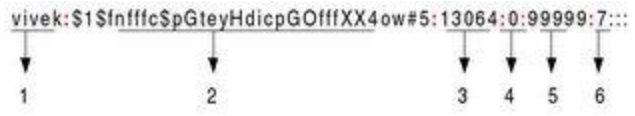
Unix Security Hardening and Monitoring

1. Password Access and Complexity

A. /etc/shadow file fields



```
vivek:$1$fnfffc$pGteyHdicpGOfffXX4ow#5:13064:0:99999:7:::
```

(Fig.01: /etc/shadow file fields)

1. User name : It is your login name
2. Password: It your encrypted password. The password should be minimum 6-8 characters long including special characters/digits
3. Last password change (lastchanged): Days since Jan 1, 1970 that password was last changed
4. Minimum: The minimum number of days required between password changes i.e. the number of days left before the user is allowed to change his/her password
5. Maximum: The maximum number of days the password is valid (after that user is forced to change his/her password)
6. Warn : The number of days before password is to expire that user is warned that his/her password must be changed
7. Inactive : The number of days after password expires that account is disabled
8. Expire : days since Jan 1, 1970 that account is disabled i.e. an absolute date specifying when the login may no longer be used

Managing Password Aging

Password aging is a mechanism you can use to force users to periodically change their passwords.

Password aging allows you to:

- Force a user to choose a new password the next time the user logs in. (See "Forcing Users to Change Passwords" for details.)
- Specify a maximum number of days that a password can be used before it has to be changed. (See "Setting a Password Age Limit" for details.)
- Specify a minimum number of days that a password has to be in existence before it can be changed. (See "Setting Minimum Password Life" for details.)
- Specify that a warning message be displayed whenever a user logs in a specified number of days before the user's password time limit is reached. (See "Establishing a Warning Period" for details.)
- Specify a maximum number of days that an account can be inactive. If that number of days pass without the user logging in to the account, the user's password will be locked. (See "Specifying Maximum Number of Inactive Days" for details.)
- Specify an absolute date after which a user's password cannot be used, thus denying the user the ability to log on to the system. (See"Password Privilege Expiration" for details.)

Keep in mind that users who are already logged in when the various maximums or dates are reached are not affected by the above features. They can continue to work as normal.

Password aging limitations and activities are only activated when a user logs in or performs one of the following operations:

- login
- rlogin
- telnet
- ftp

These password aging parameters are applied on user-by-user basis. You can have different password aging requirements for different users. (You can also set general default password aging parameters as described in "Managing Password Aging".)

Forcing Users to Change Passwords

There are two ways to force a user to change passwords the next time the user logs in:

Force change keeping password aging rules in effect

```
passwd -f username
```

Force change and turn off password aging rules

```
passwd -x 0 username
```

Setting a Password Age Limit

The **-max** argument to the passwd command sets an age limit for the current password. In other words, it specifies the number of days that a password remains valid. After that number of days, a new password must be chosen by the user. Once the maximum number of days have passed, the next time the user tries to login with the old password administrator. Your password has been expired for too long message is displayed and the user is forced to choose a new password in order to finish logging in to the system.

The max argument uses the following format:

```
passwd -x max username
```

Where:

- *username* is the login ID of the user
- *max* is one of the following values:
    - **Greater than zero**. Any number greater than zero sets that number of days before the password must be changed.
    - **Zero (0)**. A value of zero (0) forces the user to change passwords the next time the user logs in, and it then turns off password aging.
    - **Minus one (-1**). A value of minus one (-1) turns off password aging. In other words, entering **passwd -x -1** *username* cancels any previous password aging applied to that user.

For example, to force the user schweik to change passwords every 45 days, you would type the command:

```
station1% passwd -x 45 schweik
```

Setting Minimum Password Life

The *min* argument to the passwd command specifies the number of days that must pass before a user can change passwords. If a user tries to change passwords before the minimum number of days has passed, a Sorry less than N days since the last change message is displayed.

