15.2.5 User Security Facts

Your job as a system administrator includes protecting computer systems and the information they contain from harm, theft, and unauthorized use. Your best security measure is to train users to be security-savvy. There are also commands and configuration files you can use to enhance security.

This lesson covers the following topics:

- Security considerations
- Commands to enhance security
- Options for /etc/security/limits.conf
- PAM overview
- Locking accounts after failed login attempts
- LDAP integration
- tty securetty

Security Considerations

When considering user security, keep the following in mind:

- Users should be trained to use strong passwords. Strong passwords use a mixture of numbers and letters (both upper- and lower-case) and are more than eight characters in length.
- Passwords should expire periodically.
- Administrators can limit the resources users can access.

Commands to Enhance Security

The following table describes commands used to promote user security and restrictions.

Command	Function	Examples
chage	Sets user passwords to expire. Be aware of the following options: - **-M** sets the maximum number of days before the password expires. - **-W** sets the number of days before the password expires that a warning message displays. - **-m** sets the minimum number of days that must pass after a password has been changed before a user can change the password again. Look in the /etc/shadow file to view current limits for users.	**chage -M 60 -W 10 jsmith** sets the password for jsmith to expire after 60 days and gives a warning 10 days before it expires.
ulimit	Limits computer resources used for applications launched from the shell. Limits can be hard or soft limits. Soft limits can be temporarily exceeded up to the hard limit setting. Users can modify soft limits, but only root can modify hard limits. Options include the following: - **-c** limits the size of a core dump file. The value is in blocks. - **-f** limits the file size of files created using the shell session. The value is in blocks. - **-n** limits the maximum number of open files. - **-t** limits the amount of CPU time a process can use. This is set in seconds. - **-u** limits the number of concurrent processes a user can run. - **-d** limits the maximum amount of memory a process can use. The value is in kilobytes. - **-H** sets a hard resource limit. - **-S** sets a soft resource limit. - **-a** displays current limits. The default shows soft limits.	**ulimit -H -f 1024** uses a hard limit to limit the size of files to 1020 KB. **ulimit -H -a** shows current hard limits. **ulimit -a** shows the current soft limits. **ulimit -S -u 10** sets a soft limit that limits the number of processes that a single user can use to 10. **ulimit -t 600** limits CPU time for a process to 10 minutes. This sets both hard and soft limits. **ulimit -d unlimited** removes all restrictions for process memory usage.
last	Displays a list of all users logged in (and out) since the /var/log/wtmp file was created. Options include the following: - **-n *num*** limits the number of lines that last outputs. - **-f filename** reads from the file filename instead of the system's wtmp file. - **-x** prints run level changes, shutdowns, and time changes in addition to the normal records. - **-a** displays the hostname in the last column. - **-i** prints the IP address instead of the hostname.	**last** shows all of the users logged in and out since the /var/log/wtmp file was created. **last -a** displays the hostname in the last column.

	- **-w** instructs last to print out the fields in the wtmp file with full field widths. - **--debug** prints verbose internal information. - **-s** prints seconds when displaying dates. - **-y** prints years when displaying dates. - **-V** prints **last**'s version number.	

Options for /etc/security/limits.conf

Use the **/etc/security/limits.conf** file to limit resource utilization by users. This file is from the pam_limits module of the Pluggable Authentication Modules (PAM) module set. Entries in **/etc/security/limits.conf** use the following syntax: **Domain Type Limit Value**

The following table describes the entry options in the **/etc/security/limits.conf** file:

Domain	Type	Limit	Value
When specifying the Entity: - Specify a single user with a username. - Use an ampersand (@) to specify a group. - Use an asterisk (*) as a wildcard.	For the Type: - Use hard to set a limit that cannot be exceeded. - Use soft to set a limit that can be exceeded temporarily.	Limits include: - core limits the size of core dump files. The value uses kilobytes. - data limits the amount of RAM an application can use. The value uses kilobytes. - fsize limits maximum file size. The value uses kilobytes. - nofile limits the number of concurrently open data files. - stack limits the stack size (in KB). - cpu limits the amount of CPU time a process can use. The value uses minutes. - nproc limits the number of concurrent processes a user can have. - maxlogins limits the number of concurrent logins. - priority sets process priority limits. The value range is from -20 (highest priority) to 19 (lowest priority) with 0 being the default. - nice sets the lowest nice value a user is allowed to set a process to. - rss limits the total amount of memory a user can use. The value uses kilobytes.	Values include integers, such as 1, 5, or 3000.

