**Day 33**

**Exploitation Analyst**

**User Management and PAM:**

**Set Password Expiration for Users:**

**Why should password expiration be enforced?**

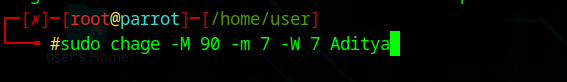
Forcing regular password changes limits the window of exposure in case a password is compromised.

**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:



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

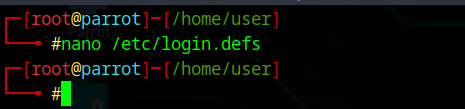
**What actually happened in background?**

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:$6$abc123...: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.

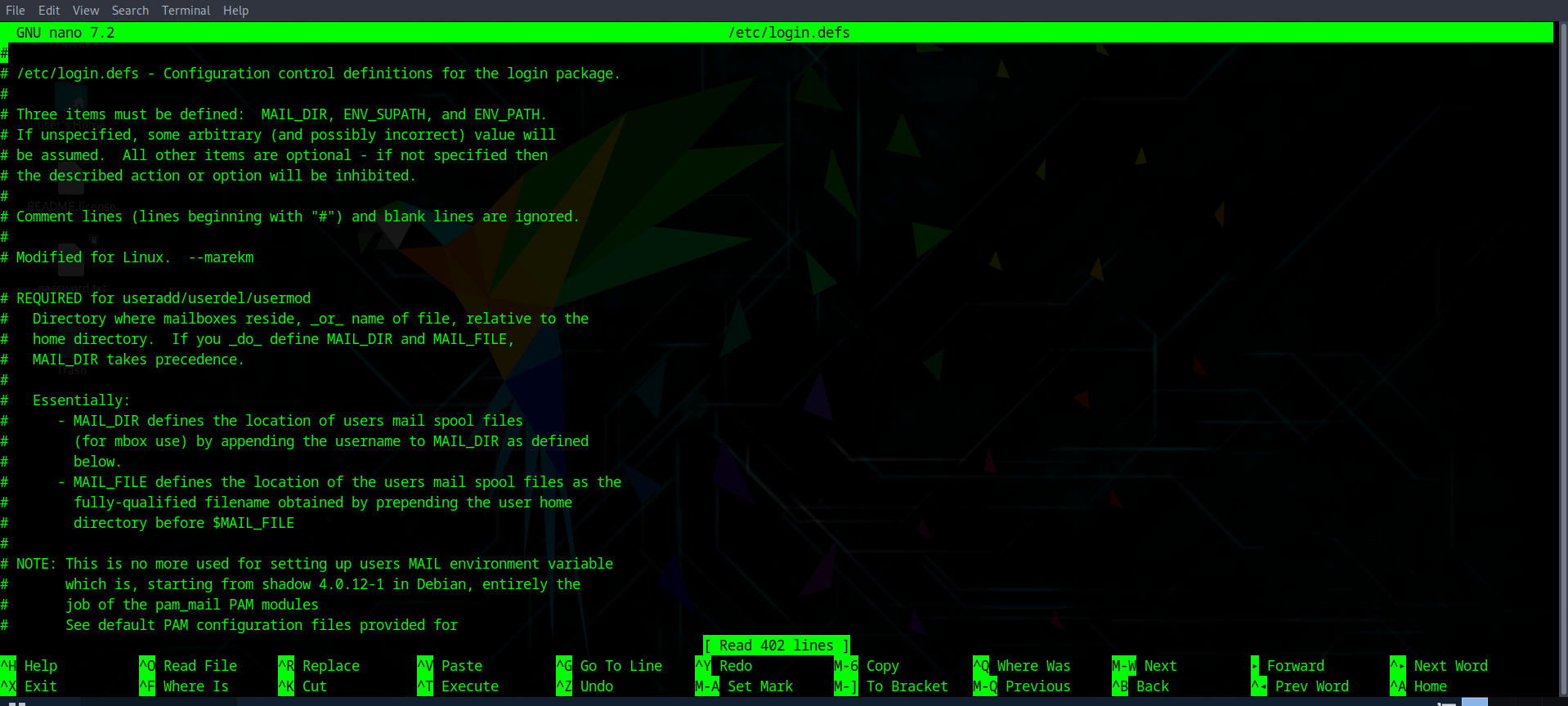
**Enforcing a consistent policy for all new users**

Steps:

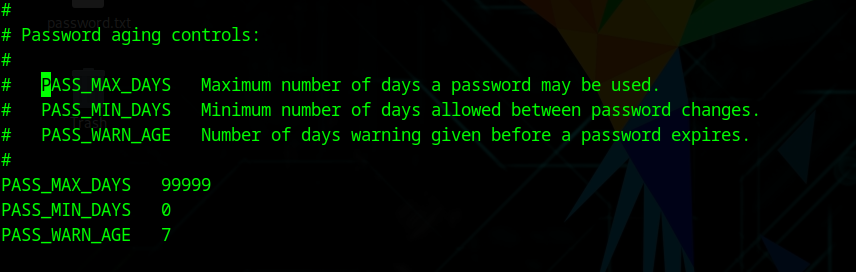
Open the /etc/login.defs using the nano:



Following screen will appear:



Look for the following section:



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--