

Computer Smiths PAM Modules  
Installation and Instructions  
Version 1.06  
Revision R

Release: 30<sup>th</sup> May 2011

## Introduction

This document provides instructions on how to install and configure the PAM modules supplied. These instructions are designed for administrators with experience of basic Solaris administration commands.

---

### **WARNING**

**This software provides for enhanced security on a system, but if used incorrectly may lead to a reduction in some security. Please ensure that the limitations of the software, found in the section titled *Limitations and Issues* are understood before adding this software to any critical systems.**

---

## Limitations and Issues

As with all security related functionality an understanding must be obtained as to the interrelationships and dependencies of the software. Also we must understand the vectors that information may leak, or be obtained from the product.

All the PAM modules provided and described in this document use and depend upon the PAM framework. They also use and depend upon the password framework. The various modules obtain password information using the `getpwnam_r` library call. This information is used in subsequent calls. If the PAM infrastructure is compromised, or if the `getpwnam` and related infrastructure, including `nsswitch/NIS/NIS+/LDAP` etc. or the password files, return bad information then these modules may be compromised.

Syslog messages are different on success or failure, so password attempts may be verified depending on the length of the messages file. Or messages may be hijacked or snooped if they traverse a network.

Further individual authentication modules process data in different ways, and may allow an attacker to discover success or failure modes.

Login limit will always update the `/var/adm/login_limit` file on success, but may not do so on failure. Password history will update the `/var/adm/passwd_history` file on success but may not do so on failure. The password history file holds previously used passwords, albeit in crypt format.

Login times is only checked at login. User jobs may continue to run and perform actions even at times no logins are allowed.

Login limit will call the `/bin/passwd` program to lock an account. There is a known Denial Of Service attack methodology that involves multiple knowingly incorrect password attempts with the goal being the locking of the accounts, not penetration. It is recommended that login limit be teamed with TCP wrappers or the remote hosts PAM module to defend against this type of attack.

Finally if a service does not use PAM then these modules do not provide protection. The most common example of this is a default installation of OpenSSH which does not use the PAM framework for authentication.

## System Requirements and Availability

The modules are supplied as a set of installable packages. These packages are added using the *pkgadd* system command.

The software uses approximately 100K of disk space per module added.

Modules are provided for SPARC Solaris 7, 8, 9, 10 and SPARC Solaris 11 Express and Solaris x86 10 and 11 Express. The layout of the installation media has the packages in different directories for each of the architectures and releases.

The directory layout is */architecture/release* and in the directory there is a series of data stream packages. These packages and their uses are:

COMSpamll	Login Attempt Limiter
COMSpamlt	Login Times Limiter
COMSpamrh	Remote Login Limiter
COMSpamph	Password History Limiter
COMSpamnl	PAM Null Module
COMSpamch	Chroot on Login

The modules are independent, so only those desired need to be installed. Note that not all modules may be present for all architectures and releases.

# Installation

Installation is performed by adding the modules required using the `pkgadd` command, and then configuring any desired functionality in the `/etc/pam.conf` control file.

If the modules are supplied on CDROM insert the CDROM into the drive and have vold automatically mount it, or mount the CDROM manually from a local or remote location. If the modules were downloaded over the Internet uncompress the modules if needed. As root run the following commands

```
# cd <path>/architecture/release
# gunzip <package file name>.gz      (if needed)
# pkgadd -d <package file name>
```

The software installs files into the following locations:

```
/usr/lib/security
/usr/local
/var/adm
```

The shared object modules are installed into `/usr/lib/security`. Manual pages and administration programs are installed into `/usr/local` and data files are installed into `/var/adm`.

After adding the packages the system wide `/etc/pam.conf` file has to be updated. The actual updates required vary depending on the functionality required and the modules installed. Each module provides a man page that describes the modifications suggested for that module. These may be accessed by the command

```
# man -M /usr/local/man pam_module_name
```

where `pam_module_name` is one of `pam_login_limit`, `pam_login_times`, `pam_remote_hosts`, `pam_history`, `pam_chroot`, or `pam_null`.

