作者：陳柏菁　E-mail：pachingko@ms96.url.com.tw

第二十章　PAM 認證模組

索引：

20.1 認識 PAM

20.1.1 PAM 的簡介

PAM 全名為 **P**luggable **A**uthentication **M**odule，翻成中文就是所謂的 " 可插入式認證模組 "，可以簡單的將其視為一個應用程式介面 (API)，而在整個認證程序中，PAM 即是用來擔任使用者與應用程式之間的中介角色。

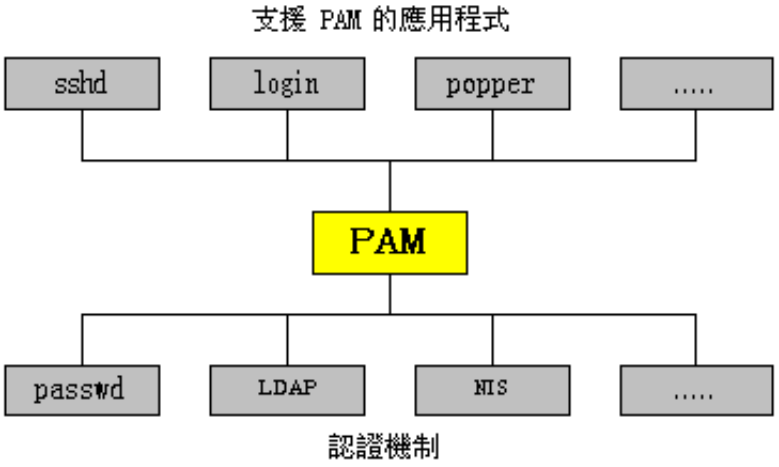
以整個認證體系而言，PAM 算是認證系統的前端，而認證機制 (即真正要採取認證的方式，比如根據 passwd file 或 LDAP 等來做認證) 則是屬於認證系統的後端。

在傳統的 UNIX 上頭，PAM 是應用程式經常使用的一種認證方式，不過您可不要以為 PAM 只能提供認證的服務噢，它還能提供帳號管理、使用者於系統活動期間的紀錄管理、控管使用者對服務的存取管理等等。

這裡舉個 PAM 認證的例子來說好了 ；當 client 端的 user 要使用 SSH 做遠端連線時，如果採用的是密碼認證的話，那就需先輸入驗證資料，而 Sever 端在收到這個認證資訊時，sshd 會呼叫 PAM 來處理，並根據 /etc/pam.d/sshd 檔案內的設定，來決定使用哪些認證模組做確認的工作，而在認證模組處理到一個段落後，會適時的回應給 sshd。由以上的例子說明，不難發現 sshd 本身必須支援 PAM 認證才行，也就是說 sshd 程式本身的原始碼裡頭必須包含 PAM 函式。您可以使用 ldd 指令來查閱一下就知道了：

```
suse:~ # ldd /usr/sbin/sshd
        linux-gate.so.1 => (0xffffe000)
        libwrap.so.0 => /lib/libwrap.so.0 (0x40031000)
        libpam.so.0 => /lib/libpam.so.0 (0x40039000)
        libdl.so.2 => /lib/libdl.so.2 (0x40041000)
        libkafs.so.0 => /usr/lib/libkafs.so.0 (0x40044000)
```

如您還不太清楚 PAM 所扮演的角色，請參考以下這個架構圖再配合上面的說明來看，應該就會有概念了：



圖一：PAM 的架構圖

至於為何要取名為可插入式的認證模組呢？因為它是一組隨時可讓管理員來彈性使用的認證函式庫，比如您只希望有少數使用者可以執行 su 指令來切換成其他 user 或 root 的身份，那麼可以在 /etc/pam.d/su 檔案中去引用 pam_wheel.so 模組，而當您想取消此項功能時，就直接將此模組拿掉即可，如此一來，我們就不必為了達到某個特定目的而去重新編譯 su 程式了。看到這裡，大家是不是覺得使用這個 PAM 真的是非常方便呢？

