Presentation Specifications

Presentation title:Making Login Services Independent of Authentication Technologies

Created for:

Last modified: 1/11/96

Product family:

Contact: Charlie Lai, Vipin Samar

Contact's e-mail address: charlie.lai@eng, vipin.samar@eng

# of pages:

Source file name: SunSoft_Corp_Color_tmp.fm

BIP #: 3404

SunSoft

Making Login Services Independent of Authentication Technologies

Charlie Lai

Network Security Group

SunSoft

Authentication Requirements

- Users
  - Remember only one password
  - Do not want to learn about new authentication schemes

SunSoft

Authentication Requirements

- Vendors
  - System entry services independent of authentication technologies
  - Flexible, modular system to accommodate new authentication technologies

SunSoft

Authentication Requirements

- System Administrators
  - Ease of use to manage N authentication systems for M system entry services
  - Flexibility to have machine specific configurations
  - Flexible, modular system to accommodate new authentication technologies

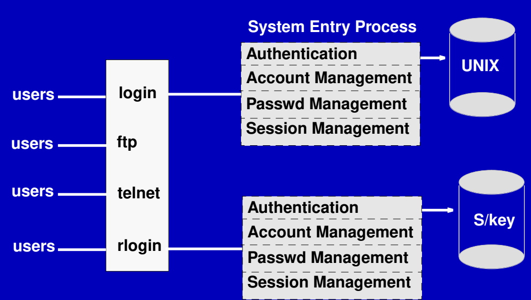
SunSoft

Control Flow of a Typical Login Service

get login id get password verify password	authentication
verify account validity	account management
change passwords	password management
update system accounts	session management

SunSoft

Current Architecture



SunSoft

Problems with Current Architecture

- All components of the system entry services are hard-coded
  - authentication
  - account management
  - password management
  - session management
- No easy way to integrate multiple authentication systems

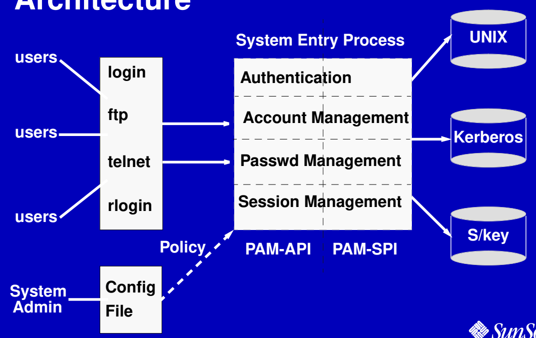
SunSoft

Problems with Current Architecture

- Authentication systems need to know about the system entry services
- Different system entry services may have different authentication needs
  - rlogin for trusted hosts
  - ftp does not allow multiple passwords
- Need separate services for graphical and non-graphical system entry

SunSoft

Pluggable Authentication Modules Architecture



PAM Configuration

service	module	control_flag	module_path	options
login	auth	required	pam_unix.so	
login	session	required	pam_unix.so	
login	account	required	pam_unix.so	nowarn
#				
rlogin	auth	required	pam_skey_auth.so	debug
rlogin	session	required	pam_unix.so	
#				
OTHER	auth	required	pam_unix.so	
OTHER	session	required	pam_unix.so	
OTHER	account	required	pam_unix.so	
OTHER	password	required	pam_unix.so	

Support for Multiple Modules

service	module	control_flag	module_path	options
login	auth	optional	pam_unix.so	
login	auth	required	pam_dce.so	use_first_pass
login	auth	required	pam_rsa.so	use_mapped_pass
#				
login	account	required	pam_users.so	allow=charlie
login	account	required	pam_unix.so	
#				
rlogin	auth	sufficient	pam_rhosts_auth.so	
rlogin	auth	required	pam_unix.so	
#				

Framework APIs

- pam_start(service, user name, callback, pam_handle)
  - Sets authentication transaction parameters
  - Specifies the callback function to read and write data
- pam_end(pam_handle, status)
  - Recovers PAM resources

Authentication APIs

- pam_authenticate(pam_handle, flags)
  - Authenticate the user
  - Invoke stacked modules specified in the PAM configuration file
  - All stacked modules are called to hide source of failure
  - Retry done by the application
- pam_setcred(pam_handle, flags)
  - Sets, refreshes, destroy's user credentials

Account Management APIs

- pam_acct_mgmt(pam_handle, flags)
  - Should this person be allowed to have a login session?
  - Checks password expiry, login hour restrictions, etc.

Session Management APIs

- pam_open_session(pam_handle, flags)
  - A new interactive session has started
  - Do any session processing
- pam_close_session(pam_handle, flags)
  - The session has terminated
  - Typically called by a different process (e.g. init)

Password Management APIs

- pam_chauthtok(pam_handle, flags)
  - Change the authentication token with all modules
  - The callback functions may be invoked to prompt for new authentication tokens

Example Login Service (login.c)

```
pam_start("login", username, &pam_conv, &pam_handle);
while (not authenticated && retry < 3)
    pam_authenticate(pam_handle, flags);
error = pam_acct_mgmt(pam_handle, flags);
if (error == PAM_AUTHTOK_EXPIRED)
    pam_chauthtok(pam_handle, flags);
pam_open_session(pam_handle, flags);
pam_setcred(pam_handle, flags);
pam_end(pam_handle);
```

Conversation Function for Login

```
login_conv(msg, response, appdata) {
    switch (message->msg_style)
    case PAM_PROMPT_ECHO_OFF:
        response = strdup(getpass(message->msg));
    case PAM_PROMPT_ECHO_ON:
        fputs(message->msg, stdout);
        fgets(response, 512, stdin);
    case PAM_ERROR_MSG:
        fputs(message->mst, stderr);
    case PAM_TEXTINFO:
        fputs(message->msg, stdout);
}
```

SunSoft

PAM Module (Unix Authentication)

pam_sm_authenticate()

```
{
    pamh->conversation();     /* prompt for user name */
    getpwnam();               /* check to see if user name valid */
    getspnam();               /* obtain encrypted password */
    pamh->conversation();     /* prompt for password */
    crypt();                  /* compare passwords */
    if (password incorrect)
        return (PAM_AUTH_ERR);
    else
        return (PAM_SUCCESS);
}
```

SunSoft

PAM Module Specific APIs

- pam_set_data(pam_handle, identifying_name, data, cleanup)
  - Stores module specific information that can be shared between different modules
  - Cleanup function invoked during pam_end()
- pam_get_data(pam_handle, identifying_name, data)
  - Returns the data stored by pam_set_data()

SunSoft

Status

- PAM available as a public API in a future release of Solaris™
- PAMified system entry services include
  - login
  - dtlogin
  - passwd
  - su
  - rlogind
  - telnetd
  - ftpd

SunSoft

Status

- Developed sample module implementations for
  - UNIX®
  - DCE
  - Kerberos
  - S/key
  - ruserok() based remote authentication
  - dialpass port based authentication

SunSoft

PAM and CDE

- Common Desktop Environment (CDE) Security Group tasked to solve the multiple authentication problem in early '95
- SunSoft's PAM solution selected in June, '95
- PAM sample implementation delivered to CDE in September '95
- Published OSF RFC 86.0 describing PAM in October '95
- CDENext will include PAM

SunSoft

Summary

- PAM provides a generic way to authenticate the user to system entry services
- PAM supports multiple authentication systems
- PAM supports machine and application specific policies
- PAM enables single-passwd login with multiple mechanisms
- Multi-vendor support
- Available as a public API in a future release of Solaris

SunSoft

Q&A

SunSoft