If `/usr/local/man` is in the standard `MANPATH` then the `-M` option is unnecessary.

Details on the configuration and options for each module are also provided in the configuration section for each module in this document.

## Configuration

Each module requires different configuration to achieve optimal performance. It is assumed that the audience has a basic understanding of PAM functionality.

In general a module installs into one section of the authentication structures and provides functionality for that section. Services that wish to use the functionality have configuration lines for that module in the appropriate section.

The sections that a module provides functionality for are:

login_limit	auth, (account), (password)
login_times	auth
remote_hosts	auth
password history	password
null	auth, account, session, password
chroot	session

Multiple modules may be added for a service, using the stacking rules provided for in the PAM infrastructure.

## Login Limit (COMSpamll)

The login limit module provides functionality to limit the number of failed logins on a per user basis. It keeps a count of the number of failed logins and when that number exceeds a defined limit the account is either locked or subsequent attempts fail for that user for either a defined time, or until the count is reset. The count is kept on a per machine basis, but locking is done globally.

The module understands a number of options that may be specified in pam.conf: (Please see the section on common options for further details)

<code>debug</code>	Turn on additional syslog debugging messages.
<code>silent</code>	Turns off Info/error/notice/crit syslog messages.
<code>nowarn</code>	Turns off any warning messages for the user.
<code>check_root</code>	Check the root user. By default the root user is partially checked. Specifying this option causes root logins to be subject to the normal checks.
<code>count_limit=<i>limit</i></code>	Override the default count limit. By default a user can supply three wrong passwords. On the third incorrect password the account will be disabled. How the account is disabled is dependent upon the <code>lock_account</code> option.
<code>lock_account</code>	By default the login limit module will return failure for any attempt after the third incorrect password. With this option specified the module will instead fork <code>passwd -l <u>username</u></code> unless a different command was specified by <code>lock_cmd</code> . The command should lock the account.
<code>timeout_account=<i>time</i></code>	When the number of incorrect passwords is exceeded the account cannot be used for this timeout period. Any access of the account during this period will restart the timeout period.
<code>lock_cmd=<i>cmd</i></code>	When <code>lock_account</code> is specified this option will change the command executed. The username is automatically added as an option to the command specified.
<code>file=<i>file</i></code>	This option overrides the default count file.
<code>check=<i>check</i></code>	Changes the checks that are performed on the count file to ensure it has not been tampered with. See <b>Common Options</b> for a full explanation of the checks that are possible.

The module will consult the `/var/adm/login_limit` count file. This file is a sparse binary database that holds an entry for each user. The entry contains the number of unsuccessful attempts, and the date of the last attempt. This file may be manipulated by the `login_limit` program.

In general three different behaviours are possible using the login\_limit module. These are:

- disable the account until the password is reset
- disable the account for a timeout period
- disable the account until the password is reset, or the lock is cleared.

The first behaviour is achieved by using the lock\_account flag in the pam.conf file. The account will have its /etc/shadow entry locked by using the passwd command when too many incorrect password attempts are made.

The second behaviour is achieved by using the timeout\_account flag in the pam.conf file. When the number of incorrect passwords is too large a timeout timer is set. Any further activity of the account during the timeout period will get the timer restarted. During the timeout period any login attempt, including an attempt using a correct password, will restart the timer.

The final behaviour is achieved by specifying neither the lock\_account nor the timeout\_account flags. When the count of incorrect passwords exceeds the limit the pam structure will refuse all further login attempts. The count of incorrect passwords can be reset by using the login\_limit command, or , if the login\_limit entry is specified in the password section of pam.conf, by changing the password.

If the account is not locked, but instead the count is used to fail login attempts then to reactivate the account the login\_limit program is required to reset the count. This can be accomplished by the command:

```
# login_limit -c username
```

An entry for the login limit module has to be placed in the authentication area of the pam.conf file. Optionally this module may also be placed in the account and password areas of the pam.conf file.