20.1.2 PAM 的相關檔案

上一小節的觀念沒問題的話，緊接著就來說明一下與 PAM 有關的幾個相關檔案介紹：

- **/lib/libpam*.so**：

  支援 PAM 的應用程式，需使用這裡所提供的 PAM 函式，來存取相關的模組設定檔。不過有的 Distribution 則是將 PAM 函式安置在 /usr/lib 目錄中。

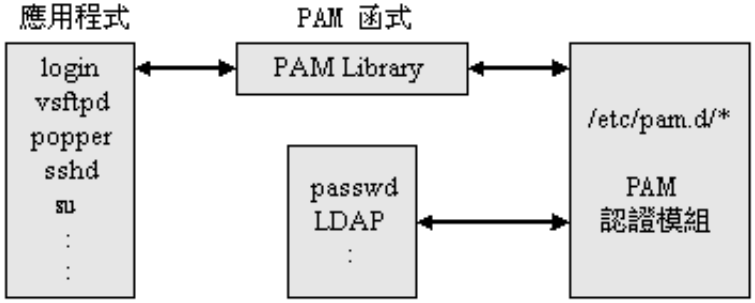
- **/etc/pam.d/***：

  PAM 認證模組設定檔就是放在這個目錄下。我們需要為支援 PAM 的應用程式安置一個檔案在裡頭，至於檔名則是寫死在程式原始碼裡面。通常這裡的檔名會取一個與程式本身的名稱相關或相同，所以平常您千萬不要隨意去更改檔名，因為萬一找不到 PAM 模組設定檔時，就會套用 /etc/pam.d/other 檔案裡的設定內容。
  另外早期的 PAM 模組設定檔是安置在 /etc/pam.conf 這個檔案中，而目前則大多是安置在 /etc/pam.d 目錄裡。

- **/lib/security/*：**
  PAM 所使用的認證模組就是安置在這裡，比如 pam_unix2.so、pam_time.so、pam_nologin.so 等等。

- **/etc/security/*：**
  此目錄中，存放著大部分 PAM 模組的全域組態檔。至於詳細的情形後面的範例會有說明。

了解了相關檔案的配置之後，我們再以 PAM 認證的詳細流程圖來跟各位說明：



圖二：PAM 的認證流程

在圖二的流程圖中，是說明應用程式會先去呼叫 PAM 函式，然後再由其去讀取 /etc/pam.d/ 目錄下的 PAM 模組設定檔，並依序載入檔案中所指定的 PAM 模組，而這些模組是以在設定檔中所設定的順序依序被堆疊起來 ；接著這些堆疊模組就開始為應用程式執行不同的認證工作，那當然執行結果或許成功，或許失敗，不過不論成功或失敗，都需要將執行結果回應給 PAM 函式。假使說第一個 PAM 模組驗證失敗後，那麼會回應給 PAM 函式知道，至於下一個 PAM 模組會不會接著被 PAM 函式載入，則是要依照剛剛上一個 PAM 模組所設定的控制旗標來決定 (下一節會說明)，而等 PAM 模組都驗證完畢後，PAM 函式才會將失敗結果傳回給應用程式。

> **Tips**：這裡來補充一下模組堆疊的概念。使用 PAM 之時，管理者往往會希望不是只單純的檢查使用者帳號密碼而已，可能還希望能進一步的作各項控管，比如開放某些服務只能在規定的時間內才能存取，或者部分服務只有少數使用者能夠存取等等，此時管理者就可以根據不同的需求來引用 (插入) 不同的模組，那當然不同的模組所執行的檢查工作也就不同囉。比如以同一個模組類型來說，當模組 1 檢查完後，可能換模組 2 接著進行其他的檢查，模組 2 檢查完畢後也可能會由模組 3 再進行檢查，這就是利用模組堆疊 (statcked modules) 的概念來驗證用戶的合法性。

