Global Information Assurance Certification Paper

L is for Login

Carolee L. Rand
GSEC
version 1.4b

L is for login...(abstract/summanry section)

The first experience most people have with any computer usually begins with the process of "logging in." With just a user name and a password (hopefully, a password) a user finds his/her way onto a computer system.

Security policies usually define who, when and how a person may log in to a particular computer or network. They might dictate which accounts are available to which computer, an account naming scheme and/or password policies. On the surface, it might appear to an end user that the login mechanisms on various UNIX platforms are the same. The user might use the same user name and password on several different types of computers and wind up in the same directory, not knowing that the identification and authentication mechanisms varies from platform to platform.

This paper will look at login commands, authentication mechanisms, passwords and password management programs used in several UNIX platforms, highlighting aspects of Solaris 8 and Red Hat Linux (RH) 7.3.

G is for getty

In the beginning there was a getty process waiting on a serial port for in input from an end user. That getty process called the login process which accepted username and password input from the end user and attempted to authenticate that information with the system password file (/etc/passwd). The password file contained most of the information about the user needed by the system to authenticate the user and create the proper environment for her/him.

Today, login processes follows similar steps, but there are other login mechanisms available. Computers are often networked and/or run X-windows or X based window managers such as CDE, KDE and gnome Protocols such as NIS, XDM, RPC, etc. provide additional means of system access and logins. All perform the same function: obtain information from an end user, use it to authenticate against a database and allow the user access to the computer if the authentication is successful.

Early in UNIX System V, /etc/getty, /bin/login and /etc/passwd were the main components of the login procedure. The password file of yesteryear provides the basis for current day password files and merits investigation.

/etc/passwd continues to be the place where account information is stored (we will not be discussing LDAP here.) It is a flat file database, formatted with one record per line. Each record contains up to seven colon-delimited fields. (see fig. 1). The first field, username, can contain up to eight characters. Although this is how the user identifies themselves, the computer uses the numerical information in the UID and GID fields to differentiate between users.

The second field in /etc/passwd can hold the 1-way encrypted password hash and its

seed. It can also hold one or more characters that are not part of the password hash or it can be left empty.

Most security plans mandate that all accounts have passwords of some sort. What they are really mandating is that some text, either an encrypted hash and seed (valid password) or one or more ASCII characters that would not be part of the hash (invalid password) be present in the password field. An empty password field, two colons following the username, is an undesirable situation as it opens the door to unauthenticated system entry.

On systems that use the /etc/password file for account information and authentication, the password field may be formatted to force the user to apply rules about password change dates, expiration and expiration warnings. The password field in this type of password file is thirteen characters without special parameters in place. By adding a "," and up to two one character codes, password aging can be put into effect. The first code stands for number of Weeks until the password expires. The second (optional) field represents the minimum number of weeks until the password may be changed again. The default is zero weeks and is represented by a "."

These fields may also be formatted to force the user to change their password at the next login. By appending a "," followed by a "." we are setting the password to expire in zero weeks, (immediately) and the password program prompts the user for a new password at his/her next login. This is the closest this form of password authentication comes to enforcing a security policy mandating that each account have its own password.

One difference between the password files of yesteryear (late 1980's, early 1990's) and the password files of contemporary UNIX systems is that the encrypted password is no longer displayed in the password field in the world readable password file. Did I mention the /etc/passwd file is world readable? Back when computing power was not what it is today, exposing the encrypted password posed little security threat as the computing power required to "crack" even a weak password was beyond the reach of most. Additionally most networks were limited to the size of a room limiting the chances of an uninvited someone snooping on the networked machines. Contemporary password files continue to hold world readable information about the users, their home directories and shells. The password field has been moved to a restricted access (root access-only) file, usually /etc/shadow, and the password fields for all records in the password file hold a single character, usually asterisk.