The entry should be placed as the last entry in a group, with a control flag of **required**. In the authentication section the entry before the pam\_login\_limit should have a control flag of **sufficient**. Because the PAM infrastructure has no branching control, other than the different sections we cannot easily tell if a previous module failed. If the module is called in the authentication section it is assumed that an earlier module failed, and a failed password attempt will be logged.

If the module before the pam\_login\_limit returns PAM\_IGNORE, such as pam\_dial\_auth in Solaris 8, 9, or 10 it may be necessary to add a pam\_null with a control flag of **sufficient** instead. This allows the pam\_login\_limit functionality to be enabled. Another option is to swap the order of pam\_unix\_auth and pam\_dial\_auth.

For example in Solaris 10 and later:

```
login  auth requisite          pam_authtok_get.so.1
login  auth required          pam_dhkeys.so.1
login  auth required          pam_unix_cred.so.1
login  auth required          pam_dial_auth.so.1
login  auth sufficient       pam_unix_auth.so.1
login  auth required          pam_login_limit.so.1 lock_account
```

If all the previous modules in the authentication section succeed the login limit module will not be called as an authentication module, but rather will be called when the authentication performs the credential setting pass of the authentication section. This action will reset the failure count to zero.

In Solaris 8 with patches and later both the format of the pam.conf file, and the default modules have changed. In these cases an example which includes the optional insertion into both the account and password sections of the /etc/pam.conf file would be:

```
#
# Default definitions for Authentication management
# Used when service name is not explicitly mentioned for authentication
#
other      auth requisite      pam_authtok_get.so.1
other      auth required       pam_dhkeys.so.1
other      auth sufficient    pam_unix_auth.so.1
other      auth required       pam_login_limit.so.1 lock_account
#
# Default definition for Account management
# Used when service name is not explicitly mentioned for account management
#
other      account required    pam_login_limit.so.1 lock_account
other      account requisite    pam_roles.so.1
other      account required     pam_projects.so.1
other      account required     pam_unix_account.so.1
#
# Default definition for Password management
# Used when service name is not explicitly mentioned for password management
#
other      password required    pam_dhkeys.so.1
other      password requisite    pam_authtok_get.so.1
other      password requisite    pam_authtok_check.so.1
other      password required     pam_authtok_store.so.1
other      password optional    pam_login_limit.so.1
```

Any of the standard login methods can utilise the login attempt counting. It is useful to add the module to all *auth* lines, except perhaps *rsh*.

Note that login limit does not require a password or other authentication token, and so should be placed after modules that do require authentication tokens. This module processes only internal information.

The login limit module provides diagnostic and status information via syslog. Each message is sent to the *auth* facility at a severity depending on the message itself. Each line includes the 'pam\_login\_limit' tag so that it can be found using an automated search. Successful login attempts are logged as INFO messages, failures as NOTICE, and any internal errors as ERR or CRIT depending on the error. Debugging messages are logged as DEBUG.

In addition the login limit module may also be added to the password area of pam.conf. This will allow the changing of a password to automatically reset the number of attempts logged against that account. This means that after locking an account the act of changing the password for that account will reset the count for that account rather than the count having to be reset manually using *login\_limit*. For Solaris 8 and before the module has to be added at the end of the password group,

with a control flag of **required**. In Solaris 9 and later the module is placed at the end of the stack just after the `pam_authtok_store` module. It should have a control flag of **optional**.

