

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 similar 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_authok
```

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_authok
```

- **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_authok
```

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_authok')
    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

5.3.3 Configure PAM Arguments

Pluggable Authentication Modules (PAM) uses arguments to pass information to a pluggable module during authentication for a particular module type. These arguments allow the PAM configuration files for particular programs to use a common PAM module but in different ways.

Invalid arguments are ignored and do not otherwise affect the success or failure of the PAM module. When an invalid argument is passed, an error is usually written to `/var/log/messages` file. However, since the reporting method is controlled by the PAM module, the module must be written correctly to log the error to this file.

Note: If custom PAM files are being used, for this section's remediation, the corresponding files in `/etc/pam.d/` would need to be edited directly, and the `pam-auth-update --enable <EDITED_PROFILE_NAME>` command skipped

5.3.3.1 Configure pam_faillock module

`pam_faillock.so` provides a way to configure the default settings for locking the user after multiple failed authentication attempts.

Options:

- `<dir=/path/to/tally-directory>` - The directory where the user files with the failure records are kept. The default is `/var/run/faillock`. Note: These files will disappear after reboot on systems configured with directory `/var/run/faillock` mounted on virtual memory.
- `audit` - Will log the user name into the system log if the user is not found.
- `silent` - Don't print informative messages to the user. Please note that when this option is not used there will be difference in the authentication behavior for users which exist on the system and non-existing users.
- `no_log_info` - Don't log informative messages via syslog(3).
- `local_users_only` - Only track failed user authentications attempts for local users in `/etc/passwd` and ignore centralized (AD, IdM, LDAP, etc.) users. The `faillock(8)` command will also no longer track user failed authentication attempts. Enabling this option will prevent a double-lockout scenario where a user is locked out locally and in the centralized mechanism.
- `nodelay` - Don't enforce a delay after authentication failures.
- `deny=<n>` - Deny access if the number of consecutive authentication failures for this user during the recent interval exceeds `n`. The default is 3.
- `fail_interval=n` - The length of the interval during which the consecutive authentication failures must happen for the user account lock out is `n` seconds. The default is 900 (15 minutes).
- `unlock_time=n` - The access will be re-enabled after `n` seconds after the lock out. The value 0 has the same meaning as value never - the access will not be re-enabled without resetting the faillock entries by the `faillock(8)` command. The default is 600 (10 minutes). Note that the default directory that `pam_faillock` uses is usually cleared on system boot so the access will be also re-enabled after system reboot. If that is undesirable a different tally directory must be set with the `dir` option. Also note that it is usually undesirable to permanently lock out users as they can become easily a target of denial of service attack unless the usernames are random and kept secret to potential attackers.
- `even_deny_root` - Root account can become locked as well as regular accounts.
- `root_unlock_time=n` - This option implies `even_deny_root` option. Allow access after `n` seconds to root account after the account is locked. In case the option is not specified the value is the same as of the `unlock_time` option.
- `admin_group=name` - If a group name is specified with this option, members of the group will be handled by this module the same as the root account (the options `even_deny_root` and `root_unlock_time` will apply to them. By default the option is not set.

5.3.3.1.1 Ensure password failed attempts lockout is configured (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The **deny=<n>** option will deny access if the number of consecutive authentication failures for this user during the recent interval exceeds .

Rationale:

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

Audit:

Run the following command to verify that Number of failed logon attempts before the account is locked is no greater than **5** and meets local site policy:

```
# grep -Pi -- '^\\h*deny\\h*=\\h*[1-5]\\b' /etc/security/faillock.conf
deny = 5
```

Run the following command to verify that the **deny** argument has not been set, or **5** or less and meets local site policy:

```
# grep -Pi -- '^\\h*auth\\h+(requisite|required|sufficient)\\h+pam_faillock\\.so\\h+([\\^#\\n\\r]+\\h+)?deny\\h*=\\h*(0|[6-9]|[1-9][0-9]+)\\b' /etc/pam.d/common-auth
```

Nothing should be returned

Remediation:

Create or edit the following line in **/etc/security/faillock.conf** setting the **deny** option to **5** or less:

```
deny = 5
```

Run the following command:

```
# grep -Pl -- '\\bpam_faillock\\.so\\h+([\\^#\\n\\r]+\\h+)?deny\\b' /usr/share/pam-configs/*
```

Edit any returned files and remove the **deny=<N>** arguments from the **pam_faillock.so** line(s):






Default Value:

deny = 3

Additional Information:

If a user has been locked out because they have reached the maximum consecutive failure count defined by **deny=** in the **pam_faillock.so** module, the user can be unlocked by issuing the command **faillock --user <USERNAME> --reset**. This command sets the failed count to 0, effectively unlocking the user.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.2 <u>Establish an Access Revoking Process</u> 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 <u>Establish Process for Revoking Access</u> 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

5.3.3.1.2 Ensure password unlock time is configured (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

`unlock_time=<n>` - The access will be re-enabled after seconds after the lock out. The value `0` has the same meaning as value never - the access will not be re-enabled without resetting the faillock entries by the `faillock(8)` command.

Note:

- The default directory that `pam_faillock` uses is usually cleared on system boot so the access will be also re-enabled after system reboot. If that is undesirable a different tally directory must be set with the `dir` option.
- It is usually undesirable to permanently lock out users as they can become easily a target of denial of service attack unless the usernames are random and kept secret to potential attackers.
- The maximum configurable value for `unlock_time` is `604800`

Rationale:

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

Impact:

Use of `unlock_time=0` may allow an attacker to cause denial of service to legitimate users. This will also require a systems administrator with elevated privileges to unlock the account.

Audit:

Run the following command to verify that the time in seconds before the account is unlocked is either 0 (never) or 900 (15 minutes) or more and meets local site policy:

```
# grep -Pi -- '^h*unlock_timeh*=\h*(0|9[0-9][0-9]|[1-9][0-9]{3,})\b'
/etc/security/faillock.conf

unlock_time = 900
```

Run the following command to verify that the `unlock_time` argument has not been set, or is either 0 (never) or 900 (15 minutes) or more and meets local site policy:

```
# grep -Pi --
'^h*authh+(requisite|required|sufficient)\h+pam_faillock\.so\h+([\#\n\r]+\h
+)?unlock_timeh*=\h*([1-9]|[1-9][0-9]|[1-8][0-9][0-9])\b' /etc/pam.d/common-
auth
```

Nothing should be returned

Remediation:

Set password unlock time to conform to site policy. `unlock_time` should be 0 (never), or 900 seconds or greater.

Edit `/etc/security/faillock.conf` and update or add the following line:

```
unlock_time = 900
```

Run the following command: remove the `unlock_time` argument from the `pam_faillock.so` module in the PAM files:

```
# grep -Pl -- '\bpam_faillock\.so\h+([\#\n\r]+\h+)?unlock_time\b'
/usr/share/pam-configs/*
```

Edit any returned files and remove the `unlock_time=<N>` argument from the `pam_faillock.so` line(s):






Default Value:

```
unlock_time = 600
```

Additional Information:

If a user has been locked out because they have reached the maximum consecutive failure count defined by `deny=` in the `pam_faillock.so` module, the user can be unlocked by issuing the command `faillock --user <USERNAME> --reset`. This command sets the failed count to 0, effectively unlocking the user.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.2 <u>Establish an Access Revoking Process</u> 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 <u>Establish Process for Revoking Access</u> 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

5.3.3.1.3 Ensure password failed attempts lockout includes root account (Automated)

Profile Applicability:

- Level 2 - Server
- Level 2 - Workstation

Description:

`even_deny_root` - Root account can become locked as well as regular accounts

`root_unlock_time=n` - This option implies `even_deny_root` option. Allow access after n seconds to root account after the account is locked. In case the option is not specified the value is the same as of the `unlock_time` option.

Rationale:

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

Impact:

Use of `unlock_time=0` or `root_unlock_time=0` may allow an attacker to cause denial of service to legitimate users.

Audit:

Run the following command to verify that **even_deny_root** and/or **root_unlock_time** is enabled:

```
# grep -Pi -- '^\\h*(even_deny_root|root_unlock_time\\h*=\\h*\\d+)\\b' /etc/security/faillock.conf
```

Example output:

```
even_deny_root

--AND/OR--

root_unlock_time = 60
```

Run the following command to verify that - **IF** - **root_unlock_time** is set, it is set to **60** (One minute) or more:

```
# grep -Pi -- '^\\h*root_unlock_time\\h*=\\h*([1-9]|[1-5][0-9])\\b' /etc/security/faillock.conf
```

Nothing should be returned

Run the following command to check the **pam_faillock.so** module for the **root_unlock_time** argument. Verify -**IF**- **root_unlock_time** is set, it is set to **60** (One minute) or more:

```
# grep -Pi -- '^\\h*auth\\h+([\\^#\\n\\r]+\\h+)pam_faillock\\.so\\h+([\\^#\\n\\r]+\\h+)?root_unlock_time\\h*=\\h*([1-9]|[1-5][0-9])\\b' /etc/pam.d/common-auth
```

Nothing should be returned

Remediation:

Edit **/etc/security/faillock.conf**:

- Remove or update any line containing **root_unlock_time**, - **OR** - set it to a value of **60** or more
- Update or add the following line:

```
even_deny_root
```

Run the following command:

```
# grep -Pl -- '\\bpam_faillock\\.so\\h+([\\^#\\n\\r]+\\h+)?(even_deny_root|root_unlock_time)' /usr/share/pam-configs/*
```

Edit any returned files and remove the **even_deny_root** and **root_unlock_time** arguments from the **pam_faillock.so** line(s):






Default Value:

disabled

Additional Information:

If a user has been locked out because they have reached the maximum consecutive failure count defined by `deny=` in the `pam_faillock.so` module, the user can be unlocked by issuing the command `faillock --user <USERNAME> --reset`. This command sets the failed count to 0, effectively unlocking the user.

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

5.3.3.2 Configure pam_pwquality module

The `pam_pwquality.so` module checks the strength of passwords. It performs checks such as making sure a password is not a dictionary word, it is a certain length, contains a mix of characters (e.g. alphabet, numeric, other) and more.

These checks are configurable by either:

- use of the module arguments
- modifying the `/etc/security/pwquality.conf` configuration file
- creating a `.conf` file in the `/etc/security/pwquality.conf.d/` directory.

Note: The module arguments override the settings in the `/etc/security/pwquality.conf` configuration file. Settings in the `/etc/security/pwquality.conf` configuration file override settings in a `.conf` file in the `/etc/security/pwquality.conf.d/` directory.

The possible options in the file are:

- `difok` - Number of characters in the new password that must not be present in the old password. (default 1). The special value of 0 disables all checks of similarity of the new password with the old password except the new password being exactly the same as the old one.
- `minlen` - Minimum acceptable size for the new password (plus one if credits are not disabled which is the default). (See `pam_pwquality(8)`.) Cannot be set to lower value than 6. (default 8)
- `dcredit` - The maximum credit for having digits in the new password. If less than 0 it is the minimum number of digits in the new password. (default 0)
- `uccredit` - The maximum credit for having uppercase characters in the new password. If less than 0 it is the minimum number of uppercase characters in the new password. (default 0)
- `lcredit` - The maximum credit for having lowercase characters in the new password. If less than 0 it is the minimum number of lowercase characters in the new password. (default 0)
- `ocredit` - The maximum credit for having other characters in the new password. If less than 0 it is the minimum number of other characters in the new password. (default 0)
- `minclass` - The minimum number of required classes of characters for the new password (digits, uppercase, lowercase, others). (default 0)
- `maxrepeat` - The maximum number of allowed same consecutive characters in the new password. The check is disabled if the value is 0. (default 0)
- `maxsequence` - The maximum length of monotonic character sequences in the new password. Examples of such sequence are '12345' or 'fedcb'. Note that most such passwords will not pass the simplicity check unless the sequence is only a minor part of the password. The check is disabled if the value is 0. (default 0)

- **maxclassrepeat** - The maximum number of allowed consecutive characters of the same class in the new password. The check is disabled if the value is 0. (default 0)
- **gecoscheck** - If nonzero, check whether the words longer than 3 characters from the GECOS field of the user's passwd(5) entry are contained in the new password. The check is disabled if the value is 0. (default 0)
- **dictcheck** - If nonzero, check whether the password (with possible modifications) matches a word in a dictionary. Currently the dictionary check is performed using the cracklib library. (default 1)
- **usercheck=<N>** - If nonzero, check whether the password (with possible modifications) contains the user name in some form. It is not performed for user names shorter than 3 characters. (default 1)
- **usersubstr=<N>** - If greater than 3 (due to the minimum length in usercheck), check whether the password contains a substring of at least N length in some form. (default 0)
- **enforcing=<N>** - If nonzero, reject the password if it fails the checks, otherwise only print the warning. This setting applies only to the pam_pwquality module and possibly other applications that explicitly change their behavior based on it. It does not affect pwmake(1) and pwscore(1). (default 1)
- **badwords** - Space separated list of words that must not be contained in the password. These are additional words to the cracklib dictionary check. This setting can be also used by applications to emulate the gecos check for user accounts that are not created yet.
- **dictpath** - Path to the cracklib dictionaries. Default is to use the cracklib default.
- **retry=<N>** - Prompt user at most N times before returning with error. The default is 1.
- **enforce_for_root** - The module will return error on failed check even if the user changing the password is root. This option is off by default which means that just the message about the failed check is printed but root can change the password anyway. Note that root is not asked for an old password so the checks that compare the old and new password are not performed.
- **local_users_only** - The module will not test the password quality for users that are not present in the /etc/passwd file. The module still asks for the password so the following modules in the stack can use the use_authok option. This option is off by default.

5.3.3.2.1 Ensure password number of changed characters is configured (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The `pwquality difok` option sets the number of characters in a password that must not be present in the old password.

Rationale:

Use of a complex password helps to increase the time and resources required to compromise the password. Password complexity, or strength, is a measure of the effectiveness of a password in resisting attempts at guessing and brute-force attacks.

Password complexity is one factor of several that determines how long it takes to crack a password. The more complex the password, the greater the number of possible combinations that need to be tested before the password is compromised.

Audit:

Run the following command to verify that the **difok** option is set to **2** or more and follows local site policy:

```
# grep -Psi -- '^\\h*difok\\h*=\\h*([2-9]|[1-9][0-9]+)\\b'
/etc/security/pwquality.conf /etc/security/pwquality.conf.d/*.conf
```

Example output:

```
/etc/security/pwquality.conf.d/50-pwdifok.conf:difok = 2
```

Verify returned value(s) are **2** or more and meet local site policy

Run the following command to verify that **difok** is not set, is **2** or more, and conforms to local site policy:

```
# grep -Psi --
'^\\h*password\\h+(requisite|required|sufficient)\\h+pam_pwquality\\.so\\h+([^\n\r]+\\h+)?difok\\h*=\\h*([0-1])\\b' /etc/pam.d/common-password
```

Nothing should be returned

Note:

- settings should be configured in only **one** location for clarity
- Settings observe an order of precedence:
 - module arguments override the settings in the **/etc/security/pwquality.conf** configuration file
 - settings in the **/etc/security/pwquality.conf** configuration file override settings in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory
 - settings in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory are read in canonical order, with last read file containing the setting taking precedence
- It is recommended that settings be configured in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory for clarity, convenience, and durability.

Remediation:

Create or modify a file ending in **.conf** in the **/etc/security/pwquality.conf.d/** directory or the file **/etc/security/pwquality.conf** and add or modify the following line to set **difok** to **2** or more. Ensure setting conforms to local site policy:

Example:

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

{
    sed -ri 's/^\s*difok\s*=/# &/' /etc/security/pwquality.conf
    [ ! -d /etc/security/pwquality.conf.d/ ] && mkdir
    /etc/security/pwquality.conf.d/
    printf '\n%s' "difok = 2" > /etc/security/pwquality.conf.d/50-pwdifok.conf
}
```

Run the following command:

```
# grep -Pl -- '\bpam_pwquality\.so\h+([\^#\n\r]+\h+)?difok\b' /usr/share/pam-
configs/*
```

Edit any returned files and remove the **difok** argument from the **pam_pwquality.so** line(s):






Default Value:

difok = 1

References:

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

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

5.3.3.2.2 Ensure minimum password length is configured (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The minimum password length setting determines the lowest number of characters that make up a password for a user account. There are many different theories about how to determine the best password length for an organization, but perhaps "passphrase" is a better term than "password".

The **minlen** option sets the minimum acceptable size for the new password (plus one if credits are not disabled which is the default). Cannot be set to lower value than 6.

Rationale:

Strong passwords help protect systems from password attacks. Types of password attacks include dictionary attacks, which attempt to use common words and phrases, and brute force attacks, which try every possible combination of characters. Also attackers may try to obtain the account database so they can use tools to discover the accounts and passwords.

Impact:

In general, it is true that longer passwords are better (harder to crack), but it is also true that forced password length requirements can cause user behavior that is predictable and undesirable. For example, requiring users to have a minimum 16-character password may cause them to choose repeating patterns like fourfourfourfour or passwordpassword that meet the requirement but aren't hard to guess. Additionally, length requirements increase the chances that users will adopt other insecure practices, like writing them down, re-using them or storing them unencrypted in their documents.

Having a reasonable minimum length with no maximum character limit increases the resulting average password length used (and therefore the strength).⁶

Audit:

Run the following command to verify that password length is **14** or more characters, and conforms to local site policy:

```
# grep -Psi -- '^h*minlen\h*=\h*(1[4-9]|[2-9][0-9]|[1-9][0-9]{2,})\b'
/etc/security/pwquality.conf /etc/security/pwquality.conf.d/*.conf
```

Example output:

```
/etc/security/pwquality.conf.d/50-pwlength.conf:minlen = 14
```

Verify returned value(s) are no less than **14** characters and meet local site policy

Run the following command to verify that **minlen** is not set, or is **14** or more characters, and conforms to local site policy:

```
# grep -Psi --
'^h*password\h+(requisite|required|sufficient)\h+pam_pwquality\.so\h+([\^#\n\
r]+\h+)?minlen\h*=\h*([0-9]|1[0-3])\b' /etc/pam.d/system-auth
/etc/pam.d/common-password
```

Nothing should be returned

Note:

- settings should be configured in only **one** location for clarity
- Settings observe an order of precedence:
 - module arguments override the settings in the **/etc/security/pwquality.conf** configuration file
 - settings in the **/etc/security/pwquality.conf** configuration file override settings in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory
 - settings in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory are read in canonical order, with last read file containing the setting taking precedence
- It is recommended that settings be configured in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory for clarity, convenience, and durability.

Remediation:

Create or modify a file ending in **.conf** in the **/etc/security/pwquality.conf.d/** directory or the file **/etc/security/pwquality.conf** and add or modify the following line to set password length of **14** or more characters. Ensure that password length conforms to local site policy:

Example:

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

{
    sed -ri 's/^\s*minlen\s*=/# &/' /etc/security/pwquality.conf
    [ ! -d /etc/security/pwquality.conf.d/ ] && mkdir
    /etc/security/pwquality.conf.d/
    printf '\n%s' "minlen = 14" > /etc/security/pwquality.conf.d/50-
    pwlength.conf
}
```

Run the following command:

