5.3 Pluggable Authentication Modules

5.3.1 Configure PAM software packages

Updated versions of PAM include additional functionality

5.3.1.1 Ensure latest version of pam is installed (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

Updated versions of PAM include additional functionality

Rationale:

To ensure the system has full functionality and access to the options covered by this Benchmark the latest version of `libpam-runtime` should be installed on the system

Audit:

Run the following command to verify the version of `libpam-runtime` on the system:

```
# dpkg-query -s libpam-runtime | grep -P -- '^(Status|Version)\b'
```

The output should be similar to:

```
Status: install ok installed
Version: 1.5.3-5
```

Remediation:

- IF - the version of `libpam-runtime` on the system is less than version `1.5.3-5`:
Run the following command to update to the latest version of `PAM`:

```
# apt upgrade libpam-runtime
```

5.3.1.2 Ensure libpam-modules is installed (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

Pluggable Authentication Modules for PAM

Rationale:

To ensure the system has full functionality and access to the PAM options covered by this Benchmark

Audit:

Run the following command to verify `libpam-modules` is installed and version `1.5.3-5` or later:

```
# dpkg-query -s libpam-modules | grep -P -- '^(Status|Version)\b'
```

The output should be similar to:

```
Status: install ok installed
Version: 1.5.3-5
```

Remediation:

- IF - the version of `libpam-modules` on the system is less than version `1.5.3-5`:
Run the following command to update to the latest version of `PAM`:

```
# apt upgrade libpam-modules
```

5.3.1.3 Ensure libpam-pwquality is installed (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

`libpwquality` provides common functions for password quality checking and scoring them based on their apparent randomness. The library also provides a function for generating random passwords with good pronounceability.

This module can be plugged into the password stack of a given service to provide some plug-in strength-checking for passwords. The code was originally based on `pam_cracklib` module and the module is backwards compatible with its options.

Rationale:

Strong passwords reduce the risk of systems being hacked through brute force methods.

Audit:

Run the following command to verify `libpam-pwquality` is installed:

```
# dpkg-query -s libpam-pwquality | grep -P -- '^(Status|Version)\b'
```

The output should be similar to:

```
Status: install ok installed
Version: 1.4.5-3+build1
```

Remediation:

Run the following command to install `libpam-pwquality`:

```
# apt install libpam-pwquality
```

References:

1. https://packages.debian.org/buster/libpam-pwquality

5.3.2 Configure pam-auth-update profiles

`pam-auth-update` is a utility that permits configuring the central authentication policy for the system using pre-defined profiles as supplied by PAM module packages.

Profiles - Shipped in the `/usr/share/pam-configs/` directory specify the modules, with options, to enable; the preferred ordering with respect to other profiles; and whether a profile should be enabled by default. Packages providing PAM modules register their profiles at install time by calling pam-auth-update --package.

Selection of profiles is done using the standard `debconf` interface. The profile selection question will be asked at `medium` priority when packages are added or removed, so no user interaction is required by default. Users may invoke `pam-auth-update` directly to change their authentication configuration.

The `pam-auth-update` script makes every effort to respect local changes to `/etc/pam.d/common-*.` Local modifications to the list of module options will be preserved, and additions of modules within the managed portion of the stack will cause pam-auth-update to treat the config files as locally modified and not make further changes to the config files unless given the `--force` option.

If the user specifies that `pam-auth-update` should override local configuration changes, the locally-modified files will be saved in /etc/pam.d/ with a suffix of `.pam-old`.

5.3.2.1 Ensure pam_unix module is enabled (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

`pam_unix` is the standard Unix authentication module. It uses standard calls from the system's libraries to retrieve and set account information as well as authentication. Usually this is obtained from the `/etc/passwd` and if shadow is enabled, the `/etc/shadow` file as well.

The account component performs the task of establishing the status of the user's account and password based on the following shadow elements: `expire`, `last_change`, `max_change`, `min_change`, `warn_change`. In the case of the latter, it may offer advice to the user on changing their password or, through the `PAM_AUTHTOKEN_REQD` return, delay giving service to the user until they have established a new password. The entries listed above are documented in the shadow(5) manual page. Should the user's record not contain one or more of these entries, the corresponding shadow check is not performed.

The authentication component performs the task of checking the users credentials (password). The default action of this module is to not permit the user access to a service if their official password is blank.

Rationale:

The system should only provide access after performing authentication of a user.

Audit:

Run the following command to verify that `pam_unix` is enabled:

```
# grep -P -- '\bpam_unix\.so\b' /etc/pam.d/common-
{account,session,auth,password}
```

Output should be simular to:

```
/etc/pam.d/common-account:account   [success=1 new_authtok_reqd=done
default=ignore]   pam_unix.so
/etc/pam.d/common-session:session   required   pam_unix.so
/etc/pam.d/common-auth:auth   [success=2 default=ignore]   pam_unix.so
try_first_pass
/etc/pam.d/common-password:password   [success=1 default=ignore]
pam_unix.so obscure use_authtok try_first_pass yescrypt
```

Remediation:

Run the following command to enable the `pam_unix` module:

```
# pam-auth-update --enable unix
```

Note: If a site specific custom profile is being used in your environment to configure PAM that includes the configuration for the `pam_faillock` module, enable that module instead

References:

1. NIST SP 800-53 Rev. 5: IA-5(1)

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	14.6 Protect Information through Access Control Lists Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1110, T1110.001, T1110.002, T1110.003, T1178.001, T1178.002, T1178.003, T1178.004	TA0006	M1027

5.3.2.2 Ensure pam_faillock module is enabled (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The `pam_faillock.so` module maintains a list of failed authentication attempts per user during a specified interval and locks the account in case there were more than the configured number of consecutive failed authentications (this is defined by the `deny` parameter in the faillock configuration). It stores the failure records into per-user files in the tally directory.

Rationale:

Locking out user IDs after n unsuccessful consecutive login attempts mitigates brute force password attacks against your systems.

Audit:

Run the following commands to verify that `pam_faillock` is enabled:

```
# grep -P -- '\bpam_faillock\.so\b' /etc/pam.d/common-{auth,account}
```

Output should be similar to:

```
/etc/pam.d/common-auth:auth     requisite
pam_faillock.so preauth
/etc/pam.d/common-auth:auth     [default=die]
pam_faillock.so authfail
/etc/pam.d/common-account:account       required
pam_faillock.so
```

Remediation:

Create two pam-auth-update profiles in `/usr/share/pam-configs/`:

1. Create the `faillock` profile in `/usr/share/pam-configs/` with the following lines:

```
Name: Enable pam_faillock to deny access
Default: yes
Priority: 0
Auth-Type: Primary
Auth:
        [default=die]                   pam_faillock.so authfail
```

Example Script:

```
#!/usr/bin/env bash

{
   arr=('Name: Enable pam_faillock to deny access' 'Default: yes' 'Priority:
0' 'Auth-Type: Primary' 'Auth:' '        [default=die]
pam_faillock.so authfail')
   printf '%s\n' "${arr[@]}" > /usr/share/pam-configs/faillock
}
```

2. Create the `faillock_notify` profile in `/usr/share/pam-configs/` with the following lines:

```
Name: Notify of failed login attempts and reset count upon success
Default: yes
Priority: 1024
Auth-Type: Primary
Auth:
        requisite                       pam_faillock.so preauth
Account-Type: Primary
Account:
        required                        pam_faillock.so
```

Example Script:

```
#!/usr/bin/env bash

{
   arr=('Name: Notify of failed login attempts and reset count upon success'
'Default: yes' 'Priority: 1024' 'Auth-Type: Primary' 'Auth:' '
requisite                       pam_faillock.so preauth' 'Account-Type:
Primary' 'Account:' '        required
pam_faillock.so')
   printf '%s\n' "${arr[@]}" > /usr/share/pam-configs/faillock_notify
}
```