For example in Solaris 9:

```
other password required pam_dhkeys.so.1
other password requisite pam_authtok_get.so.1
other password requisite pam_authtok_check.so.1
other password required pam_authtok_store.so.1
other password optional pam_login_limit.so.1
```

In Solaris 8 without patches (we recommend patching), and for Solaris 7 changes are similar to:

```
telnet auth sufficient /usr/lib/security/pam_unix.so.1
telnet auth required /usr/lib/security/pam_login_limit.so.1 lock_account

dtlogin auth sufficient /usr/lib/security/pam_unix.so.1
dtlogin auth required /usr/lib/security/pam_login_limit.so.1 lock_account

rlogin auth sufficient /usr/lib/security/pam_rhosts_auth.so.1
rlogin auth sufficient /usr/lib/security/pam_unix.so.1
rlogin auth required /usr/lib/security/pam_login_limit.so.1 lock_account

other auth sufficient /usr/lib/security/pam_unix.so.1
other auth required /usr/lib/security/pam_login_limit.so.1 lock_account

dtlogin account required /usr/lib/security/pam_login_limit.so.1 lock_account
dtlogin account required /usr/lib/security/pam_unix.so.1

other account required /usr/lib/security/pam_login_limit.so.1 lock_account
other account required /usr/lib/security/pam_unix.so.1
```

## Login Times (COMSpamlt)

The login times module provides login limiting on a time basis. When a login attempt is made the database is checked to see if a login at that time is allowed. If no login is allowed the module will return a failure. This will normally cause the login attempt to fail, although the exact actions will depend on the control flags specified.

The module understands a number of options that may be specified in pam.conf: (Please see the section on common options for further details)

<code>debug</code>	Turn on additional syslog debugging messages.
<code>silent</code>	Turns off Info/error/notice/crit syslog messages.
<code>nowarn</code>	Turns off any warning messages for the user.
<code>check_root</code>	Check the root user. By default the root user is not checked. Specifying this option causes root logins to be subject to the normal checks.
<code>use_gmt</code>	Check on the basis of GMT time, not local time. The control file must have been built on this basis also, or strange results will ensue.
<code>file=<i>file</i></code>	This option overrides the default control file.
<code>check=<i>check</i></code>	Changes the checks that are performed on the control file to ensure it has not been tampered with. See <b>Common Options</b> for a full explanation of the checks that are possible.

The module uses the `/var/adm/login_times` control file. This file is a sparse database that holds an entry for each active user. The entry contains the times when logins are permitted and prohibited. This file may be manipulated by the `login_times` program.

The entry for the login times PAM module should be placed as one of the first entries in an authentication group. It should have a control flag of **required**.

For example:

```
telnet auth required /usr/lib/security/pam_login_times.so.1
telnet auth required /usr/lib/security/pam_unix.so.1
```

The login times module provides diagnostic and status information via syslog. Each message is sent to the `auth` facility at a severity depending on the message itself. Each line includes the 'pam\_login\_times' tag so that it can be found using an automated search. Successful login attempts are logged as INFO messages, failures as NOTICE, and any internal errors as ERR or CRIT depending on the error. Debugging messages are logged as DEBUG.

## Remote Host Login (COMSpamrh)

The remote host module provides for remote connection checking on a per user basis. Each connection is checked for user, service, and host to see if the connection should be allowed or denied. This functionality is similar to the well known TCP wrappers program, but occurs at the PAM level, providing access to the user name as well as the host and service.

The module understands a number of options that may be specified in pam.conf: (Please see the section on common options for further details)

debug	Turn on additional syslog debugging messages.
silent	Turns off Info/error/notice/crit syslog messages.
nowarn	Turns off any warning messages for the user.
no_check_root	Do not check the root user. By default the root user is checked. Specifying this option causes root logins to be exempted from the normal checks. Note that this is different to the default in the other modules.
localhost	Override the remote host lookup, all host lookups will be taken as <i>localhost</i> . This is useful for non-remote services such as dtlogin.
file= <i>file</i>	Specify an alternate configuration file. Normally the <i>/var/adm/remote_hosts</i> file is used as the configuration file.
check= <i>check</i>	Changes the checks that are performed on the configuration file to ensure it has not been tampered with. See <b>Common Options</b> for a full explanation of the checks that are possible.

