical SSSD configuration at
/etc/sss/sss.conf**

**Specified Domain and
Kerberos5 configurations**

**Kerberos is the effective
authentication protocol issuing
authnz tickets**

Supports MFA and HSM

Scenario 3 – MS AD auth with local backoff

```
auth [success=2 default=ignore] pam_unix.so nullok
auth [success=1 default=ignore] pam_sss.so use_first_pass
auth requisite pam_deny.so
auth required pam_permit.so
auth optional pam_cap.so
```

```
account [success=1 default=ignore] pam_unix.so
account [default=bad success=ok user_unknown=ignore] pam_sss.so
account required pam_permit.so
```

```
password requisite pam_pwquality.so retry=3
password [success=2 default=ignore] pam_unix.so obscure use_authok try_first_pass yescrypt
password sufficient pam_sss.so use_authok
password requisite pam_deny.so
password required pam_permit.so
```

```
session required pam_unix.so
session optional pam_sss.so
session optional pam_systemd.so
session optional pam_mkhomedir.so
session required pam_permit.so
```

Try local accounts (for offline admin) and then AD
Deny the rest
Clear errors
Set inheritable capabilities

Set Session Settings from local conf, AD
Register session in systemd
Create home directory