20.2 PAM 模組設定檔

20.2.1 PAM 模組設定檔的設定格式

這部分主要是探討 /etc/pam.d/ 目錄下的 PAM 模組設定檔的內容：

```
suse:~ # ls /etc/pam.d
atd             common-password  gnome-passwd       pop3       sshd
chage           common-session   gnome-screensaver  ppp        su
chfn            crond            gnomesu-pam        pure-ftpd  sudo
chsh            cups             login              samba      useradd
common-account  gdm              other              shadow     vsftpd
common-auth     gdm-autologin    passwd             smtp       xdm
```

先簡單了解一下這些檔案的設定格式：

modules_type	control_flags	modules_path	options
模組類型	控制旗標	模組路徑	傳給模組的參數

其中除了 options 欄位外，其餘都是必要設定的欄位。底下分別來對這幾個欄位做相關說明：

模組類型

這部分總共區分成四種 modules_type：

模組類型	主　要　功　用
auth	對使用者所提供的認證資訊做驗證，比如檢查帳號密碼。
account	非認證方面的帳號管理，比如檢查使用者帳號是否已過期、哪些帳號可從哪些來源端來存取服務、及限定存取服務的時間等等。
password	與更新認證訊息有關，比如使用者變更密碼時會檢查新密碼是否夠安全。
session	這個模組類型裡所使用的模組，是與使用者在存取服務前後需執行的一些工作有關聯。比如紀錄掛載目錄的資訊、使用者做資料交換時的訊息紀錄等等。

控制旗標

一個認證模組在執行認證檢查時，有可能成功 (success)，也有可能失敗 (failure)，當其傳回 success 或 failure 時，PAM 將會採取什麼動作？是

要繼續執行下一個模組認證 (模組堆疊的特性)，還是要結束認證工作不繼續往下，這就要看您是採用什麼 control_flags 了。所以簡單的說，control_flags 是用來告知 PAM 在模組執行成功或失敗後，要不要繼續進行其它堆疊模組的驗證。底下就來說說幾個主要的控制旗標：

控制旗標	影 響 模 組 執 行 時 的 行 為
required	當使用此旗標的其中一個認證模組傳回 failure 時，還是會繼續往下執行其他堆疊模組的認證 (相同的 module_type)，等全部都執行完畢後，PAM 才傳回 failure。那如果認證模組傳回 success 時，也還是會繼續往下執行其他堆疊模組的驗證。 或許您會想說既然已經傳回失敗了，為何還要繼續往下去執行下一個 PAM 模組呢？如此不是多此一舉嗎？其實這樣做的目的是為了安全上的考量，也就是說透過堆疊模組這樣的設計，可以讓用戶端永遠不知道他是如何被拒絕的，因為如果當用戶了解其被拒絕的理由後，可能就會特別針對這方面來做進一步的突破，而這將會造成對系統安全上的一大威脅。
requisite	當使用此旗標的認證模組傳回 failure 時，則會終止整個認證程序，而不再往下驗證，PAM 會馬上傳回一個 failure 給應用程式。那如果傳回 success ，則還是會繼續往下執行其他堆疊模組的驗證。 一般人往往分不清楚 required 與 requisite 的差異，簡單的說，不論是 required 或 requisite，只要傳回 success，都還是會繼續往下驗證，只有當它們都傳回 success 時，才算是真正的 success。而如果 required 傳回 failure 時，是會繼續執行其他堆疊模組的驗證，但 requisite 傳回 failure 時，則是會終止整個驗證程序。
sufficient	以相同的 modules_type 來說，如果之前的 required 都傳回 success (或者之前根本沒有 required)，而使用此旗標的認證模組也傳回 success 時，則不再繼續往下執行此 modules_type 中的其他堆疊模組，並傳回 success 給　PAM 函式。不過這並不意味著整個認證就是成功的，萬一有其他 modules_type 的認證模組使用 required 旗標並傳回 failure 時，那還是算認證失敗。另外不論當 required 是傳回 success 或 failure 時，sufficient 的 failure 都會被當成 optional 來看待，而被忽略。
optional	使用此旗標的認證模組，無論傳回 success 或 failure，都不會影響整體的認證結果，其主要是用來做 log 而已。
include	這算是比較特殊的控制旗標，主要是用來引入其後所接的檔案 (include file)，至於檔案內容就是傳統的那四個欄位囉。另外使用 include 所含括的檔案內容，是會在 include 的那個位置被插入的。