The *min* argument uses the following format:

```
passwd -x max -n min username
```

Where:

- *username* is the login ID of the user
- *max* is the maximum number of days a password is valid as described in the section above
- *min* is the minimum number of days that must pass before the password can be changed.

For example, to force the user eponine to change passwords every 45 days, and prevent him from changing it for the first 7 days you would type the command:

```
station1% passwd -x 45 -n 7 eponine
```

The following rules apply to the *min* argument:

- You do not have to use a *min* argument or specify a minimum number of days before a password can be changed.
- If you do use the *min* argument, it must always be used in conjunction with the **-max** argument. In other words, in order to set a minimum value you must also set a maximum value.
- If you set *min* to be greater than *max*, the user is unable to change passwords at all. For example, the command passwd -x 7 -n 8prevents the user from changing passwords. If the user tries to change passwords, the You may not change this passwordmessage is displayed. Setting the *min* value greater than the *max* value has two effects:
  - The user is unable to change password. In this case, only someone with administer privileges could change the password. For example, in situations where multiple users share a common group password, setting the *min* value for that password greater than the *max* value would prevent any individual user from changing the group password.
  - The password is only valid for the length of time set by the *max* value, but the user cannot change it because the *min* value is greater than the *max* value. Thus, there is no way for the user to prevent the password from becoming invalid at the expiration of the *max* time period. In effect, this prevents the user from logging in after the *max* time period unless an administrator intervenes.

Establishing a Warning Period

The *warn* argument to the passwd command specifies the number of days before a password reaches its age limit that users will start to seeing a Your password will expire in N days message (where *N* is the number of days) when they log in.

For example, if a user's password has a maximum life of 30 days (set with the **-max** argument) and the warn value is set to 7 days, when the user logs in on the 24th day (one day past the warn value) the warning message Your password will expire in 7 days is displayed. When the user logs in on the 25th day the warning message Your password will expire in 6 days is displayed.

Keep in mind that the warning message is not sent by Email or displayed in a user's console window. It is displayed only when the user logs in. If the user does not log in during this period, no warning message is given.

Keep in mind that the *warn* value is **relative** to the *max* value. In other words, it is figured backwards from the deadline set by the *max* value. Thus, if the *warn* value is set to 14 days, the Your password will expire in N days message will begin to be displayed two weeks before the password reaches its age limit and must be changed.

Because the *warn* value is figured relative to the *max* value, it only works if a *max* value is in place. If there is no *max* value, *warn* values are meaningless and are ignored by the system.

The *warn* argument uses the following format:

```
passwd -x max -w warn username
```

Where:

- *username* is the login ID of the user.
- **max** is the maximum number of days a password is valid as described on "Setting a Password Age Limit".
- *warn* is the number of days before the password reaches its age limit that the warning message will begin to be displayed.

For example, to force the user nilovna to change passwords every 45 days, and display a warning message 5 days before the password reaches its age limit you would type the command:

```
station1% passwd -x 45 -w 5 nilovna
```

The following rules apply to the *warn* argument:

- You do not have to use the *warn* argument or specify a warning message. If no *warn* value is set, no warning message is displayed prior to a password reaching its age limit.
- If you do use the *warn* argument, it must always be used in conjunction with the *max* argument. In other words, in order to set a warning value you must also set a maximum value.

Note -

You can also use Solstice AdminSuite™ to set a warn value for a user's password.

Turning Off Password Aging

There are two ways to turn off password aging for a given user:

Turn off aging while allowing user to retain current password

```
passwd -x -1 username
```

Force user to change password at next login, and then turn off aging

```
passwd -x 0 username
```

This sets the *max* value to either zero or -1 (see "Setting a Password Age Limit" for more information on this value).