Other information requirements for a valid interactive account include the listing of a "home directory" and a default "shell" for the user. On some UNIX systems, if the home directory listed for the user is unavailable, the system will let the user log in, but places the user in the system root directory (not so good) or in /tmp (better). On other platforms, such as IRIX, a user will not be able to log in if the specified shell program is unavailable. Placing the full path of available shells in the /etc/shells file "legalizes" the use of these shells and makes them available to other programs, like ftp. It is possible

to assign a shell in /etc/password that does not exist in /etc/shells and have success in RH LINUX.

It is a good practice to assign bogus shells to system accounts present in the /etc/passwd files that exist to facilitate several facilities (like lp) but are not supposed be used for interactive login. To prevent a successful login from these accounts, change or assign invalid shells that come with most UNIX operating systems such as "/bin/false" or "/sbin/nologin" to these accounts.

S is for Shadow

Shadow password files (/etc/shadow) make it more difficult for non-root users to obtain encrypted passwords. With shadow files in use, the encrypted passwords are saved in a non-world-readable file, /etc/shadow, that exists in tandem with the traditional, world-readable /etc/password file. The format of the /etc/shadow file provides additional benefits other than the obscuring of encrypted passwords. Up to eight fields per record may be used to set password change parameters. (See figure 2.) Password change and expiration dates can be set at the "day" level, rather than at the "week" level of the /etc/passwd file. These fields use real numbers, rather than coded entries. Additionally, it is possible to set a precise expiration date for the account.

On Solaris systems, the default values for password aging and size are found in /etc/default/passwd. On RH Linux systems, the default values are set in /etc/login.defs as well as other parameters for account creation. Assigning values in the /etc/shadow files overrides these parameters.

It is still very important to protect the shadow file or whatever file is used to hold account passwords. The computing power available in todays home computers is more than enough to allow password cracking utilities like "John the Ripper" and "L0phtcrack" to uncover the weakest passwords (or more) in a shadow file.

O is for Other (places to set system-wide login defaults)

Unpassworded accounts make it possible for anyone to log into a system knowing only half of the login equation, the account name. Operating systems are configurable to require that only accounts with passwords are used, however the configuration options differ between various platforms.

For example, on System V UNIX platforms such as Solaris and IRIX, the /etc/default/login file is available to set up system-wide account defaults, including password change and warning times and whether or not the account is required to have a password.

The /etc/default/login file that is part of the IRIX UNIX operating system offers two variables that can be set to require password entry to the system.

The PASSREQ variable, when set, requires that each account has a password. If the account has no password, the user is prompted to supply one at login time. I find this setting dangerous in the case where an intruder might find an unpassworded account. If the intruder logs in, he/she will be prompted to supply a new password and the logs will show only that a user changed their password.

The MANDPASS variable, when set also requires that users use a password. If there is no password set, it locks the user out of the system at login time. This is my preferred setting because it forces the user to contact an administrator to obtain access to the system and alerts them to the problem of the non-passworded account. It also prevents an intruder from easily logging into the system if they find the non-passworded account before the administrator.

The Solaris UNIX operating system also uses the /etc/default/login configuration file, but does not include the "MANDPASS" variable. An intruder finding a non-passworded account (for whatever reason it came to exist) could log in, create a new account and enter the system undetected.

Red Hat Linux systems use /etc/login.defs to provide some of the system wide defaults defined on other platforms in /etc/default login. In this file, password change parameters can be set, but it has no variable for controlling whether or not passwords are required for all accounts.

Root privileges are required to change all fields in the password and shadow files and modify the settings in the /etc/default/login and /etc/login.defs files. Various tools for editing files and variables exist across the many UNIX platforms and are available to both administrative and end-users. Some parameters of these commands are accessible by root only.

Many commands and GUI interfaces exist to give administrators guidance and assistance with working with account management and login default files. Some commands may be found on most UNIX platforms, others are platform specific. We will look at a few of these commands here.

Syntax Verification: /etc/passwd, /etc/shadow files

pwck This command is available on most System V UNIX variants. It checks the syntax of /etc/passwd and /etc/shadow by default, and reports on incorrect number of fields, unknown shells and missing home directories. Run this command (where available) after manually editing passwd and shadow files.

logins I have found this command only on Solaris systems. It has many parameters for checking various parts of the password and shadow files. It is also capable of checking the syntax of NIS password and shadow maps. It is a very helpful tool for searching for missing passwords in accounts when large password files are used. (logins -p). Use of both of these commands is limited to

the root user.