Run the following command to update the `common-auth` and `common-account` PAM files with the new profiles:

```
# pam-auth-update --enable <profile_filename>
```

Example:

```
# pam-auth-update --enable faillock
# pam-auth-update --enable faillock_notify
```

Note:

- The name used for the file must be used in the `pam-auth-update --enable` command
- The `Name:` line should be easily recognizable and understood
- The `Priority:` Line is important as it effects the order of the lines in the `/etc/pam.d/` files
- If a site specific custom profile is being used in your environment to configure PAM that includes the configuration for the `pam_faillock` module, enable that module instead

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.2 Establish an Access Revoking Process Establish and follow a process, preferably automated, for revoking access to enterprise assets, through disabling accounts immediately upon termination, rights revocation, or role change of a user. Disabling accounts, instead of deleting accounts, may be necessary to preserve audit trails.	●	●	●
v7	16.7 Establish Process for Revoking Access Establish and follow an automated process for revoking system access by disabling accounts immediately upon termination or change of responsibilities of an employee or contractor . Disabling these accounts, instead of deleting accounts, allows preservation of audit trails.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1110, T1110.001, T1110.003	TA0006	M1027

5.3.2.3 Ensure pam_pwquality module is enabled (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The `pam_pwquality.so` module performs password quality checking. This module can be plugged into the password stack of a given service to provide strength-checking for passwords. The code was originally based on pam_cracklib module and the module is backwards compatible with its options.

The action of this module is to prompt the user for a password and check its strength against a system dictionary and a set of rules for identifying poor choices.

The first action is to prompt for a single password, check its strength and then, if it is considered strong, prompt for the password a second time (to verify that it was typed correctly on the first occasion). All being well, the password is passed on to subsequent modules to be installed as the new authentication token.

Rationale:

Use of a unique, complex passwords helps to increase the time and resources required to compromise the password.

Audit:

Run the following command to verify that pam_pwhistory is enabled:

```
# grep -P -- '\bpam_pwquality\.so\b' /etc/pam.d/common-password
```

Output should be similar to:

```
password   requisite   pam_pwquality.so retry=3
```

Remediation:

Run the following script to verify the `pam_pwquality.so` line exists in a `pam-auth-update` profile:

```
# grep -P -- '\bpam_pwquality\.so\b' /usr/share/pam-configs/*
```

Output should be similar to:

```
/usr/share/pam-configs/pwquality:       requisite
pam_pwquality.so retry=3
/usr/share/pam-configs/pwquality:       requisite
pam_pwquality.so retry=3
```

- IF - similar output is returned:
Run the following command to update `/etc/pam.d/common-password` with the returned profile:

```
# pam-auth-update --enable {PROFILE_NAME}
```

Example:

```
# pam-auth-update pwquality
```

- IF - similar output is **NOT** returned:
Create a pam-auth-update profile in `/usr/share/pam-configs/` with the following lines:

```
Name: Pwquality password strength checking
Default: yes
Priority: 1024
Conflicts: cracklib
Password-Type: Primary
Password:
        requisite                       pam_pwquality.so retry=3
```

Example:

```
#!/usr/bin/env bash

{
   arr=('Name: Pwquality password strength checking' 'Default: yes'
'Priority: 1024' 'Conflicts: cracklib' 'Password-Type: Primary' 'Password:' '
requisite                       pam_pwquality.so retry=3')
   printf '%s\n' "${arr[@]}" > /usr/share/pam-configs/pwquality
}
```

Run the following command to update `/etc/pam.d/common-password` with the `pwquality` profile:

```
# pam-auth-update --enable pwquality
```

Note:

- The name used for the file must be used in the `pam-auth-update --enable` command
- The `Name:` line should be easily recognizable and understood
- The `Priority:` Line is important as it effects the order of the lines in the `/etc/pam.d/` files
- If a site specific custom profile is being used in your environment to configure PAM that includes the configuration for the `pam_pwquality` module, enable that module instead

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.2 Use Unique Passwords Use unique passwords for all enterprise assets. Best practice implementation includes, at a minimum, an 8-character password for accounts using MFA and a 14-character password for accounts not using MFA.	●	●	●
v7	4.4 Use Unique Passwords Where multi-factor authentication is not supported (such as local administrator, root, or service accounts), accounts will use passwords that are unique to that system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1110, T1110.001, T1110.002, T1110.003, T1178.001, T1178.002, T1178.003, T1178.004	TA0006	M1027

5.3.2.4 Ensure pam_pwhistory module is enabled (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The `pam_pwhistory.so` module saves the last passwords for each user in order to force password change history and keep the user from alternating between the same password too frequently.

This module does not work together with kerberos. In general, it does not make much sense to use this module in conjunction with `NIS` or `LDAP`, since the old passwords are stored on the local machine and are not available on another machine for password history checking.

Rationale:

Use of a unique, complex passwords helps to increase the time and resources required to compromise the password.

Audit:

Run the following command to verify that pam_pwhistory is enabled:

```
# grep -P -- '\bpam_pwhistory\.so\b' /etc/pam.d/common-password
```

Output should be similar to:

```
password   requisite   pam_pwhistory.so remember=24 enforce_for_root
try_first_pass use_authtok
```

Remediation:

Run the following script to verify the `pam_pwquality.so` line exists in a `pam-auth-update` profile:

```
# grep -P -- '\bpam_pwhistory\.so\b' /usr/share/pam-configs/*
```

Output should be similar to:

```
/usr/share/pam-configs/pwhistory:   requisite   pam_pwhistory.so remember=24
enforce_for_root try_first_pass use_authtok
```

- IF - similar output is returned:
Run the following command to update `/etc/pam.d/common-password` with the returned profile:

```
# pam-auth-update --enable {PROFILE_NAME}
```

Example:

```
# pam-auth-update pwhistory
```

- IF - similar output is **NOT** returned:
Create a `pwhistory` profile in `/usr/share/pam-configs/` with the following lines:

```
Name: pwhistory password history checking
Default: yes
Priority: 1024
Password-Type: Primary
Password: requisite pam_pwhistory.so remember=24 enforce_for_root
try_first_pass use_authtok
```

Example Script:

```
#!/usr/bin/env bash

{
   arr=('Name: pwhistory password history checking' 'Default: yes' 'Priority:
1024' 'Password-Type: Primary' 'Password:' '        requisite
pam_pwhistory.so remember=24 enforce_for_root try_first_pass use_authtok')
   printf '%s\n' "${arr[@]}" > /usr/share/pam-configs/pwhistory
}
```

Run the following command to update `/etc/pam.d/common-password` with the `pwhistory` profile:

```
# pam-auth-update --enable pwhistory
```

Note:

- The name used for the file must be used in the `pam-auth-update --enable` command
- The `Name:` line should be easily recognizable and understood
- The `Priority:` Line is important as it effects the order of the lines in the `/etc/pam.d/` files
- If a site specific custom profile is being used in your environment to configure PAM that includes the configuration for the `pam_pwhistory` module, enable that module instead

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.2 Use Unique Passwords Use unique passwords for all enterprise assets. Best practice implementation includes, at a minimum, an 8-character password for accounts using MFA and a 14-character password for accounts not using MFA.	●	●	●
v7	4.4 Use Unique Passwords Where multi-factor authentication is not supported (such as local administrator, root, or service accounts), accounts will use passwords that are unique to that system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1110, T1110.001, T1110.002, T1110.003, T1178.001, T1178.002, T1178.003, T1178.004	TA0006	M1027

5.4 User Accounts and Environment

This section provides guidance on setting up secure defaults for system and user accounts and their environment.