```
# grep -Pl -- '\bpam_pwquality\.so\h+([\#\n\r]+\h+)?minlen\b' /usr/share/pam-
configs/*
```

Edit any returned files and remove the **minlen** argument from the **pam_pwquality.so** line(s):






Default Value:

minlen = 8

References:

1. pam_pwquality(8)
2. CIS Password Policy Guide
3. NIST SP 800-53 Rev. 5: IA-5(1)

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

5.3.3.2.3 Ensure password complexity is configured (Manual)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

Password complexity can be set through:

- **minclass** - The minimum number of classes of characters required in a new password. (digits, uppercase, lowercase, others). e.g. **minclass = 4** requires digits, uppercase, lower case, and special characters.
- **dcredit** - The maximum credit for having digits in the new password. If less than 0 it is the minimum number of digits in the new password. e.g. **dcredit = -1** requires at least one digit
- **ucredit** - The maximum credit for having uppercase characters in the new password. If less than 0 it is the minimum number of uppercase characters in the new password. e.g. **ucredit = -1** requires at least one uppercase character
- **ocredit** - The maximum credit for having other characters in the new password. If less than 0 it is the minimum number of other characters in the new password. e.g. **ocredit = -1** requires at least one special character
- **lcredit** - The maximum credit for having lowercase characters in the new password. If less than 0 it is the minimum number of lowercase characters in the new password. e.g. **lcredit = -1** requires at least one lowercase character

Rationale:

Strong passwords protect systems from being hacked through brute force methods.

Requiring at least one non-alphabetic character increases the search space beyond pure dictionary words, which makes the resulting password harder to crack.

Forcing users to choose an excessively complex password, e.g. some combination of upper-case, lower-case, numbers, and special characters, has a negative impact. It places an extra burden on users and many will use predictable patterns (for example, a capital letter in the first position, followed by lowercase letters, then one or two numbers, and a “special character” at the end). Attackers know this, so dictionary attacks will often contain these common patterns and use the most common substitutions like, \$ for s, @ for a, 1 for l, 0 for o.

Impact:

Passwords that are too complex in nature make it harder for users to remember, leading to bad practices. In addition, composition requirements provide no defense against common attack types such as social engineering or insecure storage of passwords

Audit:

Run the following command to verify:

- **dcredit**, **ucredit**, **lcredit**, and **ocredit** are not set to a value greater than 0
- Complexity conforms to local site policy:

```
# grep -Psi -- '^\\h*(minclass|[dulo]credit)\\b' /etc/security/pwquality.conf /etc/security/pwquality.conf.d/*.conf
```

Example output:

```
/etc/security/pwquality.conf.d/50-pwcomplexity.conf:minclass = 3
/etc/security/pwquality.conf.d/50-pwcomplexity.conf:ucredit = -2
/etc/security/pwquality.conf.d/50-pwcomplexity.conf:lcredit = -2
/etc/security/pwquality.conf.d/50-pwcomplexity.conf:dcredit = -1
/etc/security/pwquality.conf.d/50-pwcomplexity.conf:ocredit = 0
```

The example represents a requirement of three character classes, with passwords requiring two upper case, two lower case, and one numeric character.

Run the following command to verify that module arguments in the configuration file(s) are not being overridden by arguments in **/etc/pam.d/common-password**:

```
# grep -Psi -- '^\\h*password\\h+(requisite|required|sufficient)\\h+pam_pwquality\\.so\\h+([^#\\n\\r]+\\h+)?(minclass=\\d*|[dulo]credit=-?\\d*)\\b' /etc/pam.d/common-password
```

Nothing should be returned

Note:

- settings should be configured in only **one** location for clarity
- Settings observe an order of precedence:
 - module arguments override the settings in the **/etc/security/pwquality.conf** configuration file
 - settings in the **/etc/security/pwquality.conf** configuration file override settings in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory
 - settings in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory are read in canonical order, with last read file containing the setting taking precedence
- It is recommended that settings be configured in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory for clarity, convenience, and durability.

Remediation:

Run the following command:

```
# grep -Pl --
'\bpam_pwquality\.so\h+([\^#\n\r]+\h+)?(minclass|[dulo]credit)\b'
/usr/share/pam-configs/*
```

Edit any returned files and remove the **minclass**, **dcredit**, **ucredit**, **lcredit**, and **ocredit** arguments from the **pam_pwquality.so** line(s)

Create or modify a file ending in **.conf** in the **/etc/security/pwquality.conf.d/** directory or the file **/etc/security/pwquality.conf** and add or modify the following line(s) to set complexity according to local site policy:

- **minclass** = **_N_**
- **dcredit** = **_N_** # Value should be either **0** or a number proceeded by a minus (-) symbol
- **ucredit** = **-1** # Value should be either **0** or a number proceeded by a minus (-) symbol
- **ocredit** = **-1** # Value should be either **0** or a number proceeded by a minus (-) symbol
- **lcredit** = **-1** # Value should be either **0** or a number proceeded by a minus (-) symbol

Example 1 - Set **minclass** = 3:

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

{
    sed -ri 's/^\s*minclass\s*=/# &/' /etc/security/pwquality.conf
    sed -ri 's/^\s*[dulo]credit\s*=/# &/' /etc/security/pwquality.conf
    [ ! -d /etc/security/pwquality.conf.d/ ] && mkdir
    /etc/security/pwquality.conf.d/
    printf '\n%s' "minclass = 3" > /etc/security/pwquality.conf.d/50-
    pwcomplexity.conf
}
```

Example 2 - set **dcredit** = -1, **ucredit** = -1, and **lcredit** = -1:

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

{
    sed -ri 's/^\s*minclass\s*=/# &/' /etc/security/pwquality.conf
    sed -ri 's/^\s*[dulo]credit\s*=/# &/' /etc/security/pwquality.conf
    [ ! -d /etc/security/pwquality.conf.d/ ] && mkdir
    /etc/security/pwquality.conf.d/
    printf '%s\n' "dcredit = -1" "ucredit = -1" "lcredit = -1" >
    /etc/security/pwquality.conf.d/50-pwcomplexity.conf
}
```


Default Value:

minclass = 0

dcredit = 0

ucredit = 0






ocredit = 0

lcredit = 0

References:

1. pam_pwquality(8)
2. PWQUALITY.CONF(5)
3. <https://www.cisecurity.org/insights/white-papers/cis-password-policy-guide>
4. NIST SP 800-53 Rev. 5: IA-5

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

5.3.3.2.4 Ensure password same consecutive characters is configured (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The `pwquality maxrepeat` option sets the maximum number of allowed same consecutive characters in a new password.

Rationale:

Use of a complex password helps to increase the time and resources required to compromise the password. Password complexity, or strength, is a measure of the effectiveness of a password in resisting attempts at guessing and brute-force attacks.

Password complexity is one factor of several that determines how long it takes to crack a password. The more complex the password, the greater the number of possible combinations that need to be tested before the password is compromised.

Audit:

Run the following command to verify that the **maxrepeat** option is set to **3** or less, not **0**, and follows local site policy:

```
# grep -Psi -- '^h*maxrepeat\h*=\h*[1-3]\b' /etc/security/pwquality.conf
/etc/security/pwquality.conf.d/*.conf
```

Example output:

```
/etc/security/pwquality.conf.d/50-pwrepeat.conf:maxrepeat = 3
```

Verify returned value(s) are **3** or less, not **0**, and meet local site policy

Run the following command to verify that **maxrepeat** is not set, is **3** or less, not **0**, and conforms to local site policy:

```
# grep -Psi --
'^h*password\h+(requisite|required|sufficient)\h+pam_pwquality\.so\h+([\^#\n\
r]+\h+)?maxrepeat\h*=\h*(0|[4-9]|[1-9][0-9]+)\b' /etc/pam.d/common-password
```

Nothing should be returned

Note:

- settings should be configured in only **one** location for clarity
- Settings observe an order of precedence:
 - module arguments override the settings in the **/etc/security/pwquality.conf** configuration file
 - settings in the **/etc/security/pwquality.conf** configuration file override settings in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory
 - settings in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory are read in canonical order, with last read file containing the setting taking precedence
- It is recommended that settings be configured in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory for clarity, convenience, and durability.

Remediation:

Create or modify a file ending in **.conf** in the **/etc/security/pwquality.conf.d/** directory or the file **/etc/security/pwquality.conf** and add or modify the following line to set **maxrepeat** to **3** or less and not **0**. Ensure setting conforms to local site policy:

Example:

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

{
    sed -ri 's/^\s*maxrepeat\s*=/# &/' /etc/security/pwquality.conf
    [ ! -d /etc/security/pwquality.conf.d/ ] && mkdir
    /etc/security/pwquality.conf.d/
    printf '\n%s' "maxrepeat = 3" > /etc/security/pwquality.conf.d/50-
    pwrepeat.conf
}
```

Run the following command:

```
# grep -Pl -- '\bpam_pwquality\.so\h+([\^#\n\r]+\h+)?maxrepeat\b'
/usr/share/pam-configs/*
```

Edit any returned files and remove the **maxrepeat** argument from the **pam_pwquality.so** line(s):






Default Value:

maxrepeat = 0

References:

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

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

5.3.3.2.5 Ensure password maximum sequential characters is configured (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The `pwquality maxsequence` option sets the maximum length of monotonic character sequences in the new password. Examples of such sequence are `12345` or `fedcb`. The check is disabled if the value is `0`.

Note: Most such passwords will not pass the simplicity check unless the sequence is only a minor part of the password.

Rationale:

Use of a complex password helps to increase the time and resources required to compromise the password. Password complexity, or strength, is a measure of the effectiveness of a password in resisting attempts at guessing and brute-force attacks.

Password complexity is one factor of several that determines how long it takes to crack a password. The more complex the password, the greater the number of possible combinations that need to be tested before the password is compromised.

Audit:

Run the following command to verify that the **maxsequence** option is set to **3** or less, not **0**, and follows local site policy:

```
# grep -Psi -- '^h*maxsequenceh*=\h*[1-3]\b' /etc/security/pwquality.conf
/etc/security/pwquality.conf.d/*.conf
```

Example output:

```
/etc/security/pwquality.conf.d/50-pwmaxsequence.conf:maxsequence = 3
```

Verify returned value(s) are **3** or less, not **0**, and meet local site policy

Run the following command to verify that **maxsequence** is not set, is **3** or less, not **0**, and conforms to local site policy:

```
# grep -Psi --
'^h*passwordh+(requisite|required|sufficient)\h+pam_pwquality\.so\h+([\^#\n\r]+\h+)?maxsequenceh*=\h*(0|[4-9]|[1-9][0-9]+)\b' /etc/pam.d/common-password
```

Nothing should be returned

Note:

- settings should be configured in only **one** location for clarity
- Settings observe an order of precedence:
 - module arguments override the settings in the **/etc/security/pwquality.conf** configuration file
 - settings in the **/etc/security/pwquality.conf** configuration file override settings in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory
 - settings in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory are read in canonical order, with last read file containing the setting taking precedence
- It is recommended that settings be configured in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory for clarity, convenience, and durability.

Remediation:

Create or modify a file ending in **.conf** in the **/etc/security/pwquality.conf.d/** directory or the file **/etc/security/pwquality.conf** and add or modify the following line to set **maxsequence** to **3** or less and not **0**. Ensure setting conforms to local site policy:

Example:

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

{
    sed -ri 's/^\s*maxsequence\s*=/# &/' /etc/security/pwquality.conf
    [ ! -d /etc/security/pwquality.conf.d/ ] && mkdir
    /etc/security/pwquality.conf.d/
    printf '\n%s' "maxsequence = 3" > /etc/security/pwquality.conf.d/50-
    pwmaxsequence.conf
}
```

Run the following command:

```
# grep -Pl -- '\bpam_pwquality\.so\h+([\^#\n\r]+\h+)?maxsequence\b'
/usr/share/pam-configs/*
```

Edit any returned files and remove the **maxsequence** argument from the **pam_pwquality.so** line(s):






Default Value:

maxsequence = 0

References:

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

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

5.3.3.2.6 Ensure password dictionary check is enabled (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The `pwquality dictcheck` option sets whether to check for the words from the `cracklib` dictionary.

Rationale:

If the operating system allows the user to select passwords based on dictionary words, this increases the chances of password compromise by increasing the opportunity for successful guesses, and brute-force attacks.

Audit:

Run the following command to verify that the **dictcheck** option is not set to **0** (disabled) in a pwquality configuration file:

```
# grep -Psi -- '^h*dictcheck\h*=\h*0\b' /etc/security/pwquality.conf
/etc/security/pwquality.conf.d/*.conf
```

Nothing should be returned

Run the following command to verify that the **dictcheck** option is not set to **0** (disabled) as a module argument in a PAM file:

```
# grep -Psi -- '^h*password\h+(requisite|required|sufficient)\h+pam_pwquality\.so\h+([\n\r]+\h+)?dictcheck\h*=\h*0\b' /etc/pam.d/common-password
```

Nothing should be returned

Note:

- Settings observe an order of precedence:
 - module arguments override the settings in the **/etc/security/pwquality.conf** configuration file
 - settings in the **/etc/security/pwquality.conf** configuration file override settings in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory
 - settings in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory are read in canonical order, with last read file containing the setting taking precedence
- It is recommended that settings be configured in a **.conf** file in the **/etc/security/pwquality.conf.d/** directory for clarity, convenience, and durability.

Remediation:

Edit any file ending in **.conf** in the **/etc/security/pwquality.conf.d/** directory and/or the file **/etc/security/pwquality.conf** and comment out or remove any instance of **dictcheck = 0**:

Example:

```
# sed -ri 's/^s*dictcheck\s*=/# &/' /etc/security/pwquality.conf
/etc/security/pwquality.conf.d/*.conf
```

Run the following command:

```
# grep -Pl -- '\bpam_pwquality\.so\h+([\n\r]+\h+)?dictcheck\b'
/usr/share/pam-configs/*
```

Edit any returned files and remove the **dictcheck** argument from the **pam_pwquality.so** line(s)






Default Value:

dictcheck = 1

References:

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

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

5.3.3.2.7 Ensure password quality checking is enforced (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The **pam_pwquality** module can be configured to either reject a password if it fails the checks, or only print a warning.

This is configured by setting the **enforcing=<N>** argument. If nonzero, a password will be rejected if it fails the checks, otherwise only a warning message will be provided.

This setting applies only to the **pam_pwquality** module and possibly other applications that explicitly change their behavior based on it. It does not affect **pwmake(1)** and **pwscore(1)**.

Rationale:

Strong passwords help protect systems from password attacks. Types of password attacks include dictionary attacks, which attempt to use common words and phrases, and brute force attacks, which try every possible combination of characters. Also attackers may try to obtain the account database so they can use tools to discover the accounts and passwords.

Audit:

Run the following command to verify that **enforcing=0** has not been set in a **pwquality** configuration file:

```
# grep -PHsi -- '^h*enforcing\h*=\h*0\b' /etc/security/pwquality.conf
/etc/security/pwquality.conf.d/*.conf
```

Nothing should be returned

Run the following command to verify that the **enforcing=0** argument has not been set on the **pam_pwquality** module:

```
# grep -PHsi --
'^h*password\h+([^\n\r]+\h+pam_pwquality\.so\h+([^\n\r]+\h+)?enforcing=0\b'
/etc/pam.d/common-password
```

Nothing should be returned

Remediation:

Run the following command:

```
# grep -Pl -- '\bpam_pwquality\.so\h+([\#\n\r]+\h+)?enforcing=0\b' /usr/share/pam-configs/*
```

Edit any returned files and remove the **enforcing=0** argument from the **pam_pwquality.so** line(s)

Edit **/etc/security/pwquality.conf** and all files ending in **.conf** in the **/etc/security/pwquality.conf.d/** directory and remove or comment out any line containing the **enforcing = 0** argument:

Example:

```
# sed -ri 's/^\s*enforcing\s*=\s*0/# &/' /etc/security/pwquality.conf /etc/security/pwquality.conf.d/*.conf
```






Default Value:

enforcing=1

References:

1. pam_pwquality(8)
2. PWQUALITY.CONF(5)
3. NIST SP 800-53 Rev. 5: IA-5

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.			

5.3.3.2.8 Ensure password quality is enforced for the root user (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

If the `pwquality enforce_for_root` option is enabled, the module will return error on failed check even if the user changing the password is root.

This option is off by default which means that just the message about the failed check is printed but root can change the password anyway.

Note: The root is not asked for an old password so the checks that compare the old and new password are not performed.

Rationale:

Use of a complex password helps to increase the time and resources required to compromise the password. Password complexity, or strength, is a measure of the effectiveness of a password in resisting attempts at guessing and brute-force attacks.

Password complexity is one factor of several that determines how long it takes to crack a password. The more complex the password, the greater the number of possible combinations that need to be tested before the password is compromised.

Audit:

Run the following command to verify that the `enforce_for_root` option is enabled in a pwquality configuration file:

```
# grep -Psi -- '^h*enforce_for_root\b' /etc/security/pwquality.conf
/etc/security/pwquality.conf.d/*.conf
```

Example output:

```
/etc/security/pwquality.conf.d/50-pwroot.conf:enforce_for_root
```

Note:

- Settings observe an order of precedence:
 - module arguments override the settings in the `/etc/security/pwquality.conf` configuration file
 - settings in the `/etc/security/pwquality.conf` configuration file override settings in a `.conf` file in the `/etc/security/pwquality.conf.d/` directory
 - settings in a `.conf` file in the `/etc/security/pwquality.conf.d/` directory are read in canonical order, with last read file containing the setting taking precedence
- It is recommended that settings be configured in a `.conf` file in the `/etc/security/pwquality.conf.d/` directory for clarity, convenience, and durability.

Remediation:

Edit or add the following line in a `*.conf` file in `/etc/security/pwquality.conf.d` or in `/etc/security/pwquality.conf`:

Example:

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

{
    [ ! -d /etc/security/pwquality.conf.d/ ] && mkdir
    /etc/security/pwquality.conf.d/
    printf '\n%s\n' "enforce_for_root" > /etc/security/pwquality.conf.d/50-
    pwroot.conf
}
```






Default Value:

disabled

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.2 <u>Use Unique Passwords</u> 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 <u>Use Unique Passwords</u> 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.			

5.3.3.3 Configure pam_pwhistory module

pam_pwhistory - PAM module to remember last passwords

pam_history.so module - This 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.

Options:

- **debug** - Turns on debugging via syslog(3).
- **use_authtok** - When password changing enforce the module to use the new password provided by a previously stacked password module (this is used in the example of the stacking of the **pam_passwdqc** module documented below).
- **enforce_for_root** - If this option is set, the check is enforced for root, too.
- **remember=<N>** - The last <N> passwords for each user are saved. The default is **10**. Value of **0** makes the module to keep the existing contents of the opasswd file unchanged.
- **retry=<N>** - Prompt user at most <N> times before returning with error. The default is **1**.
- **authtok_type=<STRING>** - See pam_get_authtok(3) for more details.

Examples:

An example password section would be:

```
##PAM-1.0
password      required      pam_pwhistory.so
password      required      pam_unix.so          use_authtok
```

In combination with pam_passwdqc:

```
##PAM-1.0
password      required      pam_passwdqc.so    config=/etc/passwdqc.conf
password      required      pam_pwhistory.so    use_authtok
password      required      pam_unix.so          use_authtok
```

5.3.3.3.1 Ensure password history remember is configured (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The `/etc/security/opasswd` file stores the users' old passwords and can be checked to ensure that users are not recycling recent passwords. The number of passwords remembered is set via the `remember` argument value in `set` for the `pam_pwhistory` module.

- `remember=<N>` - `<N>` is the number of old passwords to remember

Rationale:

Requiring users not to reuse their passwords make it less likely that an attacker will be able to guess the password or use a compromised password.

Note: These change only apply to accounts configured on the local system.

Audit:

Run the following command and verify:

- The `pwhistory` line in `/etc/pam.d/common-password` includes `remember=<N>`
- The value of `<N>` is `24` or more
- The value meets local site policy

```
# grep -Psi --
'^\h*password\h+([\#\n\r]+\h+pam_pwhistory\.so\h+([\#\n\r]+\h+)?remember=\d+\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_authok
```

Remediation:

Run the following command:

```
# awk '/Password-Type:/{ f = 1;next } /-Type:/{ f = 0 } f {if (/pam_pwhistory\.so/) print FILENAME}' /usr/share/pam-configs/*
```

Edit any returned files and edit or add the **remember=** argument, with a value of **24** or more, that meets local site policy to the **pam_pwhistory** line in the **Password** section:

Example File:

```
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_authok # <- **ensure line includes remember=<N>**
```

Run the following command to update the files in the **/etc/pam.d/** directory:

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






Example:

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

References:

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

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		

5.3.3.3.2 Ensure password history is enforced for the root user (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

If the **pwhistory enforce_for_root** option is enabled, the module will enforce password history for the root user as well

Rationale:

Requiring users not to reuse their passwords make it less likely that an attacker will be able to guess the password or use a compromised password

Note: These change only apply to accounts configured on the local system.