Password/Shadow File Creation

pwconv, grpconv These commands may be used by the root user to convert password and group files to password/group files and shadow/gshadow files. They automatically create the shadow/gshadow files and populate them with matching entries to the passwd/group files. These commands are found on most UNIX systems.

pwunconv Reverses the work of the pwconv command, merging the information from /etc/shadow and /etc/passwd files into one single /etc/passwd file. Not all platforms that use the pwconv command use the pwunconv command. Under HP-UX, pwunconv is not used and other practices must be followed. Before using a command like pwconv, it is wise to check how the particular platform deals with reversing the program.

Password/Shadow File Modification

vipw To limit the number of mistakes that could be made while editing a password file, the vipw command creates a a vi editing environment with syntax error checking facilities. Vipw locks the passwd or shadow file, so that it cannot be accessed for writing by anyone else while editing is taking place. Upon invocation, the target file is locked and a copy is created for the purpose of the edit. Upon completion of editing, during the save process, the file is checked for syntax and the user is prompted to make changes if problems are identified. Once the program validates the syntax, the new file overwrites the original file and the file locking is removed. This command is available on most UNIX systems and is only available to the root user.

chage A RH Linux command that gives the administrator the ability to modify password change, warn and expiration dates from the command line. Regular system users can use the command with the "-l" option to check on the expiration date of their password.

passwd This is the command recognized by most users as the one used to change passwords. Users have the capability to change their own password, as long as they can supply the current password to the program. The root user can change any accounts (on that system) password and is not prompted for the users current password.

In addition to being the primary password changing mechanism, password is capable of manipulating several password characteristics. It is important to check the man(ual) page for password on whatever platform is in use to see which options are available and how they are used.

In addition to encouraging users to create and use "strong passwords", passwords that include a mix of letters, numbers and punctuation characters, it is also good (in my opinion) to NOT change or set passwords on Friday afternoons.

chsh This command allows users to change their login shell. Without this command a user would have to get the root user to make the change. The user can specify any valid shell available on the system; it does not have to be listed in /etc/shells.

GUI Account Creation Tools

Most UNIX platforms have some screen-based tool for creating user accounts. Most prompt the administrator for account information and create entries in the appropriate system files, as well as create a home directory with several shell resource (rc) scripts in place. A few are listed here:

PLATFORM:	Solaris	IRIX	HP-UX	RHLinux	AIX
TOOL:	admintool	User Mgr.	SAM	linuxconf	SMIT

P is for PAM

The suite of tools known as Pluggable Authentication Modules (PAM) was developed by SUN Computer Systems and released to the world in 1996. It is designed to give the administrator the ability to customize security settings for all applications that require authentication without having to modify the application. It gives the administrator a way to add parameters to various authentication services such as requiring a strong password rather than just a password. PAM is found on most flavors of Linux and on many commercial UNIX platforms.

The PAM suite consists of a library, (/usr/lib/)libpam, a variety of modules and either a configuration file, *pam.conf* or a directory *pam.d.* Both types of configuration files are usually found in /etc. On Solaris 8 systems, modules are found in /usr/lib/security and have a pointer to them in the /etc/pam.conf file. Modules on many Linux systems are found in the /usr/security directory.

PAM syntax is similar whether the entry is found in the /etc/pam.conf file or in a service file in /etc/pam.d. The configuration syntax for a config file found in /etc/pam.d is:

module-type control module-path module arguments

The syntax for an entry in /etc/pam.conf is:

service module-type control module-path module-arguments

Module-type refers to the type of authentication that will be applied. In a process called "stacking," modules of the same type can be stacked in a configuration file to create the need for multiple authentication requirements to be met.

