Day 33

Exploitation Analyst

User Management and PAM:

Set Password Expiration for Users:

Why should password expiration be enforced?

Forcing regular password changes limits the window of exposure in case a password is compromised.

Benefits:

1. **Mitigates leaked credentials**: If a password leaks, it will eventually be useless.
2. **Encourages proactive updates**: Promotes better hygiene, especially in long-term systems.
3. **Meets compliance**: Standards like HIPAA, PCI-DSS require expiration policies.

Performing Expiration Setup:

Steps:

Use the following command to do so:

```
┌─[x]─[root@parrot]─[/home/user]
└──╼ #sudo chage -M 90 -m 7 -W 7 Aditya
```

- **-M 90** → max password age = 90 days
- **-m 7** → min password age = 7 days
- **-W 7** → warn user 7 days before expiry

What actually happened in background?

When you run a command like ***sudo chage -M 90 -m 7 -W 7 username***, the system doesn't start a background timer; instead, it updates the user's entry in /etc/shadow. This file contains fields such as the date of the last password change (lastchg), minimum days before another change (min), maximum days before expiry (max), and days to warn before expiry (warn). For example, after the change, the entry might look like ***aditya:6abc123...:19675:7:90:7:::.*** When the user logs in, PAM or the login program reads these values and checks the current date against them—if the maximum age is exceeded, the password is marked expired, and if it's within the warning period, a notice is shown. This means password expiration is enforced at login time through date comparison, not by any constantly running process.

Enforcing a consistent policy for all new users

Steps:

Open the /etc/login.defs using the nano:

```
┌─[root@parrot]─[/home/user]
└──╼ #nano /etc/login.defs
┌─[root@parrot]─[/home/user]
└──╼ #
```

Following screen will appear:

```
File  Edit  View  Search  Terminal  Help
  GNU nano 7.2                                                    /etc/login.defs
#
# /etc/login.defs - Configuration control definitions for the login package.
#
# Three items must be defined:  MAIL_DIR, ENV_SUPATH, and ENV_PATH.
# If unspecified, some arbitrary (and possibly incorrect) value will
# be assumed.  All other items are optional - if not specified then
# the described action or option will be inhibited.
#
# Comment lines (lines beginning with "#") and blank lines are ignored.
#
# Modified for Linux.  --marekm

# REQUIRED for useradd/userdel/usermod
#   Directory where mailboxes reside, _or_ name of file, relative to the
#   home directory.  If you _do_ define MAIL_DIR and MAIL_FILE,
#   MAIL_DIR takes precedence.
#
#   Essentially:
#     - MAIL_DIR defines the location of users mail spool files
#       (for mbox use) by appending the username to MAIL_DIR as defined
#       below.
#     - MAIL_FILE defines the location of the users mail spool files as the
#       fully-qualified filename obtained by prepending the user home
#       directory before $MAIL_FILE
#
# NOTE: This is no more used for setting up users MAIL environment variable
#       which is, starting from shadow 4.0.12-1 in Debian, entirely the
#       job of the pam_mail PAM modules
#       See default PAM configuration files provided for
                                                        [ Read 402 lines ]
^G Help     ^O Read File  ^R Replace   ^V Paste     ^C Go To Line  ^Y Redo      M-6 Copy      ^Q Where Was  M-W Next      ^F Forward    ^← Next Word
^X Exit     ^W Where Is   ^K Cut       ^T Execute   ^_ Undo        M-A Set Mark  M-] To Bracket M-Q Previous  ^B Back      ^→ Prev Word  ^A Home
```

Look for the following section:

```
#
# Password aging controls:
#
#   PASS_MAX_DAYS   Maximum number of days a password may be used.
#   PASS_MIN_DAYS   Minimum number of days allowed between password changes.
#   PASS_WARN_AGE   Number of days warning given before a password expires.
#
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_WARN_AGE   7
```

Edit it according to the need, and save it.

Which method is more efficient?

-> For one-off changes to specific users, chage is quicker and more direct.
-> For enforcing a consistent policy for all new users, /etc/login.defs is more efficient because it's automatic.

What actually happens in background?

When you edit /etc/login.defs and set parameters like PASS_MAX_DAYS, PASS_MIN_DAYS, and PASS_WARN_AGE, Linux doesn't instantly change any existing user accounts—instead, these values act as default templates used by useradd when creating new users. The useradd command reads these defaults and writes them into /etc/shadow for the new account, where actual password aging information is stored and enforced by PAM (Pluggable Authentication Modules) during login. In contrast, using chage directly modifies the corresponding password-aging fields for an existing user in /etc/shadow, which takes effect immediately. Thus, /etc/login.defs is efficient for automatic, system-wide policy for future users, while chage is better for quick, per-user adjustments.

--The End--