5.4.1 Configure shadow password suite parameters

While a majority of the password control parameters have been moved to PAM, some parameters are still available through the shadow password suite. Any changes made to `/etc/login.defs` will only be applied if the `usermod` command is used. If user IDs are added a different way, use the `chage` command to effect changes to individual user IDs.

5.4.1.1 Ensure password expiration is configured (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The `PASS_MAX_DAYS` parameter in `/etc/login.defs` allows an administrator to force passwords to expire once they reach a defined age.

`PASS_MAX_DAYS` <*N*> - The maximum number of days a password may be used. If the password is older than this, a password change will be forced. If not specified, -1 will be assumed (which disables the restriction).

Rationale:

The window of opportunity for an attacker to leverage compromised credentials or successfully compromise credentials via an online brute force attack is limited by the age of the password. Therefore, reducing the maximum age of a password also reduces an attacker's window of opportunity.

We recommend a yearly password change. This is primarily because for all their good intentions users will share credentials across accounts. Therefore, even if a breach is publicly identified, the user may not see this notification, or forget they have an account on that site. This could leave a shared credential vulnerable indefinitely. Having an organizational policy of a 1-year (annual) password expiration is a reasonable compromise to mitigate this with minimal user burden.

Impact:

The password expiration must be greater than the minimum days between password changes or users will be unable to change their password.

Excessive password expiration requirements do more harm than good, because these requirements make users select predictable passwords, composed of sequential words and numbers that are closely related to each other. In these cases, the next password can be predicted based on the previous one (incrementing a number used in the password for example). Also, password expiration requirements offer no containment benefits because attackers will often use credentials as soon as they compromise them. Instead, immediate password changes should be based on key events including, but not limited to:

- Indication of compromise
- Change of user roles
- When a user leaves the organization.

Not only does changing passwords every few weeks or months frustrate the user, but it's also been suggested that it does more harm than good, because it could lead to bad practices by the user such as adding a character to the end of their existing password.

Audit:

Run the following command and verify `PASS_MAX_DAYS` is set to 365 days or less and conforms to local site policy:

```
# grep -Pi -- '^\h*PASS_MAX_DAYS\h+\d+\b' /etc/login.defs
```

Example output:

```
PASS_MAX_DAYS 365
```

Run the following command to verify all `/etc/shadow` passwords `PASS_MAX_DAYS`:

- is greater than `0` days
- is less than or equal to `365` days
- conforms to local site policy

```
# awk -F: '($2~/^\$.+\$/) {if($5 > 365 || $5 < 1)print "User: " $1 "
PASS_MAX_DAYS: " $5}' /etc/shadow
```

Nothing should be returned

Remediation:

Set the `PASS_MAX_DAYS` parameter to conform to site policy in `/etc/login.defs` :

```
PASS_MAX_DAYS 365
```

Modify user parameters for all users with a password set to match:

```
# chage --maxdays 365 <user>
```

Edit `/etc/login.defs` and set `PASS_MAX_DAYS` to a value greater than `0` that follows local site policy:
Example:

```
PASS_MAX_DAYS 365
```

Run the following command to modify user parameters for all users with a password set to a maximum age no greater than `365` or less than `1` that follows local site policy:

```
# chage --maxdays <N> <user>
```

Example:

```
# awk -F: '($2~/^\$.+\$/) {if($5 > 365 || $5 < 1)system ("chage --maxdays 365
" $1)}' /etc/shadow
```

Warning: If a password has been set at system install or kickstart, the `last change date` field is not set, In this case, setting `PASS_MAX_DAYS` will immediately expire the password. One possible solution is to populate the `last change date` field through a command like: `chage -d "$(date +%Y-%m-%d)" root`

Default Value:

PASS_MAX_DAYS 99999

References:

1. CIS Password Policy Guide
2. NIST SP 800-53 Rev. 5: CM-1, CM-2, CM-6, CM-7, IA-5

Additional Information:

A value of -1 will disable password expiration.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.2 Use Unique Passwords Use unique passwords for all enterprise assets. Best practice implementation includes, at a minimum, an 8-character password for accounts using MFA and a 14-character password for accounts not using MFA.	●	●	●
v7	4.4 Use Unique Passwords Where multi-factor authentication is not supported (such as local administrator, root, or service accounts), accounts will use passwords that are unique to that system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078, T1078.001, T1078.002, T1078.003, T1078.004, T1110, T1110.001, T1110.002, T1110.003, T1110.004		

5.4.1.2 Ensure minimum password days is configured (Manual)

Profile Applicability:

- Level 2 - Server
- Level 2 - Workstation

Description:

`PASS_MIN_DAYS` <*N*> - The minimum number of days allowed between password changes. Any password changes attempted sooner than this will be rejected. If not specified, 0 will be assumed (which disables the restriction).

Rationale:

Users may have favorite passwords that they like to use because they are easy to remember, and they believe that their password choice is secure from compromise. Unfortunately, passwords are compromised and if an attacker is targeting a specific individual user account, with foreknowledge of data about that user, reuse of old, potentially compromised passwords, may cause a security breach.

By restricting the frequency of password changes, an administrator can prevent users from repeatedly changing their password in an attempt to circumvent password reuse controls

Impact:

If a user's password is set by other personnel as a procedure in dealing with a lost or expired password, the user should be forced to update this "set" password with their own password. e.g. force "change at next logon".

If it is not possible to have a user set their own password immediately, and this recommendation or local site procedure may cause a user to continue using a third party generated password, `PASS_MIN_DAYS` for the effected user should be temporally changed to `0`, to allow a user to change their password immediately.

For applications where the user is not using the password at console, the ability to "change at next logon" may be limited. This may cause a user to continue to use a password created by other personnel.

Audit:

Run the following command to verify that PASS_MIN_DAYS is set to a value greater than 0and follows local site policy:

```
# grep -Pi -- '^\h*PASS_MIN_DAYS\h+\d+\b' /etc/login.defs
```

Example output:

```
PASS_MIN_DAYS   1
```

Run the following command to verify all passwords have a PASS_MIN_DAYS greater than 0:

```
# awk -F: '($2~/^\$.+\$/) {if($4 < 1)print "User: " $1 " PASS_MIN_DAYS: "
$4}' /etc/shadow
```

Nothing should be returned

Remediation:

Edit /etc/login.defs and set PASS_MIN_DAYS to a value greater than 0 that follows local site policy:
Example:

```
PASS_MIN_DAYS 1
```

Run the following command to modify user parameters for all users with a password set to a minimum days greater than zero that follows local site policy:

```
# chage --mindays <N> <user>
```

Example:

```
# awk -F: '($2~/^\$.+\$/) {if($4 < 1)system ("chage --mindays 1 " $1)}'
/etc/shadow
```

Default Value:

PASS_MIN_DAYS 0

References:

1. CIS Password Policy Guide

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.2 Use Unique Passwords Use unique passwords for all enterprise assets. Best practice implementation includes, at a minimum, an 8-character password for accounts using MFA and a 14-character password for accounts not using MFA.	●	●	●
v7	4.4 Use Unique Passwords Where multi-factor authentication is not supported (such as local administrator, root, or service accounts), accounts will use passwords that are unique to that system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078, T1078.001, T1078.002, T1078.003, T1078.004, T1110, T1110.004	TA0006	M1027

5.4.1.3 Ensure password expiration warning days is configured (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The PASS_WARN_AGE parameter in /etc/login.defs allows an administrator to notify users that their password will expire in a defined number of days.

PASS_WARN_AGE <*N*> - The number of days warning given before a password expires. A zero means warning is given only upon the day of expiration, a negative value means no warning is given. If not specified, no warning will be provided.

Rationale:

Providing an advance warning that a password will be expiring gives users time to think of a secure password. Users caught unaware may choose a simple password or write it down where it may be discovered.