The four PAM module types are:

account Checks for valid account parameters such as password aging and expiration. It comes into play after the user is authenticated and determines if the user should be granted access.

authentication Provides authentication through a variety of means including passwords and biometrics. Credentials may be set, refreshed or destroyed.

session Controls things that should be done before or after the user is authenticated. This can include logging, mounting disks or dismounting disks at the close of the session.

password This is the module that allows users to change their passwords.

PAM control flags are used to determine the behavior to be followed if/when authentication fails.

The four PAM control types are:

required Successful only if all modules tagged as required are successful. All modules will be checked and if some checks result in failure, the first failure will be reported.

requisite Failure to authenticate using this module results in the immediate denial of authentication. No additional authentication can occur.

optional Significant only if it is the only module of its type for this service.

sufficient If this control succeeds in authentication, then authentication is successful, and all other controls are skipped, including the ones marked, "required".

The use of these modules and control flags provides flexibility in the way authentication is configured. When a service is called, the PAM configuration entry or file contacts the PAM library which in tern calls the appropriate module. The modularity of PAM allows for several layers of authentication to be used for a particular service.

Figure 3 shows the PAM configuration file for login on a system running RHLinux 7.3. As you can see, most of the control flags are "required" which means that all the modules must pass in order for the user to log in. Figure 4 shows the entries for login taken from a Solaris 8 system.

R is for Record keeping

The last component of login that this paper will cover is login accounting. Record keeping is an important part of any security plan. By default, UNIX systems log successful logins and can be configured to log unsuccessful logins too.

Many UNIX/LINUX systems gather successful login information in log files that have names that end in tmp. In most cases a **u**tmp file/log will contain information about accounts that are currently logged in and a **w**tmp file/log will contain information about successful logins over a period of time. The information in these files is stored as data, and is only viewable using commands written to do so. These are also files that are commonly modified by attackers to cover up a system break in.

> **who** Found on all unix systems. Reads from the .../utmp file to display which accounts are currently logged in. Options are available to make who look at the .../wtmp file or look at both files and do a comparison.
>
> **w** The shortest UNIX command, w, looks at the same database as who but reports not only which accounts are logged on, but what each account is doing.
>
> **last** This command looks at the .../wtmp file and reports on logins over a period since the last time the log was rolled over. Last also reports system crashes, boots and reboots and also shows which users are currently logged in.

Additional login logging may be achieved by configuring various utilities to send error messages to syslogd, the UNIX logging daemon. PAM error messages are delivered to syslogd via the "auth" syslog facility. The syntax for adding this functionality is viewable in Figure 5.

Configuring the "auth" facility provides the added bonus of placing records of attempted network logins in the syslog log. "Auth" is the facility used by the tcpwrappers program which can be used to screen network login attempts.

E is for the End

In a homogeneous setting, in which only one flavor of UNIX is in use, management of system configurations for passwords and logins is fairly straightforward working within the structures made available by whatever operating system is in place. Often, today, more than one UNIX platform exists on any given network and administrators must be savvy with choices they make to implement security policy across the the board.

Password management can take place in the password file, shadow file, PAM subsystem and other places, dependent upon platform. Administrators need to know where all these places are and choose the best options for implementing their plans.

By looking specifically at various aspects of the UNIX, "login" procedure, we have seen several examples of variations in command availability and syntax and the evolution of UNIX through the development of /etc/shadow files from /etc/passwd. Regardless of how seasoned an administrator may be, reading the man pages for all the files and commands that will be part of the secure configuration is a must. UNIX is always changing, sometimes in slow and subtle ways.