The following are examples of entries in the /etc/security/limits.conf file:

Example	Description
jsmith hard fsize 10240	Limits the maximum file size that jsmith can create to 10240 KB.
@users hard maxlogins 3	Limits the number of concurrent logins for members of the user group to three.
* hard maxlogins 1	Limits concurrent logins for all users to one.
* hard nice 5	Limits processes for all users to a minimum value of 5.

PAM Overview

The Linux Pluggable Authentication Modules (PAM) provide dynamic authentication support for applications and services in a Linux system, such as **login** and **su**. PAM separates the tasks of authentication into four independent management groups: account management, authentication management, password management, and session management. In other words, these groups take care of the different aspects of a typical user’s request for a restricted service.

- Account modules check that the specified account is a valid authentication target under current conditions. This may include conditions like account expiration, time of day, and that the user has access to the requested service.
- Authentication modules verify the user's identity by requesting and checking a password or other secret. They may also pass authentication information on to other systems like a keyring.
- Password modules are responsible for updating passwords and are generally coupled to modules employed in the authentication step. They may also be used to enforce strong passwords.
- Session modules define actions that are performed at the beginning and end of sessions. A session starts after the user has successfully authenticated.

When using PAM, the system administrator is free to choose how individual service-providing applications will authenticate users. This dynamic configuration is set by the contents of the single Linux-PAM configuration file /etc/pam.conf. Alternatively, the configuration can be set by individual configuration files located in the /etc/pam.d/ directory. The presence of this directory will cause Linux-PAM to ignore /etc/pam.conf.

When working with PAM configuration files, be aware that for the simple syntax, valid control values are as follows:

Syntax	Description
required	Failure of a PAM will ultimately lead to the PAM-API returning failure but only after the remaining stacked modules (for this service and type) have been invoked.
requisite	Requisite is similar to required; however, if a module returns a failure, control is directly returned to the application or to the superior PAM stack.
sufficient	If a module succeeds and no prior required module has failed, the PAM framework returns success to the application or to the superior PAM stack immediately without calling any further modules in the stack. A failure of a sufficient module is ignored, and processing of the PAM module stack continues unaffected.

Locking Accounts After Failed Login Attempts

Often times, hackers or unethical employees attempt to log in using existing accounts that are not their own. Since this is not their account, they will attempt to guess the user's password. This type of attack can be weakened by placing a limit on the number of time the password can be entered incorrectly and after reaching this threshold, locking the account. You can do this using the **pam_tally2** command.

pam_tally2 comes in two parts, which include **pam_tally2.so** (the pam module) and **pam_tally2**, a stand-alone program, that can be used to interrogate and manipulate the counter file.

To lock and unlock an account after a specific number of incorrect password attempts, you must add two lines to both the **/etc/pam.d/system-auth** and the **/etc/pam.d/password-auth** files.

- In the "auth" section of each file, add the following:

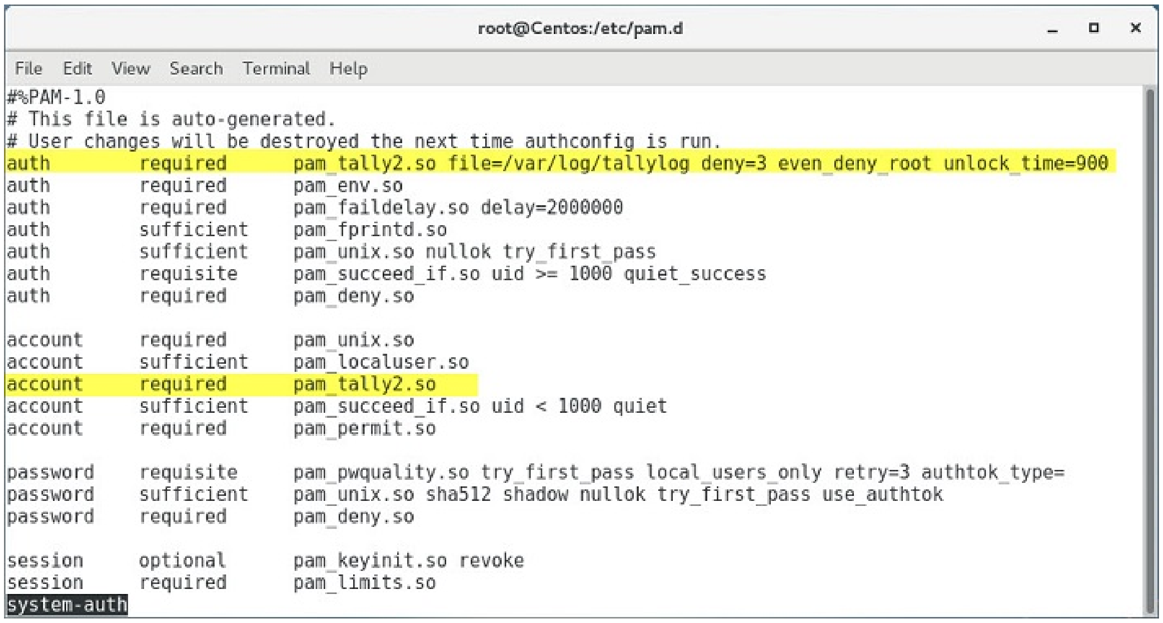
  auth required pam_tally2.so file=/var/log/tallylog deny=3 even_deny_root unlock_time=900

  The options for the above line are defined as follows:

  - **file=/var/log/tallylog** – the default log file where login counts are kept. This is a binary file.
  - **deny=3** – deny access after three attempts and lock the user account.
  - **even_deny_root** – apply this policy for the root user.
  - **unlock_time=900** – after failing the specified number of times, the account will be locked for 900 seconds (15 minutes), after which the account will be unlocked

- In the "account" section of each file add the following:
  account required pam_tally2.so

Example:

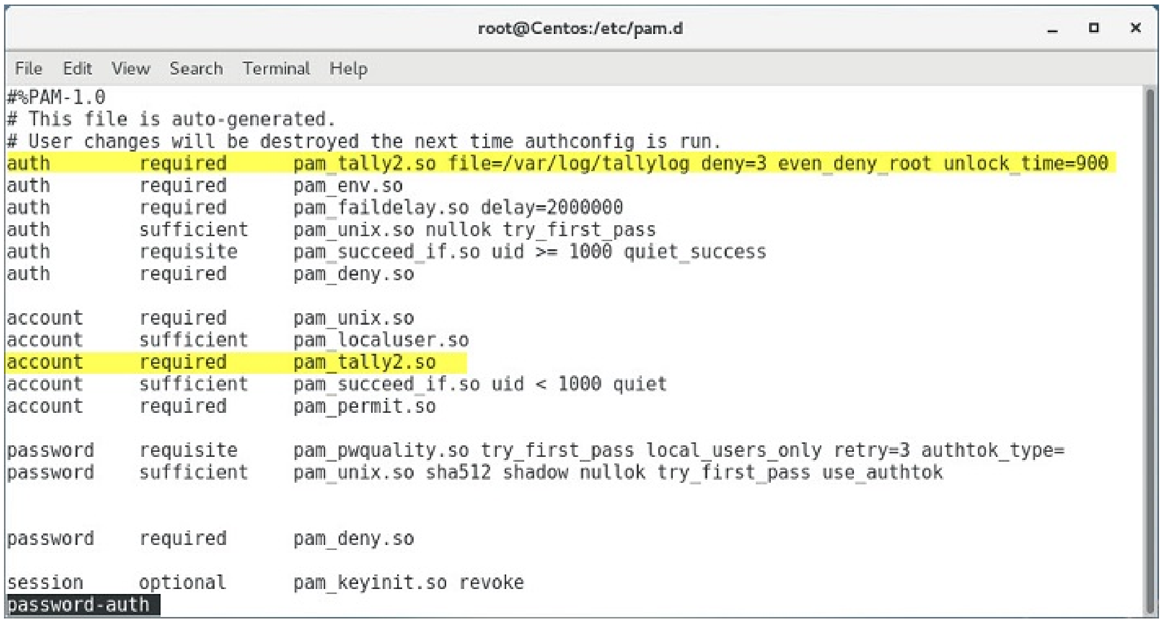


```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_tally2.so file=/var/log/tallylog deny=3 even_deny_root unlock_time=900
auth        required      pam_env.so
auth        required      pam_faildelay.so delay=2000000
auth        sufficient    pam_fprintd.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 1000 quiet_success
auth        required      pam_deny.so

account     required      pam_unix.so
account     sufficient    pam_localuser.so
account     required      pam_tally2.so
account     sufficient    pam_succeed_if.so uid < 1000 quiet
account     required      pam_permit.so

password    requisite     pam_pwquality.so try_first_pass local_users_only retry=3 authtok_type=
password    sufficient    pam_unix.so sha512 shadow nullok try_first_pass use_authtok
password    required      pam_deny.so

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
system-auth
```



```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_tally2.so file=/var/log/tallylog deny=3 even_deny_root unlock_time=900
auth        required      pam_env.so
auth        required      pam_faildelay.so delay=2000000
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 1000 quiet_success
auth        required      pam_deny.so

account     required      pam_unix.so
account     sufficient    pam_localuser.so
account     required      pam_tally2.so
account     sufficient    pam_succeed_if.so uid < 1000 quiet
account     required      pam_permit.so

password    requisite     pam_pwquality.so try_first_pass local_users_only retry=3 authtok_type=
password    sufficient    pam_unix.so sha512 shadow nullok try_first_pass use_authtok


password    required      pam_deny.so

session     optional      pam_keyinit.so revoke
password-auth
```