Audit:

Run the following command and verify PASS_WARN_AGE is 7 or more and follows local site policy:

```
# grep -Pi -- '^\h*PASS_WARN_AGE\h+\d+\b' /etc/login.defs
```

Example output:

```
PASS_WARN_AGE 7
```

Run the following command to verify all passwords have a PASS_WARN_AGE of 7 or more:

```
# awk -F: '($2~/^\$.+\$/) {if($6 < 7)print "User: " $1 " PASS_WARN_AGE: "
$6}' /etc/shadow
```

Nothing should be returned

Remediation:

Edit `/etc/login.defs` and set `PASS_WARN_AGE` to a value of `7` or more that follows local site policy:
Example:

```
PASS_WARN_AGE 7
```

Run the following command to modify user parameters for all users with a password set to a minimum warning to `7` or more days that follows local site policy:

```
# chage --warndays <N> <user>
```

Example:

```
# awk -F: '($2~/^\$.+\$/) {if($6 < 7)system ("chage --warndays 7 " $1)}'
/etc/shadow
```

Default Value:

PASS_WARN_AGE 7

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.1 Establish and Maintain a Secure Configuration Process Establish and maintain a secure configuration process for enterprise assets (end-user devices, including portable and mobile, non-computing/IoT devices, and servers) and software (operating systems and applications). Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v7	4.4 Use Unique Passwords Where multi-factor authentication is not supported (such as local administrator, root, or service accounts), accounts will use passwords that are unique to that system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078	TA0006	M1027

5.4.1.4 Ensure strong password hashing algorithm is configured (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

A cryptographic hash function converts an arbitrary-length input into a fixed length output. Password hashing performs a one-way transformation of a password, turning the password into another string, called the hashed password.

`ENCRYPT_METHOD` (string) - This defines the system default encryption algorithm for encrypting passwords (if no algorithm are specified on the command line). It can take one of these values:

- `MD5` - MD5-based algorithm will be used for encrypting password
- `SHA256` - SHA256-based algorithm will be used for encrypting password
- `SHA512` - SHA512-based algorithm will be used for encrypting password
- `BCRYPT` - BCRYPT-based algorithm will be used for encrypting password
- `YESCRYPT` - YESCRYPT-based algorithm will be used for encrypting password
- `DES` - DES-based algorithm will be used for encrypting password (default)

Note:

- This parameter overrides the deprecated `MD5_CRYPT_ENAB` variable.
- This parameter will only affect the generation of group passwords.
- The generation of user passwords is done by PAM and subject to the PAM configuration.
- It is recommended to set this variable consistently with the PAM configuration.

Rationale:

The `SHA-512` and `yescrypt` algorithms provide a stronger hash than other algorithms used by Linux for password hash generation. A stronger hash provides additional protection to the system by increasing the level of effort needed for an attacker to successfully determine local group passwords.

Audit:

Run the following command to verify the hashing algorithm is `sha512` or `yescrypt` in `/etc/login.defs`:

```
# grep -Pi -- '^\h*ENCRYPT_METHOD\h+(SHA512|yescrypt)\b' /etc/login.defs
```

Example output:

```
ENCRYPT_METHOD SHA512
 - OR -
ENCRYPT_METHOD  YESCRYPT
```

Remediation:

Edit `/etc/login.defs` and set the `ENCRYPT_METHOD` to `SHA512` or `YESCRYPT`:

```
ENCRYPT_METHOD <HASHING_ALGORITHM>
```

Example:

```
ENCRYPT_METHOD YESCRYPT
```

Note:

- This only effects local groups' passwords created after updating the file to use `sha512` or `yescrypt`.
- If it is determined that the password algorithm being used is not `sha512` or `yescrypt`, once it is changed, it is recommended that all group passwords be updated to use the stronger hashing algorithm.
- It is recommended that the chosen hashing algorithm is consistent across `/etc/login.defs` and the PAM configuration

Default Value:

ENCRYPT_METHOD SHA512

References:

1. NIST SP 800-53 Rev. 5: IA-5

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.11 Encrypt Sensitive Data at Rest Encrypt sensitive data at rest on servers, applications, and databases containing sensitive data. Storage-layer encryption, also known as server-side encryption, meets the minimum requirement of this Safeguard. Additional encryption methods may include application-layer encryption, also known as client-side encryption, where access to the data storage device(s) does not permit access to the plain-text data.		●	●
v7	16.4 Encrypt or Hash all Authentication Credentials Encrypt or hash with a salt all authentication credentials when stored.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1003, T1003.008, T1110, T1110.002	TA0006	M1041

5.4.1.5 Ensure inactive password lock is configured (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

User accounts that have been inactive for over a given period of time can be automatically disabled.

`INACTIVE` - Defines the number of days after the password exceeded its maximum age where the user is expected to replace this password.

The value is stored in the shadow password file. An input of `0` will disable an expired password with no delay. An input of `-1` will blank the respective field in the shadow password file.

Rationale:

Inactive accounts pose a threat to system security since the users are not logging in to notice failed login attempts or other anomalies.

Audit:

Run the following command and verify `INACTIVE` conforms to site policy (no more than 45 days):

```
# useradd -D | grep INACTIVE

INACTIVE=45
```

Verify all users with a password have Password inactive no more than 45 days after password expires
Verify all users with a password have Password inactive no more than 45 days after password expires: Run the following command and Review list of users and `INACTIVE` to verify that all users `INACTIVE` conforms to site policy (no more than 45 days):

```
# awk -F: '($2~/^\$.+\$/) {if($7 > 45 || $7 < 0)print "User: " $1 " INACTIVE:
" $7}' /etc/shadow
```

Nothing should be returned

Remediation:

Run the following command to set the default password inactivity period to 45 days or less that meets local site policy:

```
# useradd -D -f <N>
```

Example:

```
# useradd -D -f 45
```

Run the following command to modify user parameters for all users with a password set to a inactive age of 45 days or less that follows local site policy:

```
# chage --inactive <N> <user>
```

Example:

```
# awk -F: '($2~/^\$.+\$/) {if($7 > 45 || $7 < 0)system ("chage --inactive 45
" $1)}' /etc/shadow
```

Default Value:

INACTIVE=-1

References:

1. CIS Password Policy Guide

Additional Information:

A value of -1 would disable this setting.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.2 Use Unique Passwords Use unique passwords for all enterprise assets. Best practice implementation includes, at a minimum, an 8-character password for accounts using MFA and a 14-character password for accounts not using MFA.	●	●	●
v7	4.4 Use Unique Passwords Where multi-factor authentication is not supported (such as local administrator, root, or service accounts), accounts will use passwords that are unique to that system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078, T1078.002, T1078.003	TA0001	M1027

5.4.1.6 Ensure all users last password change date is in the past (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

All users should have a password change date in the past.

Rationale:

If a user's recorded password change date is in the future, then they could bypass any set password expiration.

Audit:

Run the following command and verify nothing is returned

```
{
   while IFS= read -r l_user; do
      l_change=$(date -d "$(chage --list $l_user | grep '^Last password
change' | cut -d: -f2 | grep -v 'never$')" +%s)
      if [[ "$l_change" -gt "$(date +%s)" ]]; then
         echo "User: \"$l_user\" last password change was \"$(chage --list
$l_user | grep '^Last password change' | cut -d: -f2)\""
      fi
   done < <(awk -F: '$2~/^\$.+\$/{print $1}' /etc/shadow)
}
```

Remediation:

Investigate any users with a password change date in the future and correct them. Locking the account, expiring the password, or resetting the password manually may be appropriate.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.2 Use Unique Passwords Use unique passwords for all enterprise assets. Best practice implementation includes, at a minimum, an 8-character password for accounts using MFA and a 14-character password for accounts not using MFA.	●	●	●
v7	4.4 Use Unique Passwords Where multi-factor authentication is not supported (such as local administrator, root, or service accounts), accounts will use passwords that are unique to that system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078, T1078.001, T1078.002, T1078.003, T1078.004, T1110, T1110.001, T1110.002, T1110.003, T1110.004		

5.4.2 Configure root and system accounts and environment

5.4.2.1 Ensure root is the only UID 0 account (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

Any account with UID 0 has superuser privileges on the system.

