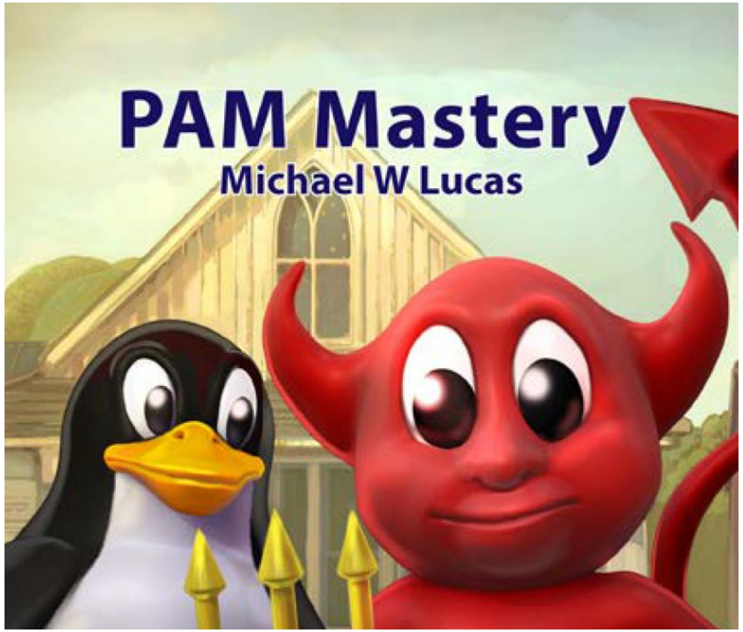
PAM

Pluggable Authentication Module

Our Source: PAM Mastery (Michael W Lucas)



Michael W. Lucas is a widely respected author and system administrator in the Unix and security world. He has written more than 15 technical books, many of which have become standard references — including *SSH Mastery*, *sudo Mastery*, *Absolute FreeBSD*, and of course, *PAM Mastery*.

He is known for his clear writing style, practical insights, and ability to make complex technical topics both accessible and enjoyable. His books are widely used by system administrators, network engineers, and security professionals around the world

Plugable Authentication Module

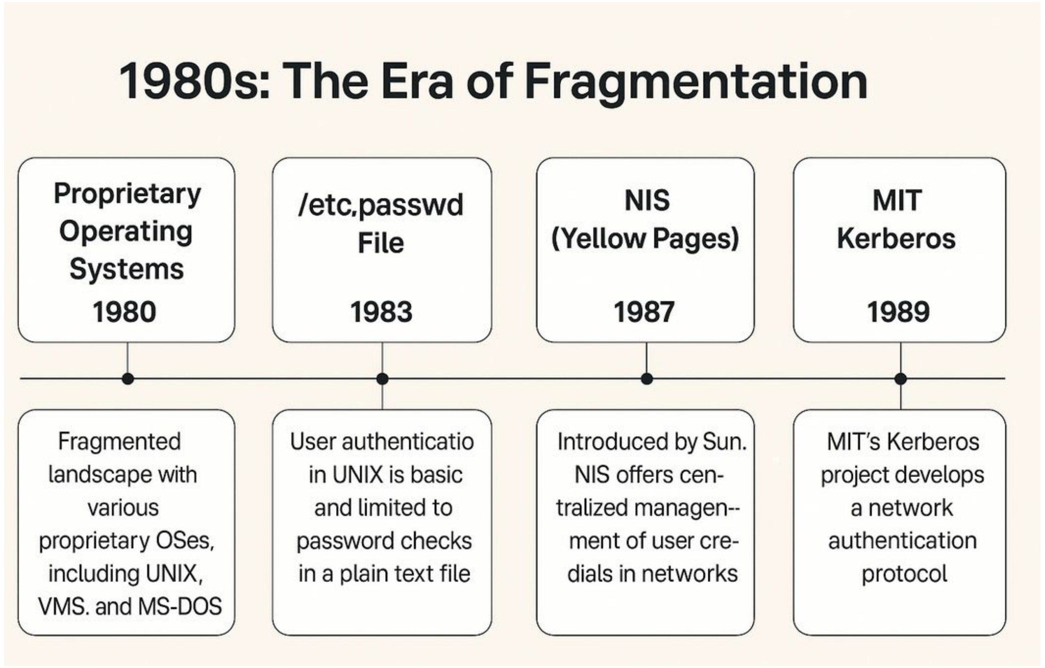
PAM, or *Pluggable Authentication Module*, is a modular framework used in Unix and Linux systems to separate authentication logic from individual applications. Instead of each program implementing its own user authentication, it delegates this task to PAM, which provides a set of configurable modules. This separation allows system administrators to apply centralized and consistent authentication policies across the system without modifying application code