The commands used in this example were for Centos. How **pam_tally2** is implemented on your distribution may vary.

Using pam_faillock

You can lock a user's account for failed password attempts using the pam_faillock module. This is accomplished in a similar fashion by adding the required lines to the same two files used with **pam_tally2**; **/etc/pam.d/system-auth** and **/etc/pam.d/password-auth**.

Although similar, the commands added are slightly different as shown here:

For both files, add the following in the "auth" section:

auth required pam_faillock.so preauth silent audit deny=3 unlock_time=900
auth [default=die] pam_faillock.so authfail audit deny=3 unlock_time=900

For both files, add the following in the "account" section **account required pam_faillock.so**

The options for the above line are defined as follows:

- **audit –** enables user auditing.
- **deny** – used to define the number of attempts (three in this case), after which the user account should be locked.
- **unlock_time** – sets the time (900 seconds = 15 minutes) for which the account should remain locked.

LDAP Integration

LDAP stands for Lightweight Directory Access Protocol. It is an application protocol used over an IP network to manage and access the distributed directory information service.

When authenticating or authorizing a user, pam_ldap first maps the user's login name to a distinguished name by searching the directory server. This must be possible using the local system's identity, which is specified in pam_ldap.conf.

To authenticate a user, pam_ldap attempts to bind to the directory server using the distinguished name of the user (retrieved previously). Both simple and SASL authentication mechanisms are supported; in the former case, one should take care to use transport security to prevent the user's password being transmitted in the clear.

Pam_ldap stores its configuration in the pam_ldap.conf file. See your man pages for the configuration options available for your distribution.

tty Security

When a root user attempts to log in to a system using tty, the pam_securetty module uses the /etc/securetty file to decide which virtual terminals (tty#) root is allowed to log in from. In older distributions, /etc/securetty was referenced by programs like login directly, but PAM handles this on its own. Therefore, changes to the /etc/securetty file affect anything using PAM with a configuration file that uses pam_securetty.

When using the /etc/securetty file, consider the following:

- If /etc/securetty doesn't exist, root is allowed to login from any tty.
- If /etc/securetty exist and is empty, root access will be restricted to single user mode or programs that are not restricted by pam_securetty (such as su, sudo, ssh, scp, and sftp).
- Removing a tty# (where # is a number) entry from this file will disable a root user from logging on using the associated /dev/tty# terminal.
- A pseudo-terminal (pty) is a pair of virtual character devices that provide a bidirectional communication channel. One end of the channel is called the master; the other end is called the slave. The slave end of the pseudo-terminal provides an interface that behaves exactly like a classic terminal. Adding entries to the /etc/securetty file (such as pts/[0-9]) allow programs that use pseudo-terminals and pam_securetty to log in into root, assuming the allocated pty is one of the ones listed.

Best practice is to exclude these entries because it's a security risk since it would allow, for example, someone to login into root via telnet, which sends passwords in plain text.