Rationale:

This access must be limited to only the default `root` account and only from the system console. Administrative access must be through an unprivileged account using an approved mechanism as noted in Item 5.6 Ensure access to the su command is restricted.

Audit:

Run the following command and verify that only "root" is returned:

```
# awk -F: '($3 == 0) { print $1 }' /etc/passwd

root
```

Remediation:

Run the following command to change the `root` account UID to `0`:

```
# usermod -u 0 root
```

Modify any users other than `root` with UID `0` and assign them a new UID.

References:

1. NIST SP 800-53 Rev. 5: CM-1, CM-2, CM-6, CM-7, IA-5

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1548, T1548.000	TA0001	M1026

5.4.2.2 Ensure root is the only GID 0 account (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The `usermod` command can be used to specify which group the `root` account belongs to. This affects permissions of files that are created by the `root` account.

Rationale:

Using GID 0 for the `root` account helps prevent `root` -owned files from accidentally becoming accessible to non-privileged users.

Audit:

Run the following command to verify the `root` user's primary GID is `0`, and no other user's have GID `0` as their primary GID:

```
# awk -F: '($1 !~ /^(sync|shutdown|halt|operator)/ && $4=="0") {print
$1":"$4}' /etc/passwd

root:0
```

Note: User's: sync, shutdown, halt, and operator are excluded from the check for other user's with GID `0`

Remediation:

Run the following command to set the `root` user's GID to `0`:

```
# usermod -g 0 root
```

Run the following command to set the `root` group's GID to `0`:

```
# groupmod -g 0 root
```

Remove any users other than the `root` user with GID 0 or assign them a new GID if appropriate.

References:

1. NIST SP 800-53 Rev. 5: CM-1, CM-2, CM-6, CM-7, IA-5

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	14.6 Protect Information through Access Control Lists Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1548, T1548.000	TA0005	M1026

5.4.2.3 Ensure group root is the only GID 0 group (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The `groupmod` command can be used to specify which group the `root` group belongs to. This affects permissions of files that are group owned by the `root` group.

Rationale:

Using GID 0 for the `root` group helps prevent `root` group owned files from accidentally becoming accessible to non-privileged users.

Audit:

Run the following command to verify no group other than `root` is assigned GID `0`:

```
# awk -F: '$3=="0"{print $1":"$3}' /etc/group

root:0
```

Remediation:

Run the following command to set the `root` group's GID to `0`:

```
# groupmod -g 0 root
```

Remove any groups other than the `root` group with GID 0 or assign them a new GID if appropriate.

References:

1. NIST SP 800-53 Rev. 5: CM-1, CM-2, CM-6, CM-7, IA-5

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	14.6 Protect Information through Access Control Lists Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1548, T1548.000	TA0005	M1026

5.4.2.4 Ensure root account access is controlled (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

There are a number of methods to access the root account directly. Without a password set any user would be able to gain access and thus control over the entire system.

Rationale:

Access to `root` should be secured at all times.

Impact:

If there are any automated processes that relies on access to the root account without authentication, they will fail after remediation.

Audit:

Run the following command to verify that either the root user's password is set or the root user's account is locked:

```
# passwd -S root | awk '$2 ~ /^(P|L)/ {print "User: \"" $1 "\" Password is
status: " $2}'
```

Verify the output is either:

```
User: "root" Password is status: P
- OR -
User: "root" Password is status: L
```

Note:

- `P` - Password is set
- `L` - Password is locked

Remediation:

Run the following command to set a password for the `root` user:

```
# passwd root
```

- OR -

Run the following command to lock the `root` user account:

```
# usermod -L root
```

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	14.6 Protect Information through Access Control Lists Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078	TA0005	M1026

5.4.2.5 Ensure root path integrity (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The `root` user can execute any command on the system and could be fooled into executing programs unintentionally if the `PATH` is not set correctly.

Rationale:

Including the current working directory (.) or other writable directory in `root`'s executable path makes it likely that an attacker can gain superuser access by forcing an administrator operating as `root` to execute a Trojan horse program.

Audit:

Run the following script to verify root's path does not include:

- Locations that are not directories
- An empty directory (`::`)
- A trailing (`:`)
- Current working directory (`.`)
- Non `root` owned directories
- Directories that less restrictive than mode `0755`

```
#!/usr/bin/env bash

{
   l_output2=""
   l_pmask="0022"
   l_maxperm="$( printf '%o' $(( 0777 & ~$l_pmask )) )"
   l_root_path="$(sudo -Hiu root env | grep '^PATH' | cut -d= -f2)"
   unset a_path_loc && IFS=":" read -ra a_path_loc <<< "$l_root_path"
   grep -q "::" <<< "$l_root_path" && l_output2="$l_output2\n - root's path
contains a empty directory (::)"
   grep -Pq ":\h*$" <<< "$l_root_path" && l_output2="$l_output2\n - root's
path contains a trailing (:)"
   grep -Pq '(\h+|:)\.(:|\h*$)' <<< "$l_root_path" && l_output2="$l_output2\n
- root's path contains current working directory (.)"
   while read -r l_path; do
      if [ -d "$l_path" ]; then
         while read -r l_fmode l_fown; do
            [ "$l_fown" != "root" ] && l_output2="$l_output2\n - Directory:
\"$l_path\" is owned by: \"$l_fown\" should be owned by \"root\""
            [ $(( $l_fmode & $l_pmask )) -gt 0 ] && l_output2="$l_output2\n -
Directory: \"$l_path\" is mode: \"$l_fmode\" and should be mode:
\"$l_maxperm\" or more restrictive"
         done <<< "$(stat -Lc '%#a %U' "$l_path")"
      else
         l_output2="$l_output2\n - \"$l_path\" is not a directory"
      fi
   done <<< "$(printf "%s\n" "${a_path_loc[@]}")"
   if [ -z "$l_output2" ]; then
      echo -e "\n- Audit Result:\n  *** PASS ***\n - Root's path is correctly
configured\n"
   else
      echo -e "\n- Audit Result:\n  ** FAIL **\n - * Reasons for audit
failure * :\n$l_output2\n"
   fi
}
```

Remediation:

Correct or justify any:

- Locations that are not directories
- Empty directories (`::`)
- Trailing (`:`)
- Current working directory (`.`)
- Non `root` owned directories
- Directories that less restrictive than mode `0755`

References:

1. NIST SP 800-53 Rev. 5: CM-1, CM-2, CM-6, CM-7, IA-5

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1204, T1204.002	TA0006	M1022

5.4.2.6 Ensure root user umask is configured (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The user file-creation mode mask (`umask`) is used to determine the file permission for newly created directories and files. In Linux, the default permissions for any newly created directory is 0777 (`rwxrwxrwx`), and for any newly created file it is 0666 (`rw-rw-rw-`). The `umask` modifies the default Linux permissions by restricting (masking) these permissions. The `umask` is not simply subtracted, but is processed bitwise. Bits set in the `umask` are cleared in the resulting file mode.