For example, to force the user mendez to change passwords the next time he logs in and then turn off password aging you would type the command:

```
station% passwd -x 0 mendez
```

B. /etc/default/passwd

Specifing a password policy

There is a central file in Solaris controling the password policy. In /etc/default/passwd you define what requirements a password must fulfill before Solaris allows the user to set this password. Let´s have a look in the actual file of a standard solaris system. You have to log into your system as root. One important note for trying out this feature. You need to log into your system as a normal user in a different window.root can set any password without a check by the password policy thus it would look like that your configuration changes had no effect:

```
# cat passwd
[... omitted CDDL header ...]
#
MAXWEEKS=
MINWEEKS=
PASSLENGTH=6
#NAMECHECK=NO
#HISTORY=0
#MINDIFF=3
#MINALPHA=2
#MINNONALPHA=1
#MINUPPER=0
#MINLOWER=0
#MAXREPEATS=0
#MINSPECIAL=0
#MINDIGIT=0
#WHITESPACE=YES
#DICTIONLIST=
#DICTIONDBDIR=/var/passwd
```

You enable the checks by uncommenting it and set a reasonable value to the line. When you enable all the checks, it´s actually harder to find a valid password than a non-valid one. Whenever thinking about a really hard password policy you should take into consideration, that people tend to make notes about their password when they can´t remember it. And a strong password under the keyboard is obviously less secure than a weak password in the head of the user.

Parameter	Description
MAXWEEKS	This variable specifies the maximum age for a password.
MINWEEKS	This variable specifies the minimum age for a password. The rationale for this

	settings gets clearer when i talk about the HISTORY setting
PASSLENGTH	The minimum length for a password
HISTORY	This variable specifies the length of a history buffer. You can specify a length of up to 26 passwords in the buffer. The MINWEEKS buffer is useful in conjunction with this parameter. There is a trick to circumvent this buffer and to get you old password back. Just change it as often as the length of the buffer plus one time. The MINWEEK parameter prevents this.
WHITESPACE	This variable defines if you you are allowed to use a whitespace in your password
NAMECHECK	When you set this variable to YES, the system checks if the password and login name are identical. So using the password root for the useroot would be denied by this setting. The default, by the way is, yes

Besides of this basic checks you can use /etc/default/passwd/ enforce checks for the complexity of passwords. So you can prevent the user from setting to simple passwords.

Parameter	Description
MINDIFF	Let´s assume you´ve used 3 here. If your old password was batou001, a new password would be denied, if you try to use batou002 as only on character was changed. batou432 would be a valid password.
MINUPPER	With this variable you can force the usage of upper case characters. Let´s assume you´ve specified 3 here, a password like wasabi isn´t an allowed choice, but you could use WaSaBi
MINLOWER	With this variable you enable the check for the amount of lower case characters in your password. In the case you´ve specified 2 here, a password like WASABI isn´t allowed, but you can use WaSaBI
MAXREPEATS	Okay, some users try to use passwords like aaaaaa2=. Obviously this isn´t really a strong password. When you set this password to 2 you, it checks if at most 2 consecutive characters are identical. A password like waasabi would be allowed, but not a password like waaasabi
MINSPECIAL	The class SPECIAL consists out of characters like !=(). Let´s assume you´ve specified 2, a password like !ns!st= would be fine, but the password insist is

	not a valid choice.
MINDIGIT	With this password you can specify the amount of the numbers in your password. Let´s a assume you specify 2, a password like snafu01would will be allowed. A password like snafu1 will be denied.
MINALPHA	You can check with this variable for a minimum amount of alpha chars (a-z and A-Z) . When you set a value of 2 on this variable, a password like aa23213 would be allowed, a password like 0923323 would be denied
MINNONALPHA	This checks for the amount of non-alpha characters (0-9 and special chars). A value of 2 would lead to the denial of wasabi, but a password like w2sab! is okay