模組路徑

如果這個欄位只設定模組名稱，未指定其實際路徑，則表示此模組是被安置在預設的 /lib/security 目錄中。至於在 include 旗標後的 include file，則是被放置在 /etc/pam.d 目錄中。

模組參數

這裡舉幾個模組參數給大家參考一下：

模組參數	主　要　作　用
use_first_pass	認證模組不會要求使用者輸入密碼，而是從之前的 auth modules 來取得使用者密碼，若密碼不符合或未輸入密碼，則視為認證失敗。
try_first_pass	認證模組不會要求使用者輸入密碼，而是從之前的 auth modules 來取得使用者密碼，若密碼不符合或未輸入密碼則要求重新輸入一次。
debug	讓 syslogd 將 level 為 debug 等級的資訊寫入紀錄檔內。
nullok	允許無密碼 (/etc/shadow 的密碼欄位是空的) 的使用者可以登入系統。

20.2.2 PAM 認證模組的功能

PAM 模組是存放在 /lib/security 目錄中：

```
suse:~ # ls /lib/security
pam_access.so     pam_lastlog.so     pam_permit.so       pam_unix.so
pam_chroot.so     pam_limits.so      pam_pwcheck.so      pam_unix2.so
pam_cracklib.so   pam_listfile.so    pam_resmgr.so       pam_unix_acct.so
pam_debug.so      pam_localuser.so   pam_rhosts_auth.so  pam_unix_auth.so
pam_deny.so       pam_mail.so        pam_rootok.so       pam_unix_passwd.so
pam_devperm.so    pam_make.so        pam_rpasswd.so      pam_unix_session.so
pam_echo.so       pam_mkhomedir.so   pam_securetty.so    pam_userdb.so
pam_env.so        pam_mktemp.so      pam_shells.so       pam_userpass.so
pam_filter        pam_motd.so        pam_smbpass.so      pam_warn.so
pam_filter.so     pam_ncp_auth.so    pam_stress.so       pam_wheel.so
pam_ftp.so        pam_nologin.so     pam_succeed_if.so   pam_winbind.so
pam_group.so      pam_opensc.so      pam_tally.so        pam_xauth.so
pam_homecheck.so  pam_opie.so        pam_time.so
pam_issue.so      pam_passwdqc.so    pam_umask.so
```

由於實在是太多了些，因此這裡就挑幾個 PAM 認證模組的功能來說明囉：

認證模組名稱	主　　要　　用　　途	載 入 檔 案	搭配的模組類型
pam_access.so	可限制特定的使用者及群組，從哪個特定的來源端登入。	access.conf	account

pam_cracklib.so	檢查使用者密碼是否夠安全。此模組可搭配的 options 如 difok、retry、minlen、dcredit、ocredit 等。		password
pam_deny.so	拒絕一切存取。當使用這個模組時，第四個欄位的 options 會被忽略。		皆可
pam_env.so	用以載入額外的環境變數來使用。	pam_env.conf	auth
pam_limits.so	限制使用者所能使用的資源，如 CPU 使用時間、user 能重複登入的次數、user 能開啟的檔案數等等。	limits.conf	session
pam_listfile.so	此模組會依您所做的設定及所指定的檔案，來決定允許或拒絕使用者存取該服務。		auth
pam_mail.so	當使用者有新的信件進來時，會在該使用者登入時告知這個訊息。	/var/spool/mail/*	auth session
pam_nologin.so	此模組會檢查 /etc/nologin 是否存在，如果存在的話，則會拒絕所有一般使用者登入 (包含遠端登入)，並顯示 /etc/nologin 檔案中的訊息內容。	/etc/nologin	auth、account
pam_permit.so	允許一切存取。		皆可
pam_pwcheck.so	當使用者變更密碼時，此模組會執行額外的檢查工作。	pam_pwcheck.conf	password
pam_rootok.so	此模組如設定在 /etc/pam.d/su 檔案中，是表示允許 root 在執行 su 指令來切換成其他 user 的身分時，可以不需輸入密碼。		auth
pam_securetty.so	限定 root 可從哪幾個安全的終端機登入。	/etc/securetty	auth
pam_time.so	依照 /etc/security/time.conf 的設定，來限制使用者可以在什麼時間登入哪幾個終端機。	time.conf	account
pam_unix2.so	此模組可用來檢查使用者所提供的帳號密碼是否正確或過期等，端看我們所設定的模組類型為何，而有不同的功用。		皆可
pam_wheel.so	此模組設定在 /etc/pam.d/su 檔案中時，表示只允許 wheel 群組的成員，可以執行 su 指令來變換成其他 user 的身份。		auth