What happened in the 80s?

1. The 1980s: An Era of Diversity, Disorder, and the Rise of Modern Systems
2. Authentication Problems of the Time
3. The common method was checking passwords stored in plaintext or hashed form in files like /etc/passwd.
4. Each application—such as `login`, `su`, `telnet`, `ftp`—had to implement its own authentication logic.

This meant that if an organization wanted to introduce a new authentication method (e.g., token-based), it had to modify or recompile every relevant program. This was time-consuming, error-prone, and often not feasible.

What happened in the 80s?



1980s: The Era of Fragmentation
Proprietary Operating Systems 1980
/etc,passwd File 1983
NIS (Yellow Pages) 1987
MIT Kerberos 1989
Fragmented landscape with various proprietary OSes, including UNIX, VMS. and MS-DOS
User authenticatio in UNIX is basic and limited to password checks in a plain text file
Introduced by Sun. NIS offers cen-tralized managen--ment of user cre-dials in networks
MIT's Kerberos project develops a network authentication protocol

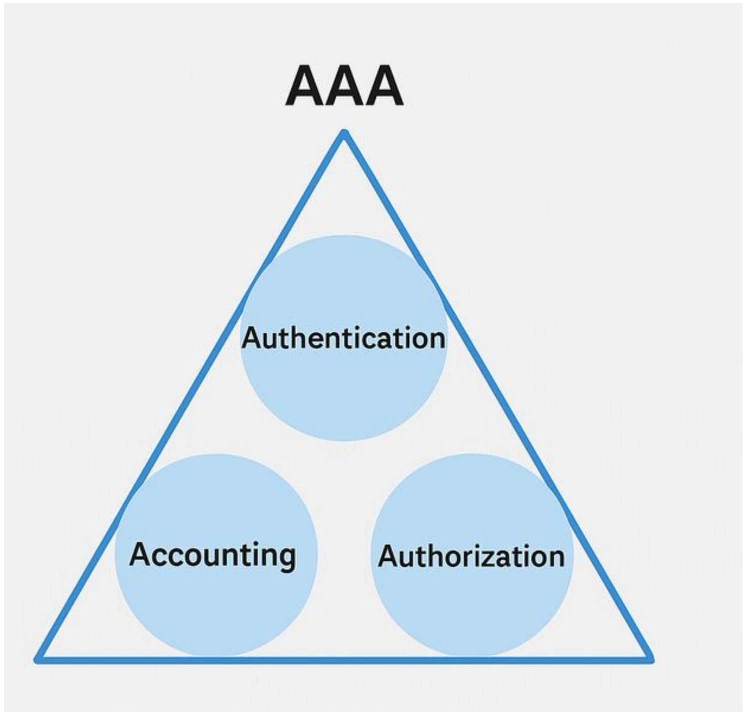
PAM vs AAA

AAA (Authentication, Authorization, Accounting) is a conceptual security model used in networks and systems to define

01 | Authentication – Who are you?

02 | Authorization – What are you allowed to do?

03 | Accounting – What did you do?



PAM vs IAA

IAA stands for three essential stages in access control and information security

01 | Identification – Who are you? The user claims an identity — for example, by entering a username or presenting a certificate. It’s just a declaration, not yet verified.

02 | Authentication– The system verifies that the identity is valid — for example, by checking a password, public key, or OTP.

03 | **Authorization**– Once identity is authenticated, the system determines what resources or actions the user is allowed to access or perform.