`umask` can be set with either `Octal` or `Symbolic` values:

- `Octal` (Numeric) Value - Represented by either three or four digits. ie `umask 0027` or `umask 027`. If a four digit umask is used, the first digit is ignored. The remaining three digits effect the resulting permissions for user, group, and world/other respectively.
- `Symbolic` Value - Represented by a comma separated list for User `u`, group `g`, and world/other `o`. The permissions listed are not masked by `umask`. ie a `umask` set by `umask u=rwx,g=rx,o=` is the `Symbolic` equivalent of the `Octal` `umask 027`. This `umask` would set a newly created directory with file mode `drwxr-x---` and a newly created file with file mode `rw-r-----`.

root user Shell Configuration Files:

- `/root/.bash_profile` - Is executed to configure the root users' shell before the initial command prompt. **Is only read by login shells.**
- `/root/.bashrc` - Is executed for interactive shells. **only read by a shell that's both interactive and non-login**

`umask` is set by order of precedence. If `umask` is set in multiple locations, this order of precedence will determine the system's default `umask`.

Order of precedence:

1. `/root/.bash_profile`
2. `/root/.bashrc`
3. The system default umask

Rationale:

Setting a secure value for `umask` ensures that users make a conscious choice about their file permissions. A permissive `umask` value could result in directories or files with excessive permissions that can be read and/or written to by unauthorized users.

Audit:

Run the following to verify the root user `umask` is set to enforce a newly created directories' permissions to be `750 (drwxr-x---)`, and a newly created file's permissions be `640 (rw-r-----)`, or more restrictive:

```
# grep -Psi -- '^\h*umask\h+(([0-7][0-7][01][0-7]\b|[0-7][0-7][0-7][0-
6]\b)|([0-7][01][0-7]\b|[0-7][0-7][0-
6]\b)|(u=[rwx]{1,3},)?(((g=[rx]?[rx]?w[rx]?[rx]?\b)(,o=[rwx]{1,3})?)|((g=[wrx
]{1,3},)?o=[wrx]{1,3}\b)))' /root/.bash_profile /root/.bashrc
```

Nothing should be returned.

Remediation:

Edit `/root/.bash_profile` and `/root/.bashrc` and remove, comment out, or update any line with `umask` to be `0027` or more restrictive.

Default Value:

System default `umask`

References:

1. NIST SP 800-53 Rev. 5: AC-3, MP-2

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	14.6 Protect Information through Access Control Lists Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1083	TA0007	

5.4.2.7 Ensure system accounts do not have a valid login shell (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

There are a number of accounts provided with most distributions that are used to manage applications and are not intended to provide an interactive shell. Furthermore, a user may add special accounts that are not intended to provide an interactive shell.

Rationale:

It is important to make sure that accounts that are not being used by regular users are prevented from being used to provide an interactive shell. By default, most distributions set the password field for these accounts to an invalid string, but it is also recommended that the shell field in the password file be set to the `nologin` shell. This prevents the account from potentially being used to run any commands.

Audit:

Run the following command to verify system accounts, except for `root`, `halt`, `sync`, `shutdown` or `nfsnobody`, do not have a valid login shell:

```
#!/usr/bin/env bash

{
   l_valid_shells="^($(awk -F\/ '$NF != "nologin" {print}' /etc/shells | sed
-rn '/^\//{s,/,\\\\/,g;p}' | paste -s -d '|' - ))$"
   awk -v pat="$l_valid_shells" -F:
'($1!~/^(root|halt|sync|shutdown|nfsnobody)$/ && ($3<'"$(awk
'/^\s*UID_MIN/{print $2}' /etc/login.defs)"' || $3 == 65534) && $(NF) ~ pat)
{print "Service account: \"" $1 "\" has a valid shell: " $7}' /etc/passwd
}
```

Nothing should be returned

Remediation:

Run the following command to set the shell for any service accounts returned by the audit to `nologin`:

```
# usermod -s $(command -v nologin) <user>
```

Example script:

```
#!/usr/bin/env bash

{
   l_valid_shells="^($( awk -F\/ '$NF != "nologin" {print}' /etc/shells | sed
-rn '/^\//{s,/,\\\\/,g;p}' | paste -s -d '|' - ))$"
   awk -v pat="$l_valid_shells" -F:
'($1!~/^(root|halt|sync|shutdown|nfsnobody)$/ && ($3<'"$(awk
'/^\s*UID_MIN/{print $2}' /etc/login.defs)"' || $3 == 65534) && $(NF) ~ pat)
{system ("usermod -s '"$(command -v nologin)"' " $1)}' /etc/passwd
}
```

References:

1. NIST SP 800-53 Rev. 5: AC-2(5), AC-3, AC-11, MP-2

Additional Information:

The `root`, `sync`, `shutdown`, and `halt` users are exempted from requiring a non-login shell.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	14.6 Protect Information through Access Control Lists Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078, T1078.001, T1078.003	TA0005	M1026

5.4.2.8 Ensure accounts without a valid login shell are locked (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

There are a number of accounts provided with most distributions that are used to manage applications and are not intended to provide an interactive shell. Furthermore, a user may add special accounts that are not intended to provide an interactive shell.

Rationale:

It is important to make sure that accounts that are not being used by regular users are prevented from being used to provide an interactive shell. By default, most distributions set the password field for these accounts to an invalid string, but it is also recommended that the shell field in the password file be set to the `nologin` shell. This prevents the account from potentially being used to run any commands.

Audit:

Run the following script to verify all non-root accounts without a valid login shell are locked.

```
#!/usr/bin/env bash

{
   l_valid_shells="^($(awk -F\/ '$NF != "nologin" {print}' /etc/shells | sed
-rn '/^\//{s,/,\\\\/,g;p}' | paste -s -d '|' - ))$"
   while IFS= read -r l_user; do
      passwd -S "$l_user" | awk '$2 !~ /^L/ {print "Account: \"" $1 "\" does
not have a valid login shell and is not locked"}'
   done < <(awk -v pat="$l_valid_shells" -F: '($1 != "root" && $(NF) !~ pat)
{print $1}' /etc/passwd)
}
```

Nothing should be returned

Remediation:

Run the following command to lock any non-root accounts without a valid login shell returned by the audit:

```
# usermod -L <user>
```

Example script::

```
#!/usr/bin/env bash

{
   l_valid_shells="^($(awk -F\/ '$NF != "nologin" {print}' /etc/shells | sed
-rn '/^\//{s,/,\\\\/,g;p}' | paste -s -d '|' - ))$"
   while IFS= read -r l_user; do
      passwd -S "$l_user" | awk '$2 !~ /^L/ {system ("usermod -L " $1)}'
   done < <(awk -v pat="$l_valid_shells" -F: '($1 != "root" && $(NF) !~ pat)
{print $1}' /etc/passwd)
}
```

References:

1. NIST SP 800-53 Rev. 5: AC-2(5), AC-3, AC-11, MP-2

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	14.6 Protect Information through Access Control Lists Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078, T1078.001, T1078.003	TA0005	M1026

5.4.3 Configure user default environment

5.4.3.1 Ensure nologin is not listed in /etc/shells (Automated)

Profile Applicability:

- Level 2 - Server
- Level 2 - Workstation

Description:

`/etc/shells` is a text file which contains the full pathnames of valid login shells. This file is consulted by `chsh` and available to be queried by other programs.

Be aware that there are programs which consult this file to find out if a user is a normal user; for example, FTP daemons traditionally disallow access to users with shells not included in this file.

Rationale:

A user can use `chsh` to change their configured shell.

If a user has a shell configured that isn't in in `/etc/shells`, then the system assumes that they're somehow restricted. In the case of `chsh` it means that the user cannot change that value.

Other programs might query that list and apply similar restrictions.

By putting `nologin` in `/etc/shells`, any user that has `nologin` as its shell is considered a full, unrestricted user. This is not the expected behavior for `nologin`.

Audit:

Run the following command to verify that `nologin` is not listed in the `/etc/shells` file:

```
# grep -Ps '^\h*([^#\n\r]+)?\/nologin\b' /etc/shells
```

Nothing should be returned

Remediation:

Edit `/etc/shells` and remove any lines that include `nologin`

References:

1. shells(5)
2. NIST SP 800-53 Rev. 5: CM-1, CM-2, CM-6, CM-7, IA-5

5.4.3.2 Ensure default user shell timeout is configured (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

`TMOUT` is an environmental setting that determines the timeout of a shell in seconds.