Tips：上面表格中的 [載入檔案] 欄位，如未指出路徑，即是存在於 /etc/security 目錄中。另外如您想找更多的說明文件，請自行查閱 /usr/share/doc/packages/pam/modules。

20.3 PAM 的設定範例

看了前面兩節的介紹後，可能還是有點模模糊糊的，不過沒關係，底下會舉幾個模組設定檔的範例來加以解說，這樣應該就可以對 PAM 認證模組有更進一步的認識。

範例一：/etc/pam.d/login

```
suse:~ # cat /etc/pam.d/login
#%PAM-1.0
auth       required     pam_securetty.so
auth       include      common-auth
auth       required     pam_nologin.so
account    include      common-account
password   include      common-password
session    include      common-session
session    required     pam_lastlog.so nowtmp
session    required     pam_resmgr.so
session    optional     pam_mail.so standard
```

這是 login 程式所使用的模組設定檔。先來說說檔案中的四個 include 敘述。

在一般 PAM 模組設定檔裡，可以使用 include 敘述來分別含括四個檔案進來，以個別提供給四個 module type 使用 ；比如 module type 為 auth 時所使用的 common-auth 檔案，account 所使用的 common-account 檔案，password 使用的 common-password 檔案，session 使用的 common-session 檔案，而這些 include file 掌控著各 module type 的預設組態。

至於使用 include 的好處是，管理者只需藉由修改 include file 裡的預設組態，就能自動更新所有的 PAM 模組設定檔，而不必為每個應用程式來一一調整其模組設定檔內容。

為了方便解釋，我們把 /etc/pam.d/login 這個檔案裡的 include file 內容取出，並在該位置進行插入，因此檔案內容看起來就像這個樣子：

```
suse:~ # vi /etc/pam.d/login
#%PAM-1.0
auth       required     pam_securetty.so
auth       required     pam_env.so      # 這是 common-auth 內容
auth       required     pam_unix2.so    # 這是 common-auth 內容
auth       required     pam_nologin.so
account    required     pam_unix2.so    # 這是 common-account 內容
```

```
password   required     pam_pwcheck.so nullok
password   required     pam_unix2.so nullok use_first_pass use_authtok
# 以上兩行是 common-password 內容
session    required     pam_limits.so    # 這是 common-session 內容
session    required     pam_unix2.so     # 這是 common-session 內容
session    required     pam_lastlog.so nowtmp
session    required     pam_resmgr.so
session    optional     pam_mail.so standard
```

首先 auth 這個模組類型皆使用 required 旗標，此乃意味著要等所有 auth 的堆疊模組都成功處理完畢後，login 才會收到 success 的訊息，那萬一有某個模組的處理是失敗的，則還是會繼續呼叫其它的堆疊模組來進行，執行結束後才會通知 login 失敗的結果。

假使 auth 的堆疊模組都已成功處理後，接著就會執行 account 類型的模組了，由於 account 只包含一個模組，因此只要此模組也成功處理完，那麼 login 就會收到 success 訊息，並緊接著呼叫 password 類型的模組，餘依此類推囉。