Audit:

Run the following command to verify that the **enforce_for_root** argument is exists on the **pwhistory** line in **/etc/pam.d/common-password**:

```
# grep -Psi --  
'^\h*password\h+([\#\n\r]+\h+pam_pwhistory\.so\h+([\#\n\r]+\h+)?enforce_for_ro  
ot\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_authok
```

Remediation:

Run the following command:

```
# awk '/Password-Type:/{ f = 1;next } /-Type:/{ f = 0 } f {if (/pam_pwhistory\.so/) print FILENAME}' /usr/share/pam-configs/*
```

Edit any returned files and add the **enforce_for_root** argument to the **pam_pwhistory** line in the **Password** section:

Example File:

```
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_authok # <- **ensure line includes enforce_for_root**
```

Run the following command to update the files in the **/etc/pam.d/** directory:

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

Example:

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

Default Value:

disabled

References:

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

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

5.3.3.3.3 Ensure pam_pwhistory includes use_authtok (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

use_authtok - When password changing enforce the module to set the new password to the one provided by a previously stacked password module

Rationale:

use_authtok allows multiple pam modules to confirm a new password before it is accepted.

Audit:

Run the following command to verify that the **use_authtok** argument exists on the **pwhistory** line in **/etc/pam.d/common-password**:

```
# grep -Psi --  
'^\h*password\h+([\#\n\r]+\h+pam_pwhistory\.so\h+([\#\n\r]+\h+)?use_authtok\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 command:

```
# awk '/Password-Type:/{ f = 1;next } /-Type:/{ f = 0 } f {if (/pam_pwhistory\.so/) print FILENAME}' /usr/share/pam-configs/*
```

Edit any returned files and add the **use_authok** argument to the **pam_pwhistory** line in the **Password** section:

Example File:

```
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_authok # <- **ensure line includes use_authok**
```

Run the following command to update the files in the **/etc/pam.d/** directory:

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

Example:

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

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.11 <u>Encrypt Sensitive Data at Rest</u> 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 <u>Encrypt or Hash all Authentication Credentials</u> 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

5.3.3.4 Configure pam_unix module

The `pam_unix.so` module 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 the `/etc/shadow` file as well if shadow is enabled.

5.3.3.4.1 Ensure pam_unix does not include nullok (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The **nullok** argument overrides the default action of **pam_unix.so** to not permit the user access to a service if their official password is blank.

Rationale:

Using a strong password is essential to helping protect personal and sensitive information from unauthorized access

Audit:

Run the following command to verify that the **nullok** argument is not set on the **pam_unix.so** module:

```
# grep -PH -- '^\\h*^\\h*[\\^#\\n\\r]+\\h+pam_unix\\.so\\b' /etc/pam.d/common-  
{password,auth,account,session,session-noninteractive} | grep -Pv --  
'\\bnullok\\b'
```

Output should be similar to:

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

Remediation:

Run the following command:

```
# grep -PH -- '^h*([^\n\r]+\h+)?pam_unix\.so\h+([^\n\r]+\h+)?nullok\b' /usr/share/pam-configs/*
```

Edit any files returned and remove the **nullok** argument for the **pam_unix** lines

Example File:

```
Name: Unix authentication
Default: yes
Priority: 256
Auth-Type: Primary
Auth:
    [success=end default=ignore]    pam_unix.so try_first_pass # <-
**ensure line does not include nullok nullok**
Auth-Initial:
    [success=end default=ignore]    pam_unix.so # <- **ensure line does
not include nullok nullok**
Account-Type: Primary
Account:
    [success=end new_authtok_reqd=done default=ignore]    pam_unix.so
Account-Initial:
    [success=end new_authtok_reqd=done default=ignore]    pam_unix.so
Session-Type: Additional
Session:
    required    pam_unix.so
Session-Initial:
    required    pam_unix.so
Password-Type: Primary
Password:
    [success=end default=ignore]    pam_unix.so obscure use_authtok
try_first_pass yescrypt
Password-Initial:
    [success=end default=ignore]    pam_unix.so obscure yescrypt
```

Run the following command to update the files in the **/etc/pam.d/** directory:






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

Example:

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

Note: If custom files are being used, the corresponding files in **/etc/pam.d/** would need to be edited directly, and the **pam-auth-update --enable <EDITED_PROFILE_NAME>** command skipped

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
T1003, T1003.008, T1110, T1110.002	TA0006	M1041

5.3.3.4.2 Ensure pam_unix does not include remember (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

The `remember=n` argument saves the last n passwords for each user in `/etc/security/opasswd` in order to force password change history and keep the user from alternating between the same password too frequently. The MD5 password hash algorithm is used for storing the old passwords. Instead of this option the `pam_pwhistory` module should be used. The `pam_pwhistory` module saves the last n passwords for each user in `/etc/security/opasswd` using the password hash algorithm set on the `pam_unix` module. This allows for the `yescrypt` or `sha512` hash algorithm to be used.

Rationale:

The `remember=n` argument should be removed to ensure a strong password hashing algorithm is being used. A stronger hash provides additional protection to the system by increasing the level of effort needed for an attacker to successfully determine local user's old passwords stored in `/etc/security/opasswd`.

Audit:

Run the following command to verify that the `remember` argument is not set on the `pam_unix.so` module:

```
# grep -PH -- '^h*\^h*[^#\n\r]+\h+pam_unix\.so\b' /etc/pam.d/common-  
{password,auth,account,session,session-noninteractive} | grep -Pv --  
'\bremember=\d+\b'
```

Output should be similar to:

```
/etc/pam.d/common-password:password [success=1 default=ignore]  
pam_unix.so obscure yescrypt  
/etc/pam.d/common-auth:auth [success=1 default=ignore] pam_unix.so  
/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-session-noninteractive:session required pam_unix.so
```


Remediation:

Run the following command:

```
# grep -PH -- '^h*([\#\n\r]+\h+)?pam_unix\.so\h+([\#\n\r]+\h+)?remember\b' /usr/share/pam-configs/*
```

Edit any files returned and remove the **remember=<N>** argument for the **pam_unix** lines

Example output:

```
[success=end default=ignore] pam_unix.so obscure use_authtok try_first_pass yescrypt remember=5 # **<- remove remember=<N>**  
[success=end default=ignore] pam_unix.so obscure yescrypt remember=5 # **<- remove remember=<N>**
```

Run the following command to update the files in the **/etc/pam.d/** directory:






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

Example:

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

Note: If custom files are being used, the corresponding files in **/etc/pam.d/** would need to be edited directly, and the **pam-auth-update --enable <EDITED_PROFILE_NAME>** command skipped

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
T1003, T1003.008, T1110, T1110.002	TA0006	M1041

5.3.3.4.3 Ensure pam_unix includes a strong password hashing algorithm (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.

The **pam_unix** module can be configured to use one of the following hashing algorithms for user's passwords:

- **md5** - When a user changes their password next, encrypt it with the **MD5** algorithm.
- **bigcrypt** - When a user changes their password next, encrypt it with the **DEC C2** algorithm.
- **sha256** - When a user changes their password next, encrypt it with the **SHA256** algorithm. The **SHA256** algorithm must be supported by the crypt(3) function.
- **sha512** - When a user changes their password next, encrypt it with the **SHA512** algorithm. The **SHA512** algorithm must be supported by the crypt(3) function.
- **blowfish** - When a user changes their password next, encrypt it with the **blowfish** algorithm. The **blowfish** algorithm must be supported by the crypt(3) function.
- **gost-yescrypt** - When a user changes their password next, encrypt it with the **gost-yescrypt** algorithm. The **gost-yescrypt** algorithm must be supported by the crypt(3) function.
- **yescrypt** - When a user changes their password next, encrypt it with the **yescrypt** algorithm. The **yescrypt** algorithm must be supported by the crypt(3) function.

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 user passwords.

Note: These changes only apply to the local system.

Audit:

Run the following command to verify that a strong password hashing algorithm is set on the `pam_unix.so` module:

```
# grep -PH --
'^\h*password\h+([\#\n\r]+)\h+pam_unix\.so\h+([\#\n\r]+\h+)?(sha512|yescrypt)
\b' /etc/pam.d/common-password
```

Output should be similar to:

```
/etc/pam.d/common-password:password [success=1 default=ignore]
pam_unix.so obscure use_authtok try_first_pass yescrypt
```

Verify that the line(s) include either **sha512** - OR - **yescrypt**

Remediation:

Run the following command:

```
# awk '/Password-Type:/{ f = 1;next } /-Type:/{ f = 0 } f {if
(/pam_unix\.so/) print FILENAME}' /usr/share/pam-configs/*
```

Edit any returned files and edit or add a strong hashing algorithm, either `sha512` or `yescrypt`, that meets local site policy to the `pam_unix` lines in the **Password** section:

Example File:

```
Name: Unix authentication
Default: yes
Priority: 256
Auth-Type: Primary # <- Start of "Auth" section
Auth:
    [success=end default=ignore]    pam_unix.so try_first_pass
Auth-Initial:
    [success=end default=ignore]    pam_unix.so
Account-Type: Primary # <- Start of "Account" section
Account:
    [success=end new_authtok_reqd=done default=ignore]    pam_unix.so
Account-Initial:
    [success=end new_authtok_reqd=done default=ignore]    pam_unix.so
Session-Type: Additional # <- Start of "Session" section
Session:
    required    pam_unix.so
Session-Initial:
    required    pam_unix.so
Password-Type: Primary # <- Start of "Password" section
Password:
    [success=end default=ignore]    pam_unix.so obscure use_authtok
try_first_pass yescrypt # <- **ensure hashing algorithm is either sha512 or
yescrypt**
Password-Initial:
    [success=end default=ignore]    pam_unix.so obscure yescrypt # <-
**ensure hashing algorithm is either sha512 or yescrypt**
```

Run the following command to update the files in the `/etc/pam.d/` directory:

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

Example:

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

References:





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

Additional Information:

The following command may be used to expire all non-system user ID's immediately and force them to change their passwords on next login. Any system accounts that need to be expired should be carefully done separately by the system administrator to prevent any potential problems.

```
# awk -F: ' ( $3<"$(awk '/^\s*UID_MIN/{print $2}' /etc/login.defs)"' && $1 != "nfsnobody" ) { print $1 }' /etc/passwd | xargs -n 1 chage -d 0
```

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.11 <u>Encrypt Sensitive Data at Rest</u> 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 <u>Encrypt or Hash all Authentication Credentials</u> 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

5.3.3.4.4 Ensure pam_unix includes use_authtok (Automated)

Profile Applicability:

- Level 1 - Server
- Level 1 - Workstation

Description:

use_authtok - When password changing enforce the module to set the new password to the one provided by a previously stacked password module

Rationale:

use_authtok allows multiple pam modules to confirm a new password before it is accepted.

Audit:

Run the following command to verify that **use_authtok** is set on the pam_unix.so module lines in the password stack:

```
# grep -PH --  
'^\h*password\h+([\h#\n\r]+\h)pam_unix\.so\h+([\h#\n\r]+\h+)?use_authtok\b'  
/etc/pam.d/common-password
```

Output should be similar to:

```
/etc/pam.d/common-password:password    [success=1 default=ignore]  
pam_unix.so obscure use_authtok try_first_pass yescrypt
```

Verify that the line(s) include **use_authtok**

Remediation:

Run the following command:

```
# awk '/Password-Type:/{ f = 1;next } /-Type:/{ f = 0 } f {if (/pam_unix\.so/) print FILENAME}' /usr/share/pam-configs/*
```

Edit any returned files add **use_authtok** to the **pam_unix** line in the **Password** section under **Password:** subsection:

Note: The if the file's **Password** section includes a **Password-Initial:** subsection, **use_authtok** should not be added to the **pam_unix** line in the **Password-Initial:** subsection

Example File:

```
Name: Unix authentication
Default: yes
Priority: 256
Auth-Type: Primary # <- Start of "Auth" section
Auth:
    [success=end default=ignore]    pam_unix.so try_first_pass
Auth-Initial:
    [success=end default=ignore]    pam_unix.so
Account-Type: Primary # <- Start of "Account" section
Account:
    [success=end new_authtok_reqd=done default=ignore]    pam_unix.so
Account-Initial:
    [success=end new_authtok_reqd=done default=ignore]    pam_unix.so
Session-Type: Additional # <- Start of "Session" section
Session:
    required    pam_unix.so
Session-Initial:
    required    pam_unix.so
Password-Type: Primary # <- Start of "Password" section
Password:
    [success=end default=ignore]    pam_unix.so obscure use_authtok
try_first_pass yescrypt # <- **ensure line includes use_authtok**
Password-Initial:
    [success=end default=ignore]    pam_unix.so obscure yescrypt # <-
**Password-Initial: subsection does not include use_authtok
```

Run the following command to update the files in the **/etc/pam.d/** directory:

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





Example:

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

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.11 <u>Encrypt Sensitive Data at Rest</u> 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 <u>Encrypt or Hash all Authentication Credentials</u> 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