Assignment 2 - `/etc/shadow` File Modification

COSC1131/1133 Unix Systems Administration
RMIT University
School of Computer Science and Information Technology

2 May 2005

Introduction

Without providing a major discussion of the issues, there are arguments both for and against having policies that require computer system users to change passwords on a regular basis. The primary argument in favour is that, if a password is compromised, it will have a limited useful life. The primary argument against is that, if they are forced to change passwords frequently, users will either use weak passwords, insert some "date" code and re-use passwords, or write the passwords down and store them near the workstation. (*e.g.* on a "Post-It Note"^{TM} attached to the monitor or stuck to the underside of the keyboard.

Management has asked you to implement a system of forced password changes in an environment with approximately 10,000 users. (This is far too many to simply edit the shadow file by hand – a script will be required.)

The only good news is that you have convinced them that changing passwords more often than every 93 days (three months) is likely to be counter-productive. But the system you develop will need to force changes with a maximum of 1 day of warning for cases where there is a suspected system compromise.

General Specification

Your task is to write a script that will modify the file "`/etc/shadow`" so that the following requirements will be met. While there may be specific commands available to achieve the required changes on some systems, they are not portable. Thus your script must make the changes directly.

The requirements for the script are as follows:

- It must be designed to run from a command line
- It must accept and validate command line parameters, as required.
- The primary requirement is that it must calculate the days since the epoch (1 January 1970) for the current date and modify this value as required. (Note that this is something that you would not generally attempt in a script – but it is an interesting exercise.) See the notes at the end for a web page that may help. At the very least, a loop that counts up the years from 1970 to the present, adding 365 or 366 days, as required, followed by a loop that adds days for each month (with an if statement to deal with February). Maybe use a case statement for this. Then add the days in the current month. (Using a variant date command is not an option since it would not be portable.)
- To initiate the system of passwords changed on a quarterly (three monthly) basis, the initial setting will be to force users to change their password within one week.
- To achieve this, the recommended method is to set the values in `/etc/shadow` so that:
    1. The "date of last password change" is 86 days earlier than the current date.
    2. There is a 7 day warning period so that users will be warned that a password change is required.
    3. The maximum days for a password is 93 days
    4. The minimum days before a password *may* be changed is 14 days
    5. The inactivity lock on the accounts is set to 30 days.

- To deal with a suspected system compromise the system will need to do the above, but to set the "date of last change" to 92 days previous.
- Potentially there are two ways that the command line could be structured:
  1. Accept a command option "`-d`" followed by the number of days that users are to be allowed to change their password
  2. Accept two command line options, "`-I` for initial to set the seven day period and "`-E` for emergency to force a one day period.
  3. In either case, a "`-u`" option followed by either a particular userid or a regular expression for a group of users will specify a limited subset of users to be changed
  4. Specifying "`-r`" *will* modify the root user's entry. Otherwise this entry should remain untouched.
  5. And "`-h`" will display a help screen that explains your syntax
- Note that your script should only modify "real" users who have a password in the `shadow` file.
- Finally, in a comment at the end of your script (no more than 1 paragraph) discuss other changes that you would either make or at least make sure are present on the system to make it more secure than the standard system. (Probably one to three points with one sentence each.)

> **Note:** For testing, simply copy a shadow file from one of the lab machines and add a bunch of new invented users.

Marking

You will earn 50-60% of the available marks if your script meets the minimum requirements and is not portable, *etc.*. If you meet all requirements and use some skill in the construction of your script you should receive a grade in the 70-80% range, and if you demonstrate considerable skill and perhaps manage to find a superior approach or a way to uses a command particularly well then you are into the 80-100% range. In general, the expectation is that a grade of 50-60% should be reasonably easy to achieve. Higher marks require substantially more thought. This section is intentionally a bit vague since a specific set of criteria often are used as a design basis for assignments. Also, when marking, it is generally clear when one assignment is arguably better than another, markers need some flexibility to ensure that all students are rewarded fairly.

A script that meets the requirements should be less than 100 lines of shell code including comments. If you're writing a *lot* more than 100 lines then perhaps you need to re-consider the design of your script.

1. Uses good shell programming practice as mentioned in the relevant lab sheets.
   (a) Comments - a key point is to comment stuff that may be difficult to understand if you come back to it after six months. An example would be any sort of complex regular expression. But *do not* comment obvious stuff *e.g.* - a **bad** example – do not do this:
   `foo="some stuff" # assign "some stuff" to foo`
   (b) Use of built-in shell variables as required
   (c) Unique filenames in the standard temporary file directory location for any temporary files
2. Correctly modifies the required users in `/etc/shadow`
3. Correctly implements command line options.
4. Some system that allows for easy changes to take into account differences between operating system and allows adjustment for differences in command and log locations (For the purposes of the assignment you may limit this to Solaris, Slackware and "unspecified".)

Submission Information

- This assignment is worth 10% of your final mark.
- This assignment is due on 22 May, and must be submitted before 23:59.
- Late submissions will incur a 10% penalty per day (or part thereof) it is late and a 100% penalty if it is 5 or more days late. Submission of your assignment is via `turnin` only. E-mailed submissions will be discarded and no response will be sent. If you do not know how to use `turnin`, it is your responsibility to find out. (**Hint:** `man turnin` on yallara or numbat) Failure to use `turnin` is not grounds for any extensions or special consideration.
- Use `COSC1131` as the course code and '`assignment_2`' as the project name in your submission.

> **Important Announcement Regarding Requesting Extensions.**
>
> With the exception of dire circumstances, no extension requests will be considered within 5 working days of the submission date.
> (By "Dire Circumstances", we mean things like hospitalisation of you or a close relative, *etc.* We may also require that a person requesting a late extension to prove that a significant body of the work has already been completed.)

Warning on Plagiarism

Students are reminded that this assignment is to be attempted individually. Plagiarism of any form will result in zero marks being given for this assessment, and can result in disciplinary action. See the school and university rules on plagiarism for more details.

Hints

You may find the following UNIX commands useful:
`grep`, `expr`, `getopt(s)`, and `awk`.

You may want to look at:

`http://www.frozenblue.net/tools/slack-book/?op=c2773.html`
`http://www.wilsonmar.com/datepgms.htm`

> **Note**
>
> While this specification has been written to be as un-ambiguous as possible, the chances are that some clarification will be required. Any clarification will be published to the subject newsgroup, `rmit.cs.UnixSystemsAdmin` with a subject line that contains the words "`Assignment 2 Clarification`".