另外在這個檔案裡，您可能注意到四個 modules type 中，都有個叫 pam_unix2.so 的模組，不過此模組運用在不同的 module type 時，所要執行的工作可是不相同的。當 pam_unix2.so 運用在 auth 中時，是用來檢查使用者的登入帳號及密碼是否正確 ；運用在 account 中時，是檢查使用者所提供的密碼是否有效，或者需要再建立新的密碼 ；運用在 password 中時，是把使用者所修改的密碼進行加密後寫入認證資料庫裡 ；運用在 session 中時，則是告知 syslogd 將使用者的登入資訊紀錄下來。

了解了 pam_unix2.so 的大致用途後，現在做個修改，把第三筆的控制旗標由 required 改成 sufficient：

```
auth       required     pam_securetty.so
auth       required     pam_env.so
auth       sufficient   pam_unix2.so
auth       required     pam_nologin.so
```

此時登入的使用者，縱使輸入錯誤的密碼，還是可以順利登入，因為使用 sufficient 旗標的認證模組所傳回的 failure 會被忽略掉。因此控制旗標的使用可要小心啊，不然可是會一失足成千古恨的。

那如果把剛剛那筆設定的 required 改成 requisite 又如何呢？這就表示 pam_unix2.so 模組若是傳回 failure，則不會再往下執行其他的堆疊模組，而是直接宣告失敗。但如果是傳回 success， 則還是會繼續往下執行其他的堆疊模組。 只有當 requisite 及 required 都傳回 success，整個 auth 的處理才算成功。

再來就分別針對幾個 PAM 模組做個測試練習，藉由實做的演練，能讓您對 PAM 有更進一步的認識：

- 測試 **pam_securetty.so** 模組

  這個模組主要參考的檔案為 /etc/securetty，是用來決定 root 可以從哪幾個安全的終端機登入。作個實驗吧！首先請您打開 /etc/securetty 檔案，並把 tty6 那一行註解起來：

```
suse:~ # vi /etc/securetty
tty1
tty2
tty3
tty4
tty5
# tty6 → 將該行註解起來。
```

  存檔離開後，請切換至 tty6，並使用 root 身分登入看看，結果如何呢？應該是會無法登入對吧！因為此終端機已經不是安全的終端機了。如果現在換成以一般 user 的身分登入，則是不會受到這裡的限制。

- 測試 **pam_nologin.so** 模組

  此認證模組只要發覺系統上存在一個 /etc/nologin 檔案，就會拒絕除 root 外的其它一般使用者登入。

```
suse:~ # touch /etc/nologin
```

- 測試 **pam_limits.so** 模組

  此模組會參照 /etc/security/limits.conf 的設定，來限制 user 的使用資源。請參考以下範例：

```
suse:~ # vi /etc/security/limits.conf
#domain type    item      value
@users     -   maxlogins    2

# 簡單介紹一下這四個欄位：
# domain：可以是使用者或群組名稱，當設定群組時，請記得在前面加上個 "@" 符號。
# type：可設定項目有 soft、hard 及 -，其代表的是 soft limit、hard limit 限制。
# item：為限定存取資源的項目，如 core、data、nproc、maxlogins、... 等等。
```

value：針對 item 所設定的值。

以這個範例來說，是要限制 users 群組的成員，最多只能同時登入兩次 (含遠端登入)。比如使用者 barry 為 users 群組成員，目前已登入 tty1 及 tty2，則其無法再由其他的終端機登入。

- 測試 **pam_time.so** 模組

此模組所參考的檔案為 /etc/security/time.conf，可用來限定使用者登入 tty 的時間。至於 time.conf 的設定格式如下：

```
services;ttys;users;times
```

接著分別來說明這幾個欄位的意義：

欄 位	意　義
services	設定服務名稱。
ttys	設定所使用的終端機。
users	所限定的使用者名稱列表。
times	限定存取的時間。至於時間格式的寫法，週一至週日分別為 Mo、Tu、We、Th、Fr、Sa、Su，而 Al 是表示週一至週日的每一天，AlWe 則是除了週三以外的每一天。另外您還需搭配確切的存取時間來使用，比如 "MoWeFr0800-1200" 是每週一三五的早上八點至十二點，"!Al0000-2400" 則是表示所有時間都禁止的意思。