---------- references

Aileen Frisch, ***Essential System Administration*** December 1995
O'Reilly and Associates

Rebecca Thomas, Rik Farrow, ***UNIX Administration Guide for System V,*** 1989
Prentice Hall

Chris Hare, "Revisiting UNIX Password Controls", **SysAdmin Journal**, Vol10, Number 10, October 2001.

http://tldp.org/LPD/User_Authentication-HOWTO
Linux Documentation Website

http://docs.sun.com
SUN Solaris Documentation Website

http://docs.sgi.com
SGI IRIX Documentation Website

http://www.linux.org/apps/AppId_2506.html
Download site for Linux-PAM

http://wwws.sun.com/software/solaris/pam/
Sun Site for PAM

Figures:

Figure 1. /etc/passwd

username:password:UID:GID:GECOS:/home/dir:/startup program
jsmith:dfGzpC/Ufda.dz:505:1000: John Smith:/home/jsmith:/bin/bash

username	The name of the account
password	13 character encrypted password
UID	User ID (number)
GID	Group ID (number)
GECOS	Comment Field
/home/dir	Full path to account home directory
/startup/program	Full path to program that will start when user logs in. Typically a shell.

Figure 2. /etc/shadow

```
username:encrypted password:last-change date:min-days:max-days:warning-
days:inactive-day: expiration date:

jsmith:dfGzpC/Ufda.dz:12182::::::
```

username	The name of the account, matches entry in /etc/passwd
password	The encrypted password
last-change date	The number of days since 1/1/70 (start of UNIX time) that the password was changed.
min-days	Minimum number of days that must pass before user can reset password
max-days	Maximum number of days that can pass before user is required to change password.

warning-days Number of days before max-days for the system to warn the user of the impending required password change.

inactive day The account will expire on the actual date specified in this field.

Figure 3. /etc/pam.d/login (RH 7.3)

```
#%PAM-1.0
auth       required     /lib/security/pam_securetty.so
auth       required     /lib/security/pam_stack.so service=system-auth
auth       required     /lib/security/pam_nologin.so
account    required     /lib/security/pam_stack.so service=system-auth
password   required     /lib/security/pam_stack.so service=system-auth
session    required     /lib/security/pam_stack.so service=system-auth
session    optional     /lib/security/pam_console.so
```

Figure 4. /etc/pam.conf (login lines found in pam.conf)(Solaris 8)

```
login auth        required    /usr/lib/security/$ISA/pam_unix.so.1
login auth        required    /usr/lib/security/$ISA/pam_dial_auth.so.1
login account     requisite   /usr/lib/security/$ISA/pam_roles.so.1
login account     required    /usr/lib/security/$ISA/pam_projects.so.1
login account     required    /usr/lib/security/$ISA/pam_unxi.so.1
```

Figure 5. PAM logging in /etc/syslog.conf

auth.alert	/dev/console
auth.crit	root
auth.info;auth.debug	/var/log/pamlog

Last Updated: May 28th, 2017

Upcoming Training

SANS Stockholm 2017	Stockholm, Sweden	May 29, 2017 - Jun 03, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
Community SANS Ottawa SEC401	Ottawa, ON	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Rocky Mountain 2017 - SEC401: Security Essentials Bootcamp Style	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
Community SANS Portland SEC401	Portland, OR	Jun 12, 2017 - Jun 17, 2017	Community SANS
SANS Secure Europe 2017	Amsterdam, Netherlands	Jun 12, 2017 - Jun 20, 2017	Live Event
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Minneapolis SEC401	Minneapolis, MN	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Phoenix SEC401	Phoenix, AZ	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Munich Summer 2017	Munich, Germany	Jul 10, 2017 - Jul 15, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Jul 12, 2017 - Sep 13, 2017	Mentor
Mentor Session - SEC401	Macon, GA	Jul 12, 2017 - Aug 23, 2017	Mentor
Community SANS Colorado Springs SEC401	Colorado Springs, CO	Jul 17, 2017 - Jul 22, 2017	Community SANS
Community SANS Atlanta SEC401	Atlanta, GA	Jul 17, 2017 - Jul 22, 2017	Community SANS
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANSFIRE 2017 - SEC401: Security Essentials Bootcamp Style	Washington, DC	Jul 24, 2017 - Jul 29, 2017	vLive
Community SANS Charleston SEC401	Charleston, SC	Jul 24, 2017 - Jul 29, 2017	Community SANS
Community SANS Fort Lauderdale SEC401	Fort Lauderdale, FL	Jul 31, 2017 - Aug 05, 2017	Community SANS
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event