The module will consult the */var/adm/remote\_hosts* control file. This file contains a series of lines that are searched from front to end. Each line consists of a user field, service field, host field, and an allow or deny separated by colons (:).

The first field is a comma separated list of user names, or the wild card '\*' to apply to all users. The username is case sensitive in matches.

The second field is a comma separated list of services, or the wild card '\*' which allows all services. The services are the same as the PAM service descriptor.

The third field is a comma separated list of hosts, or the wild card '\*' which means all hosts. Each hostname can include a netmask, either specified as a CIDR */number*, or as a dot separated group of octets after an initial slash (*/*).

The final field is either 'allow' or 'deny'.

If no match occurs in the control file the default is to deny access. This default can be changed by adding as the last line the wildcard entry '\*:\*:\*:allow' which provides for a

default allow.

The entries for the remote hosts pam module should be added as one of the first lines in a PAM stack. It should have a control flag of **required**.

For example:

```
telnet auth required /usr/lib/security/pam_remote_hosts.so.1
telnet auth required /usr/lib/security/pam_unix.so.1
```

The remote hosts module provides diagnostic and status information via syslog. Each message is sent to the *auth* facility at a severity depending on the message itself. Each line includes the 'pam\_remote\_hosts' tag so that it can be found using an automated search. Successful login attempts are logged as INFO messages, failures as NOTICE, and any internal errors as ERR or CRIT depending on the error. Debugging messages are logged as DEBUG.

The remote hosts module includes a testing program, usually installed in /usr/local/sbin. This program, test\_remote\_hosts, can be used to check whether a particular combination of user, service and host would be allowed or denied.

## Login History (COMSpamph)

The history module provides password re-use control. When a password change attempt is made the database is checked to see if that password has been used previously within the saved password set. If a previous use of that password is found the module will return a failure. This will normally cause the password change attempt to fail, although the exact actions will depend on the control flags specified.

The module understands a number of options that may be specified in `pam.conf`: (Please see the section on common options for further details)

<code>debug</code>	Turn on additional syslog debugging messages. If this options is repeated three or more times then passwords are sent to syslog in clear text.
<code>silent</code>	Turns off Info/error/notice/crit syslog messages.
<code>nowarn</code>	Turns off any warning messages for the user.
<code>try_first_pass</code>	The default. Attempts to check a password that has already been obtained by an earlier module. If no passwords were obtained then we prompt for the passwords.
<code>use_first_pass</code>	Use the passwords obtained by an earlier module. If no passwords were obtained then we do not request the passwords, but fail instead.
<code>check_root</code>	Check the root user. By default the root user is not checked. Specifying this option causes root logins to be subject to the normal checks.
<code>root_bypass</code>	Do not check if <code>uid()==0</code> is true. Allows root to change passwords to entries that are already in the history file.
<code>get_oldauthtok</code>	Ask the user for the old password if not already supplied. If the login history module is used early in the stack, and further modules require the old authentication token use this option to ask for it. Normally the module only deals with the new password.
<code>simple_prelim_check</code>	Do not check the history on the first pass. Only check the history on the second pass. The Sun standard is to acquire and check on the first pass, and update if necessary on the second pass. If Linux derived modules are used in the stack they usually only acquire the passwords on the second pass. This option makes the history module use this behaviour.
<code>history=<i>number</i></code>	Set the history to this many passwords. The default is 5. The bounds are min 2, max 256. Having a larger history reduces the possibility of cycling, but increases the compute time necessary to change a password.

<code>file=<i>filename</i></code>	Changes the default history file. The default file name is <code>/var/adm/passwd_history</code> . This options allows different programs to use different histories.
<code>check=<i>options</i></code>	Changes the checks that are performed on the history file to ensure it has not been tampered with. See <b>Common Options</b> for a full explanation of the checks that are possible.
<code>func=<i>salt</i></code>	Can be used to force a particular salt for the encryption of the passwords stored in the history file. This option is better used to cause an alternate encryption technique to be used in Solaris 9 or later systems that have alternate password encryption routines installed. See below for more details.