Tips：往往您還可以搭配萬用字元 " * " 及邏輯運算符 **!** (NOT)、| (OR)、**&** (AND) 於其中。

以上如果沒問題的話，馬上舉個例子來練習看看。首先先於 /etc/pam.d/login 檔案中插入 pam_time.so 模組：

```
suse:~ # vi /etc/pam.d/login
 #%PAM-1.0
                :略
account    include      common-account
account    required     pam_time.so ←引用 pam_time.so 模組。
                :略
```

再至 /etc/security/time.conf 中做設定：

```
suse:~ # vi /etc/security/time.conf
login;tty3|tty4|tty5;tina|david;MoWe0900-2200
# 允許 tina 或 david 在週一及週三的早上九點至晚上十點登入本機的第三、四、五
# 個終端機，至於其他時間則禁止登入。

login;tty*;user*;!Al0000-2400
# 使用者的帳號名稱是以 user 做開始者 (如 user01、user225 等)，
# 則無論什麼時候都禁止登入所有的終端機。
```

最後如果您想禁止所有一般使用者登入本機，可以這麼設定：

```
suse:~ # vi /etc/security/time.conf
login;tty*;!root;!Al0000-2400
```

範例二：**/etc/pam.d/pop3**

```
suse:~ #  vi /etc/pam.d/pop3
#%PAM-1.0
auth    required  pam_unix2.so
auth    required  pam_listfile.so  onerr=succeed item=user sense=deny file=/etc/poplist
account required  pam_unix2.so
```

這是與收信服務相關的模組設定檔。而這裡引用了 pam_listfile.so 模組，它可以用來載入其後所指定的檔案 /etc/poplist，以決定要允許或拒絕使用者存取 pop3 服務。

另外 "onerr=succeed" 的意思是說，如果 /etc/poplist 檔案不存在或無法開啟，驗證模組就把它當作是成功的 ；反過來說，如果希望 /etc/poplist 檔案不存在時，就當成驗證失敗來處理的話，那設定成 "onerr=fail" 即可。

再來是 "sense=deny"，其表示存在於 /etc/poplist 中的使用者是無法存取 pop3 服務的，至於其他使用者則是會被允許。一樣道理，如設定成 "sense=allow"，則只有 /etc/poplist 中的使用者可以存取 pop3 服務，其他人則會被拒絕。

假使依照目前範例的設定，想要拒絕使用者 barry 及 mary 存取 pop3 服務，那麼就把這兩個 user 寫入 /etc/poplist：

```
suse:~ #vi /etc/poplist
barry
mary
```

夠簡單吧！沒問題的話，再來看個類似設定的檔案 /etc/pam.d/vsftpd：

```
suse:~#vi /etc/pam.d/vsftpd
#%PAM-1.0
auth       required   pam_listfile.so onerr=succeed item=user sense=deny file=/etc/ftpusers
auth       required   pam_unix2.so
auth       required   pam_shells.so
account    required   pam_unix2.so
password   required   pam_unix2.so
session    required   pam_unix2.so
```

第一行的設定應該看得懂了吧！簡單的說，就是設定在 /etc/ftpusers 檔案中的使用者，無法存取 ftp 服務啦。還記得當初在學習 FTP Server 有提過，可以把要拒絕登入的使用者帳號設定在這個檔案內，現在應該恍然大悟了吧！

範例三：**/etc/pam.d/sshd**

```
suse:~ # vi /etc/pam.d/sshd
#%PAM-1.0
auth       include    common-auth
auth       required   pam_nologin.so
account    include    common-account
account    required   pam_access.so   ← 引入 pam_access.so 模組。
password   include    common-password
session    include    common-session
```