What is NSS?

NSS (Name Service Switch) is a system component in Unix and Linux that defines **how the system retrieves various types of configuration and identity information** — such as users, groups, hostnames, networks, and more.

Instead of hardcoding where to look for this data (e.g., always using `/etc/passwd`), NSS gives the system flexibility by allowing multiple sources (like local files, LDAP, NIS, or DNS) to be searched in a defined order.

/etc/nsswitch.conf

This file contains rules like:

```
passwd:     files ldap
group:      files ldap
hosts:      files dns
```

HOWTO? HOW NOT TO?

The Main Problem with PAM: Fragmented, Confusing, and Inconsistent Documentation

1. No Formal, Unified Standard

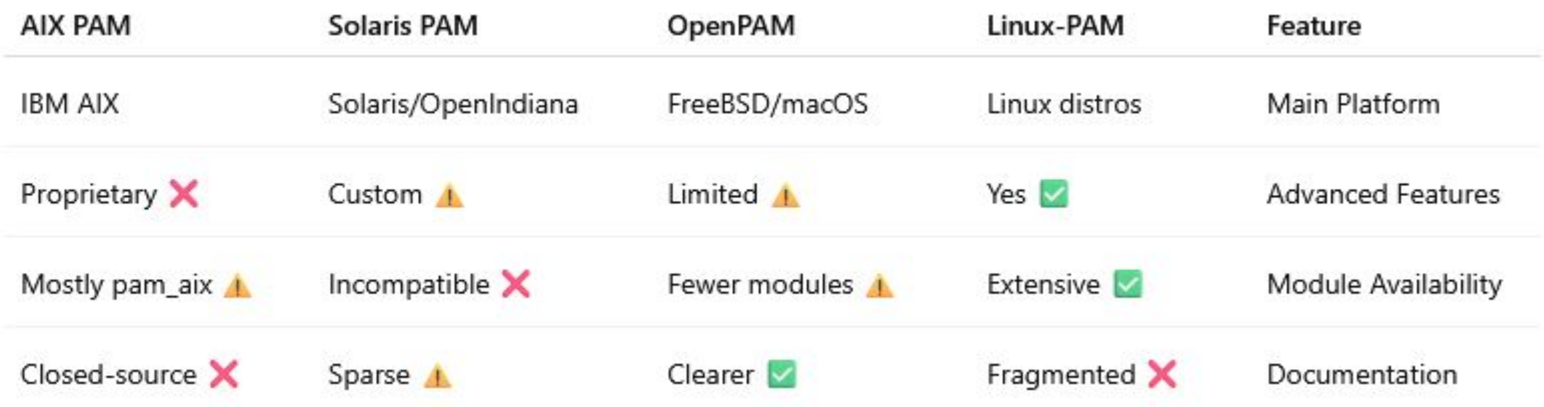
2. Divergent Behavior Across Distributions