The module uses the `/var/adm/passwd_history` control file. This file is a sparse database that holds an entry for each user. Each entry contains the previous passwords in an encrypted form. This file should be protected in a fashion identical to the `/etc/shadow` file, as it also holds the current encrypted passwords.

For Solaris 8 and before the entry for the history PAM module should be placed as the last entry in a password group. It should have a control flag of **required**.

For example:

```
other password required /usr/lib/security/pam_unix.so.1
other password required /usr/lib/security/pam_history.so.1
```

For Solaris 9 and patched Solaris 8 the history module should be placed into the middle of the stack just below the `pam_authtok_check` module. It should have a control flag of **requisite**.

For example:

```
other password required pam_dhkeys.so.1
other password requisite pam_authtok_get.so.1
other password requisite pam_authtok_check.so.1
other password requisite pam_history.so.1
other password required pam_authtok_store.so.1
```

In later versions of Solaris 9 and later the password encryption routine `crypt(3c)` has been changed. It now uses a library of encryption routines. The default is specified in the `/etc/security/policy.conf` file. These routines use the salt string to decide which encryption routine to use. By forcing the salt to a particular value, notably of the form `$<routine name>$` we can force all saved passwords to be stored using the specified algorithm.

For example:

```
other password required pam_dhkeys.so.1
other password requisite pam_authtok_get.so.1
other password requisite pam_authtok_check.so.1
other password requisite pam_history.so.1 func=$md5$
other password required pam_authtok_store.so.1
```

## NULL (COMSpamnl)

The Null module provides a means to place debugging statements, and affect the control flow of the PAM stack. By default the module will return PAM\_SUCCESS and have no output.

The module understands a number of options that may be specified in pam.conf: (Please see the section on common options for further details)

debug	Turn on additional syslog debugging messages.
silent	Turns off Info/error/notice/crit syslog messages.
nowarn	Turns off any warning messages for the user.
fail	changes the return value to PAM_AUTH_ERR
ignore	changes the return value to PAM_IGNORE
echo= <i>msg</i>	echoes the specified message to syslog at LOG_INFO.

The module may be placed into either the auth, account, session, or passwd sections of the pam.conf file. The syslog messages indicate which section was invoked, and the name of the invoking service.

## Chroot (COMSpamch)

The chroot module provides the ability to chroot a user on successful login. When a login attempt is made the module checks the home directory path. If the directory path holds two directory separators next to each other the path leading to this pair of characters is used for a chroot operation. If any chroot and subsequent chdir fails the module returns failure, otherwise, if no chroot was necessary, or the chroot and chdir succeeded the module returns success.

The module understands a number of options that may be specified in pam.conf: (Please see the section on common options for further details)

debug	Turn on additional syslog debugging messages.
silent	Turns off Info/error/notice/crit syslog messages.
nowarn	Turns off any warning messages for the user.
check_root	Check the root user. By default the root user is not checked. Specifying this option causes root logins to be subject to the normal chroot procedure.

The entry for the chroot PAM module should be placed as the last entry in a session group. It should have a control flag of **required**.

For example:

```
telnet session required /usr/lib/security/pam_unix.so.1
telnet session required /usr/lib/security/pam_chroot.so.1
```

The chroot module provides diagnostic and status information via syslog. Each message is sent to the *auth* facility at a severity depending on the message itself. Each line includes the 'pam\_chroot' tag so that it can be found using an automated search. Successful chroot attempts are logged as INFO messages, failures as NOTICE, and any internal errors as ERR or CRIT depending on the error. Debugging messages are logged as DEBUG.

The chroot module requires that a sufficient environment exists in the new root tree for any required programs. An example on how to build a chroot environment is included in the ftpd manual page. For typical users additional libraries and devices would probably be needed.