- TMOUT=*n* - Sets the shell timeout to *n* seconds. A setting of `TMOUT=0` disables timeout.
- readonly TMOUT- Sets the TMOUT environmental variable as readonly, preventing unwanted modification during run-time.
- export TMOUT - exports the TMOUT variable

System Wide Shell Configuration Files:

- `/etc/profile` - used to set system wide environmental variables on users shells. The variables are sometimes the same ones that are in the `.bash_profile`, however this file is used to set an initial PATH or PS1 for all shell users of the system. **is only executed for interactive *login* shells, or shells executed with the --login parameter.**
- `/etc/profile.d` - `/etc/profile` will execute the scripts within `/etc/profile.d/*.sh`. It is recommended to place your configuration in a shell script within `/etc/profile.d` to set your own system wide environmental variables.
- `/etc/bashrc` - System wide version of `.bashrc`. In Fedora derived distributions, `/etc/bashrc` also invokes /etc/profile.d/*.sh if *non-login* shell, but redirects output to `/dev/null` if *non-interactive.* **Is only executed for *interactive* shells or if `BASH_ENV` is set to `/etc/bashrc`.**

Rationale:

Setting a timeout value reduces the window of opportunity for unauthorized user access to another user's shell session that has been left unattended. It also ends the inactive session and releases the resources associated with that session.

Audit:

Run the following script to verify that `TMOUT` is configured to: include a timeout of no more than `900` seconds, to be `readonly`, to be `exported`, and is not being changed to a longer timeout.

```
#!/usr/bin/env bash

{
   output1="" output2=""
   [ -f /etc/bashrc ] && BRC="/etc/bashrc"
   for f in "$BRC" /etc/profile /etc/profile.d/*.sh ; do
      grep -Pq '^\s*([^#]+\s+)?TMOUT=(900|[1-8][0-9][0-9]|[1-9][0-9]|[1-
9])\b' "$f" && grep -Pq
'^\s*([^#]+;\s*)?readonly\s+TMOUT(\s+|\s*;|\s*$|=(900|[1-8][0-9][0-9]|[1-
9][0-9]|[1-9]))\b' "$f" && grep -Pq
'^\s*([^#]+;\s*)?export\s+TMOUT(\s+|\s*;|\s*$|=(900|[1-8][0-9][0-9]|[1-9][0-
9]|[1-9]))\b' "$f" &&
   output1="$f"
   done
   grep -Pq '^\s*([^#]+\s+)?TMOUT=(9[0-9][1-9]|9[1-9][0-9]|0+|[1-9]\d{3,})\b'
/etc/profile /etc/profile.d/*.sh "$BRC" && output2=$(grep -Ps
'^\s*([^#]+\s+)?TMOUT=(9[0-9][1-9]|9[1-9][0-9]|0+|[1-9]\d{3,})\b'
/etc/profile /etc/profile.d/*.sh $BRC)
   if [ -n "$output1" ] && [ -z "$output2" ]; then
      echo -e "\nPASSED\n\nTMOUT is configured in: \"$output1\"\n"
   else
      [ -z "$output1" ] && echo -e "\nFAILED\n\nTMOUT is not configured\n"
      [ -n "$output2" ] && echo -e "\nFAILED\n\nTMOUT is incorrectly
configured in: \"$output2\"\n"
   fi
}
```

Remediation:

Review `/etc/bashrc`, `/etc/profile`, and all files ending in `*.sh` in the `/etc/profile.d/` directory and remove or edit all `TMOUT=_n_` entries to follow local site policy. `TMOUT` should not exceed 900 or be equal to `0`.
Configure `TMOUT` in **one** of the following files:

- A file in the `/etc/profile.d/` directory ending in `.sh`
- `/etc/profile`
- `/etc/bashrc`

`TMOUT` configuration examples:

- As multiple lines:

```
TMOUT=900
readonly TMOUT
export TMOUT
```

- As a single line:

```
readonly TMOUT=900 ; export TMOUT
```

Additional Information:

The audit and remediation in this recommendation apply to bash and shell. If other shells are supported on the system, it is recommended that their configuration files also are checked. Other methods of setting a timeout exist for other shells not covered here.

Ensure that the timeout conforms to your local policy.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.3 Configure Automatic Session Locking on Enterprise Assets Configure automatic session locking on enterprise assets after a defined period of inactivity. For general purpose operating systems, the period must not exceed 15 minutes. For mobile end-user devices, the period must not exceed 2 minutes.	●	●	●
v7	16.11 Lock Workstation Sessions After Inactivity Automatically lock workstation sessions after a standard period of inactivity.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078	TA0005	M1026

5.4.3.3 Ensure default user umask is configured (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The user file-creation mode mask (`umask`) is used to determine the file permission for newly created directories and files. In Linux, the default permissions for any newly created directory is 0777 (`rwxrwxrwx`), and for any newly created file it is 0666 (`rw-rw-rw-`). The `umask` modifies the default Linux permissions by restricting (masking) these permissions. The `umask` is not simply subtracted, but is processed bitwise. Bits set in the `umask` are cleared in the resulting file mode.

`umask` can be set with either `Octal` or `Symbolic` values:

- `Octal` (Numeric) Value - Represented by either three or four digits. ie `umask 0027` or `umask 027`. If a four digit umask is used, the first digit is ignored. The remaining three digits effect the resulting permissions for user, group, and world/other respectively.
- `Symbolic` Value - Represented by a comma separated list for User `u`, group `g`, and world/other `o`. The permissions listed are not masked by `umask`. ie a `umask` set by `umask u=rwx,g=rx,o=` is the `Symbolic` equivalent of the `Octal umask 027`. This `umask` would set a newly created directory with file mode `drwxr-x---` and a newly created file with file mode `rw-r-----`.

The default `umask` can be set to use the `pam_umask` module or in a `System Wide Shell Configuration File`. The user creating the directories or files has the discretion of changing the permissions via the chmod command, or choosing a different default `umask` by adding the `umask` command into a `User Shell Configuration File`, (`.bash_profile` or `.bashrc`), in their home directory.

Setting the default umask:

- pam_umask module:
  - will set the umask according to the system default in `/etc/login.defs` and user settings, solving the problem of different `umask` settings with different shells, display managers, remote sessions etc.
  - `umask=<mask>` value in the `/etc/login.defs` file is interpreted as Octal
  - Setting `USERGROUPS_ENAB` to yes in `/etc/login.defs` (default):
    - will enable setting of the `umask` group bits to be the same as owner bits. (examples: 022 -> 002, 077 -> 007) for non-root users, if the `uid` is the same as `gid`, and `username` is the same as the `<primary group name>`
    - userdel will remove the user's group if it contains no more members, and useradd will create by default a group with the name of the user
- `System Wide Shell Configuration File`:
  - `/etc/profile` - used to set system wide environmental variables on users shells. The variables are sometimes the same ones that are in the `.bash_profile`, however this file is used to set an initial PATH or PS1 for all shell users of the system. **is only executed for interactive *login* shells, or shells executed with the --login parameter.**
  - `/etc/profile.d` - `/etc/profile` will execute the scripts within `/etc/profile.d/*.sh`. It is recommended to place your configuration in a shell script within `/etc/profile.d` to set your own system wide environmental variables.
  - `/etc/bashrc` - System wide version of `.bashrc`. In Fedora derived distributions, `etc/bashrc` also invokes /etc/profile.d/*.sh if *non-login* shell, but redirects output to `/dev/null` if *non-interactive.* **Is only executed for *interactive* shells or if `BASH_ENV` is set to `/etc/bashrc`.**

User Shell Configuration Files:

- `~/.bash_profile` - Is executed to configure your shell before the initial command prompt. **Is only read by login shells.**
- `~/.bashrc` - Is executed for interactive shells. **only read by a shell that's both interactive and non-login**

`umask` is set by order of precedence. If `umask` is set in multiple locations, this order of precedence will determine the system's default `umask`.