3. Dry, Low-Level Documentation

PAM Implen

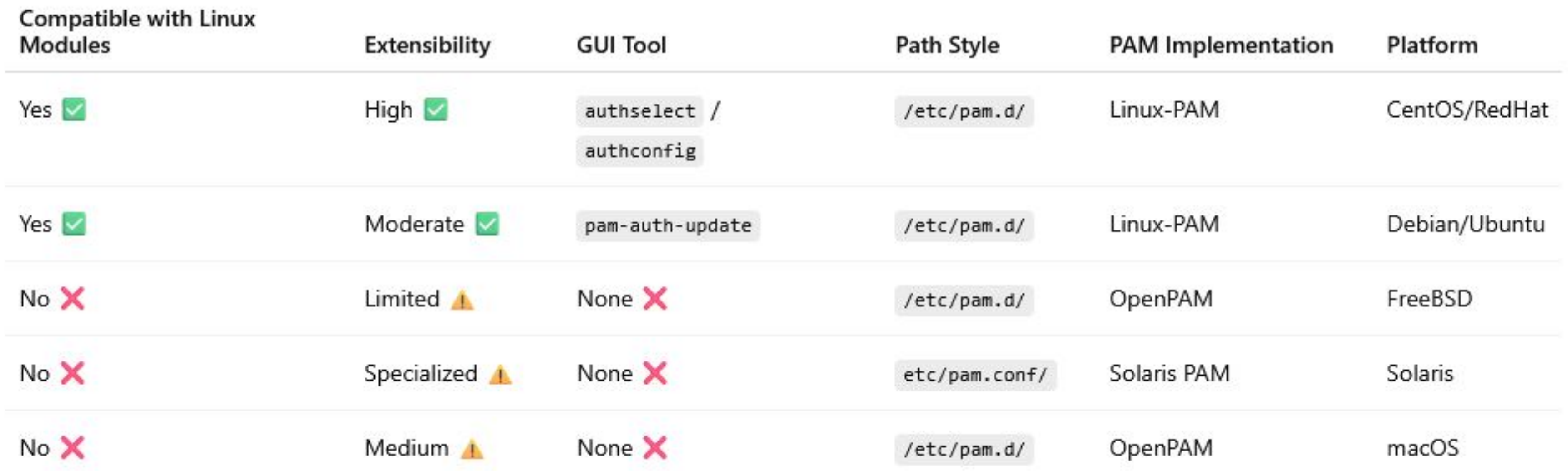
- No Direct Client Interaction
- Cannot Implement Full Authentication Protocols
- Inconsistent Implementations Across Platforms
- Security Depends on Host Application
- Difficult to Debug and High Risk of Lockout

PAM Implen



AIX PAM	Solaris PAM	OpenPAM	Linux-PAM	Feature
IBM AIX	Solaris/OpenIndiana	FreeBSD/macOS	Linux distros	Main Platform
Proprietary ❌	Custom ⚠️	Limited ⚠️	Yes ✅	Advanced Features
Mostly pam_aix ⚠️	Incompatible ❌	Fewer modules ⚠️	Extensive ✅	Module Availability
Closed-source ❌	Sparse ⚠️	Clearer ✅	Fragmented ❌	Documentation

PAM Platforms



Compatible with Linux Modules	Extensibility	GUI Tool	Path Style	PAM Implementation	Platform
Yes ✅	High ✅	`authselect` / `authconfig`	`/etc/pam.d/`	Linux-PAM	CentOS/RedHat
Yes ✅	Moderate ✅	`pam-auth-update`	`/etc/pam.d/`	Linux-PAM	Debian/Ubuntu
No ❌	Limited ⚠️	None ❌	`/etc/pam.d/`	OpenPAM	FreeBSD
No ❌	Specialized ⚠️	None ❌	`etc/pam.conf/`	Solaris PAM	Solaris
No ❌	Medium ⚠️	None ❌	`/etc/pam.d/`	OpenPAM	macOS

OpenSSH vs PAM

Explanation	Step
.SSH client connects to `sshd`	Initial SSH connection
. `sshd` verifies key or password at protocol level	Transport layer check
. `sshd` hands off authentication to **PAM** modules	If password is required
Executes modules like `pam_unix` , `pam_google_authenticator` , . `pam_faillock` etc	PAM's job
.If PAM approves, `sshd` grants access	Final decision

OpenSSH vs PAM

```
UsePAM yes

ChallengeResponseAuthentication yes

PasswordAuthentication yes

PermitUserEnvironment yes

AuthenticationMethods:
    publickey, keyboard-ineractive
```

PAM Config Files

- /etc/pam.d/
- /etc/pam.config
- /usr/local/etc/pam.d

File named after service: sshd, login, gdm, etc

Pam Statement Parts

- Statement type (or facility)
- PAM Control
- PAM shared library
- Shared library options

```
<module-type>   <control-flag>   <module-path>   [arguments...]
```

```
auth    required    pam_unix.so    try_first_pass
```

PAM Statement Type (module-type)

Type	Description
`auth`	Authentication (e.g., verifying password or OTP)
`account`	Account checks (e.g., account expiration, group access)
`password`	Password management (e.g., password change)
`session`	Session management (e.g., setting environment, mounting home, umask)