這裡主要是針對 pam_access.so 來做說明。此模組預設會載入 /etc/security/access.conf 檔案，來進一步允許或拒絕哪些帳號能從哪些特定的來源端登入。以下接著就介紹 access.conf 這個檔案：

```
suse:~# vi /etc/security/access.conf

# 檔案設定格式如下：
# permission : users : origins
# permission 可以設定成 " + " (允許存取) 或 " - " (拒絕存取)。
# users 可以是使用者名稱、群組名稱或所有人 (ALL)。
# origins 則為來源端的設定，可以是主機名稱、網域名稱 (.domain.com.tw)、主機
# 位址、網路位址 (192.168.1.)、所有來源端 (ALL) 及不含 " . " 的主機名稱 (LOCAL)。

- : user01 user02 : ALL EXCEPT .paching.com.tw 192.168.1.
- : tina david    : LOCAL 192.168.2.
- : ALL EXCEPT group01 group02 : 192.168.1.222
```

第一行設定是說，user01 及 user02 這兩個使用者，除了從來源端為 xx.paching.com.tw 及 192.168.1.x 以外，拒絕從其他來源端存取 SSH 服務。

第二行設定是說，不允許 tina 及 david 從不含「.」的來源主機名稱或 192.168.2.x 的來源端存取 SSH 服務。

最後一行的設定是說，除了 group01 及 group02 的群組成員外，禁止其他人從 192.168.1.222 的來源端存取 SSH 服務。

如果現在您只希望能讓某個特定的使用者來存取 SSH 服務，其他人全部拒絕 (包括 root 在內)，那麼可以這麼設定：

```
suse:~# vi /etc/security/access.conf

- : ALL EXCEPT barry : ALL
```

再來參考下個範例：

```
suse:~# vi /etc/security/access.conf

+:root:192.168.1.1
+:barry:all
-:all:all
```

這個範例的設定是說，只允許管理者從 192.168.1.1 的來源端登入，而使用者 barry 能從所有來源端登入，至於其他人則全部會被拒絕。

另外利用範例二的 pam_listfile.so 來限制服務的存取，也是不錯的選擇：

```
suse:~ #  vi /etc/pam.d/sshd
#%PAM-1.0
auth       include    common-auth
auth       required   pam_nologin.so
auth       required   pam_listfile.so onerr=succeed  item=user sense=allow file=/etc/sshallow
              ：略                        ：略
```

這樣就只有 /etc/sshallow 檔案中的使用者，才能存取 SSH 服務。

範例四：**/etc/pam.d/su**

```
suse:~ #vi /etc/pam.d/su
#%PAM-1.0
auth       sufficient  pam_rootok.so
auth       include     common-auth
account    include     common-account
password   include     common-password
session    include     common-session
session    optional    pam_xauth.so
```

先看看 auth 第一筆的設定，在這裡因為引用了 pam_rootok.so 模組，所以當 root 執行 su 指令來變換成其他 user 的身分時，不需輸入密碼即可轉換成功。

不過您有沒有注意到第一筆是採用 sufficient 控制旗標，這就表示說當 pam_rootok.so 傳回 success 時，就不再呼叫 auth 中其他的堆疊模組。現在如果把 sufficient 改成 required 又如何呢？此時當然是會要求 root 輸入密碼囉，因為使用 required 旗標的模組，無論傳回 success 或 failure，都還是要呼叫其它的堆疊模組來繼續處理。

- 測試 **pam_wheel.so** 模組

當您要限定一般使用者執行 su 的話，就可以引入這個模組，請參考以下的範例：

```
suse:~ #vi /etc/pam.d/su
#%PAM-1.0
auth       sufficient   pam_rootok.so
auth       include      common-auth
auth       required     pam_wheel.so      group=wheel
account    include      common-account
password   include      common-password
session    include      common-session
session    optional     pam_xauth.so
```

這裡是引用了 pam_wheel.so 模組，其後是指定可使用 su 指令來轉換身分的群組名稱，預設是 wheel 群組，當然您也可以指定其他的群組，這樣一來，就只有是該群組的成員才可以執行 su 囉。自己測試一下吧。

PAM 認證模組就介紹到這裡了。不過憑良心說，您要是初次接觸 PAM 的話，會覺得好像不是那麼容易入門，但偏偏它對系統安全方面的管理又非常有幫助，因此多花點時間在這上頭學習是有其必要的。