Order of precedence:

1. A file in `/etc/profile.d/` ending in `.sh` - This will override any other system-wide `umask` setting
2. In the file `/etc/profile`
3. On the `pam_umask.so` module in `/etc/pam.d/postlogin`
4. In the file `/etc/login.defs`
5. In the file `/etc/default/login`

Rationale:

Setting a secure default value for `umask` ensures that users make a conscious choice about their file permissions. A permissive `umask` value could result in directories or files with excessive permissions that can be read and/or written to by unauthorized users.

Audit:

Run the following to verify the default user `umask` is set to `027`(octal) or `u=rwx,g=rx,o=` (Symbolic) to enforce newly created directories' permissions to be `750 (drwxr-x---)`, and newly created file's permissions be `640 (rw-r-----)`, or more restrictive:

```
#!/usr/bin/env bash

{
    l_output="" l_output2=""
    file_umask_chk()
    {
       if grep -Psiq -- '^\h*umask\h+(0?[0-7][2-
7]7|u(=[rwx]{0,3}),g=([rx]{0,2}),o=)(\h*#.*)?$' "$l_file"; then
          l_output="$l_output\n - umask is set correctly in \"$l_file\""
       elif grep -Psiq -- '^\h*umask\h+(([0-7][0-7][01][0-7]\b|[0-7][0-7][0-
7][0-6]\b)|([0-7][01][0-7]\b|[0-7][0-7][0-
6]\b)|(u=[rwx]{1,3},)?(((g=[rx]?[rx]?w[rx]?[rx]?\b)(,o=[rwx]{1,3})?)|((g=[wrx
]{1,3},)?o=[wrx]{1,3}\b)))' "$l_file"; then
          l_output2="$l_output2\n - umask is incorrectly set in \"$l_file\""
       fi
    }
    while IFS= read -r -d $'\0' l_file; do
       file_umask_chk
    done < <(find /etc/profile.d/ -type f -name '*.sh' -print0)
    [ -z "$l_output" ] && l_file="/etc/profile" && file_umask_chk
    [ -z "$l_output" ] && l_file="/etc/bashrc" && file_umask_chk
    [ -z "$l_output" ] && l_file="/etc/bash.bashrc" && file_umask_chk
    [ -z "$l_output" ] && l_file="/etc/pam.d/postlogin"
    if [ -z "$l_output" ]; then
       if grep -Psiq --
'^\h*session\h+[^#\n\r]+\h+pam_umask\.so\h+([^#\n\r]+\h+)?umask=(0?[0-7][2-
7]7)\b' "$l_file"; then
          l_output1="$l_output1\n - umask is set correctly in \"$l_file\""
       elif grep -Psiq
'^\h*session\h+[^#\n\r]+\h+pam_umask\.so\h+([^#\n\r]+\h+)?umask=(([0-7][0-
7][01][0-7]\b|[0-7][0-7][0-7][0-6]\b)|([0-7][01][0-7]\b))' "$l_file"; then
          l_output2="$l_output2\n - umask is incorrectly set in \"$l_file\""
       fi
    fi
    [ -z "$l_output" ] && l_file="/etc/login.defs" && file_umask_chk
    [ -z "$l_output" ] && l_file="/etc/default/login" && file_umask_chk
    [[ -z "$l_output" && -z "$l_output2" ]] && l_output2="$l_output2\n -
umask is not set"
    if [ -z "$l_output2" ]; then
       echo -e "\n- Audit Result:\n  ** PASS **\n - * Correctly configured *
:\n$l_output\n"
    else
       echo -e "\n- Audit Result:\n  ** FAIL **\n - * Reasons for audit
failure * :\n$l_output2"
       [ -n "$l_output" ] && echo -e "\n- * Correctly configured *
:\n$l_output\n"
    fi
}
```

Remediation:

Run the following script and perform the instructions in the output to set the default umask to `027` or more restrictive:

```
#!/usr/bin/env bash

{
   l_output="" l_output2="" l_out=""
   file_umask_chk()
   {
      if grep -Psiq -- '^\h*umask\h+(0?[0-7][2-
7]7|u(=[rwx]{0,3}),g=([rx]{0,2}),o=)(\h*#.*)?$' "$l_file"; then
         l_out="$l_out\n - umask is set correctly in \"$l_file\""
      elif grep -Psiq -- '^\h*umask\h+(([0-7][0-7][01][0-7]\b|[0-7][0-7][0-
7][0-6]\b)|([0-7][01][0-7]\b|[0-7][0-7][0-
6]\b)|(u=[rwx]{1,3},)?(((g=[rx]?[rx]?w[rx]?[rx]?\b)(,o=[rwx]{1,3})?)|((g=[wrx
]{1,3},)?o=[wrx]{1,3}\b)))' "$l_file"; then
         l_output2="$l_output2\n   - \"$l_file\""
      fi
   }
   while IFS= read -r -d $'\0' l_file; do
      file_umask_chk
   done < <(find /etc/profile.d/ -type f -name '*.sh' -print0)
   [ -n "$l_out" ] && l_output="$l_out"
   l_file="/etc/profile" && file_umask_chk
   l_file="/etc/bashrc" && file_umask_chk
   l_file="/etc/bash.bashrc" && file_umask_chk
   l_file="/etc/pam.d/postlogin"
   if grep -Psiq
'^\h*session\h+[^#\n\r]+\h+pam_umask\.so\h+([^#\n\r]+\h+)?umask=(([0-7][0-
7][01][0-7]\b|[0-7][0-7][0-7][0-6]\b)|([0-7][01][0-7]\b))' "$l_file"; then
      l_output2="$l_output2\n   - \"$l_file\""
   fi
   l_file="/etc/login.defs" && file_umask_chk
   l_file="/etc/default/login" && file_umask_chk
   if [ -z "$l_output2" ]; then
      echo -e " - No files contain a UMASK that is not restrictive enough\n
No UMASK updates required to existing files"
   else
      echo -e "\n - UMASK is not restrictive enough in the following
file(s):$l_output2\n\n- Remediation Procedure:\n - Update these files and
comment out the UMASK line\n   or update umask to be \"0027\" or more
restrictive"
   fi
   if [ -n "$l_output" ]; then
      echo -e "$l_output"
   else
      echo -e " - Configure UMASK in a file in the \"/etc/profile.d/\"
directory ending in \".sh\"\n\n   Example Command (Hash to represent being
run at a root prompt):\n\n# printf '%s\\\n' \"umask 027\" >
/etc/profile.d/50-systemwide_umask.sh\n"
   fi
}
```

Notes:

- This method only applies to bash and shell. If other shells are supported on the system, it is recommended that their configuration files also are checked
- If the `pam_umask.so` module is going to be used to set `umask`, ensure that it's not being overridden by another setting. Refer to the PAM_UMASK(8) man page for more information

Default Value:

UMASK 022

References:

1. NIST SP 800-53 Rev. 5: AC-3, MP-2

Additional Information:

- Other methods of setting a default user umask exist
- If other methods are in use in your environment they should be audited
- The default user umask can be overridden with a user specific umask
- The user creating the directories or files has the discretion of changing the permissions:
    - Using the chmod command
    - Setting a different default umask by adding the umask command into a User Shell Configuration File, (.bashrc), in their home directory
    - Manually changing the umask for the duration of a login session by running the umask command

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	14.6 Protect Information through Access Control Lists Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1083	TA0007	