PAM Control Flag (control-flag)

Flag	Meaning
`required`	Module must succeed; if it fails, the entire stack fails (but other modules still execute)
`requisite`	Must succeed; if it fails, authentication immediately stops
`sufficient`	If this module succeeds, and no prior required module has failed, PAM succeeds
`optional`	Module's result is ignored unless it's the only module for this type
`include`	Includes another configuration file (e.g., `common-auth`)
`substack`	Includes another stack but treats it as part of the current one (used in modular configs)

PAM Shared Library (shared-library)

Module	Description
`pam_unix.so`	Uses local `/etc/passwd` and `/etc/shadow` for authentication
`pam_ldap.so`	LDAP-based authentication
`pam_google_authenticator.so`	2FA via Google Authenticator
`pam_env.so`	Set environment variables
`pam_faillock.so`	Lock account after failed login attempts
`pam_exec.so`	Execute an external script or program

Shared Library Options (module-options)

Examples for `pam_unix.so`:

Option	Purpose
`nullok`	Allow blank passwords
`try_first_pass`	Try the previously typed password first
`use_first_pass`	Only use the previously typed password (no prompt)
`shadow`	Use the shadow password file for verification

Shared Library Options (module-options)

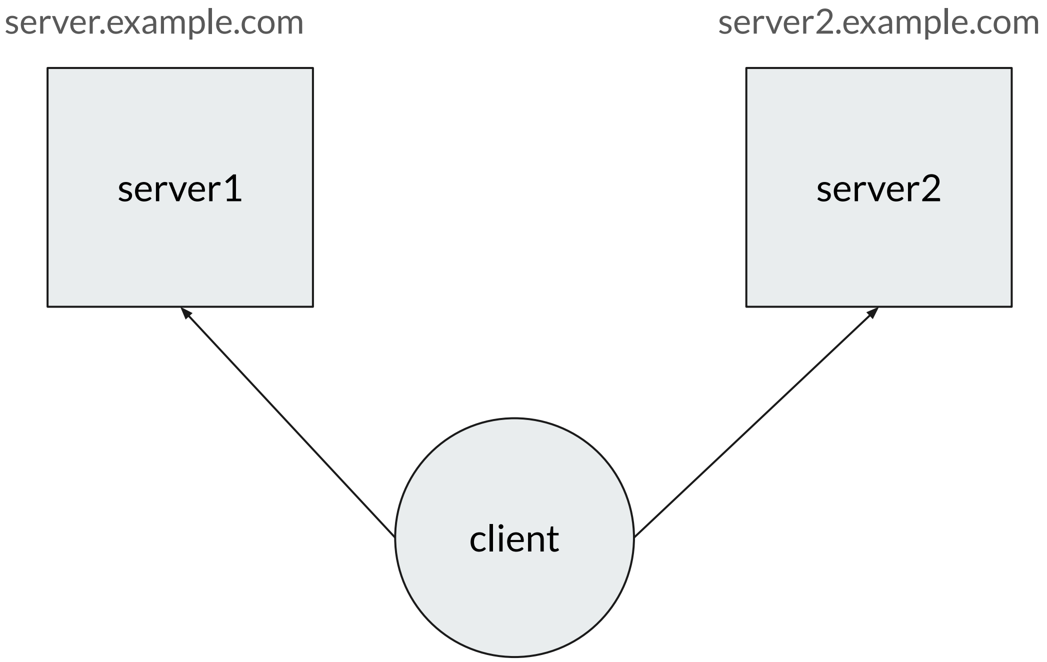
Examples for `pam_google_authenticator.so`:

Option	Purpose
`secret=/path`	Path to user-specific TOTP secret
`authtok_prompt`	Custom prompt for OTP entry

PAM Example (etc/pam.d/sshd)

```
#%PAM-1.0
auth        required      pam_unix.so
account     required      pam_unix.so
password    required      pam_unix.so
session     required      pam_unix.so
session     optional      pam_motd.so motd=/etc/motd
```

PAM Tutorial



username: testuser
password: testpassword

Thank you.