As an example, using the loopback filesystem, the following vfstab provides the ability to have normal users, including CDE users, chroot to /export/home/user\_root. Since the environment from that point is reasonably identical most programs will continue to work. The exact vfstab that would be required for a given situation may differ, but programs that fail can be checked using truss to see what facilities they lack.

#device #to mount #	device to fsck #	mount point	FS type	fsck pass	mount at boot	mount options
/dev/dsk/c1d0s2	/dev/rdisk/c1d0s2	/usr	ufs	1	yes	-
fd	-	/dev/fd	fd	-	no	-
/proc	-	/proc	proc	-	no	-
/dev/dsk/c0t0d0s1	-	-	swap	-	no	-
/dev/dsk/c0t0d0s0	/dev/rdisk/c0t0d0s0	/	ufs	1	no	-
swap	-	/tmp	tmpfs	-	yes	-
/usr	-	/export/home/user_root/usr	lofs	-	yes	ro
/etc/dt	-	/export/home/user_root/etc/dt	lofs	-	yes	ro
/etc/net	-	/export/home/user_root/etc/net	lofs	-	yes	ro
/tmp	-	/export/home/user_root/tmp	lofs	-	yes	-
/proc	-	/export/home/user_root/proc	proc	-	no	-
fd	-	/export/home/user_root/dev/fd	fd	-	no	-

User home directories would be placed below /export/home/user\_root, and within the chroot environment /export/home/user\_root has to point back to the root, I.e. /export/home/user\_root -> ../../.. as a symbolic link. This link is due to the different places that home directories are obtained, and used.

## Common Options

All the PAM modules have three common options. These options affect the messages that are produced by the modules. The common options and their effect are:

`debug`        turn on debugging messages. If this option is repeated then additional debugging is produced. This option also disables the silent option.

---

**CAUTION at debugging level three and above password information may be produced and stored in clear text. If this level of debugging is selected then a critical syslog message is produced to warn against this eventuality.**

---

`silent`        turn off informational syslog messages. Depending on the number of times this option is repeated fewer and fewer messages are produced.

One:            do not produce login success messages  
Two:            do not produce login success or system error messages  
Three:          do not produce login success, system error, or login failure messages  
Four:           do not produce any syslog messages, including critical system error messages.

It is recommended that `silent` never exceed three in normal operations.

`nowarn`        do not print any warning messages to the user about soon, but not current, events.

These three options allow the customisation of the level of output sent to the user and logged for future reference. The default is to have no debugging but all informational syslog messages generated, and to have messages generated warning of events for the user.

In addition to these three options several modules have a `check` option. The `check` option allows fine grain control over the file checking routine. By default the modules which use a control or data file check that file for ownership, permissions, and various other attributes. The `check` option can be used to control these checks.

The `check` option takes a series of control characters. Non regular files will be flagged and not processed. In general the module will fail if the check fails. The meaning of each character is:

`R`            check that the file is a regular file.  
`M`            check the mode of the file. The file must be mode 0400 or 0600 only.  
`S`            check the file owner is uid 0.

U check the file is owned by the uid that is running the module. This is useful if an application such as Apache is running the PAM modules as apache instead of as root.

A check that the file has no ACL entries. This prohibits backdoors through the ACL system.

For Solaris 10 and 11 the ACL check functions for both UFS style ACLs and ZFS style ACEs. If UFS style ACLs are found there should be no additional ACLs. If ACEs are found then we check for 6 ACEs (Solaris 10) or 3 ACEs (Solaris 11). Please note that we do not check the information in the ACEs, only the count of ACEs.

If ACLs or ACEs have been added according to local security policy then the A control character can be optionally followed by a numeric value which is the number of ACLs or ACEs expected.

T check that the file has no attribute entries (Solaris 9 onwards). We do not allow the file to be the root of an attribute directory structure.

The default control characters are: RMSAT

Control characters that are specified but have no meaning are silently ignored.