Using wordlists

There is another way to force stronger passwords. You can deny every password that is located in a list of words. The program for changing password is capable to compare the new password against a list of words. With this function you can deny the most obvious choices of passwords. But you should initialize the dictionary with a list of words before you can use this feature.

```
# mkpwdict -s /usr/share/lib/dict/words
mkpwdict: using default database location: /var/passwd.
```

The file /usr/share/lib/dicts/words is a file in the Solaris Operating System containing a list of words. It´s normally used by spell checking tools. Obviously you should use a workdlist in your own language, as user tend do choose words from their own language as passwords. So an english wordlist in Germany may be not that effective.You find a list of other wordlists here
Now you have to tell Solaris to use this lists. There are some parameters in the/etc/default/password i didn´t covered before:

Parameter	Description
DICTIONLIST	This variable can contain a list of dictionary files seperated by a comma. You must specify full pathnames. The words from these files are merged into a database that is used to determine whether a password is based on a dictionary word

DICTIONDBDIR The directory where the generated dictionary databases reside

When none of the both variables is specified in the /etc/default/passwd then no dictionary check is performed.

Let´s try it. I´ve uncommented the DICTIONDBDIR line of the /etc/default/passwd file and used the standard value /var/passwd. One of the word in the dictionary i imported is the wordairplane:

```
$passwd
passwd:Changing password for jmoekamp
Enter existing login password: chohw!2
New Password: airplane
passwd: password is based on a dictionary word.
```

Solaris denies the password as it´s based on a word in the imported dictionary.

Monitoring Logins

How to Monitor All Failed Login Attempts

This procedure captures in a `syslog` file all failed login attempts.

1. Assume the Primary Administrator role, or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2,*Working With the Solaris Management Console (Tasks),* in *System Administration Guide: Basic Administration*.

2. Set up the `/etc/default/login` file with the desired values for `SYSLOG` and `SYSLOG_FAILED_LOGINS`

   Edit the `/etc/default/login` file to change the entry. Make sure that **`SYSLOG=YES`** is uncommented.

```
# grep SYSLOG /etc/default/login
# SYSLOG determines whether the syslog(3) LOG_AUTH facility
# should be used
SYSLOG=YES
…
SYSLOG_FAILED_LOGINS=0
#
```

3. Create a file with the correct permissions to hold the logging information.
   a. Create the `authlog` file in the `/var/adm` directory.

      ```
      # touch /var/adm/authlog
      ```

   b. Set read-and-write permissions for `root` user on the `authlog` file.

      ```
      # chmod 600 /var/adm/authlog
      ```

   c. Change group membership to `sys` on the `authlog` file.

      ```
      # chgrp sys /var/adm/authlog
      ```

4. Edit the `syslog.conf` file to log failed password attempts.

   The failures should be sent to the `authlog` file.

   a. Type the following entry into the `syslog.conf` file.

      Fields on the same line in `syslog.conf` are separated by tabs.

      ```
      auth.notice <Press Tab>  /var/adm/authlog
      ```

   b. Refresh the configuration information for the `syslog` daemon.

      ```
      # svcadm refresh system/system-log
      ```

5. Verify that the log works.

   For example, as an ordinary user, log in to the system with the wrong password. Then, in the Primary Administrator role or as superuser, display the `/var/adm/authlog` file.

```
# more /var/adm/authlog
Nov  4 14:46:11 example1 login: [ID 143248 auth.notice]
 Login failure on /dev/pts/8 from example2, stacey
#
```

6. Monitor the `/var/adm/authlog` file on a regular basis.

Example 3–4 Logging Access Attempts After Three Login Failures

Follow the preceding procedure, except set the value of `SYSLOG_FAILED_LOGINS` to `3` in the `/etc/default/login` file.

Example 3–5 Closing Connection After Three Login Failures

Uncomment the `RETRIES` entry in the `/etc/default/login` file, then set the value of `RETRIES` to `3`. Your edits take effect immediately. After three login retries in one session, the system closes the connection.
