Project: FileSystem AND Storage (Mount, fstab, crypttab, LUKS encryption, Simulated Boot Lockout

Project Done by: Aashish Chaudhary

Concepts:

fstab, crypttab, dd, lsblk, losetup, cryptsetup, luks, mkfs, fdisk

Overview:

In this project, I created a dummy file (simulates .img) from the root filesystem and used it to mount both temporarily and permanent using manual 'sudo mount /mnt/point' and later using /etc/fstab file. Later I manually set up encryption on dummy img using luksFormat and again mounted it to have system write the data. Finally, I simulated the Boot failure by messing up the config in /etc/fstab file and successfully, troubleshooted the Problem and successfully logged back in.

Section: Permanent Mount using /etc/fstab:

I have demonstrated how to do manual mount or any temporary mount on my previous project **'Dummy Disk Creation Project' under System Administrative Tasks repository.**

In this Parmanent Mounting attempt, I used the following steps:

RootFileSystem (/) --→create a dummy.file (.img)--->edit /etc/fstab file--→\

→test if any error →daemon-reload--→reboot---→verify

Let's Dive in what I actually did:

1. Create Dummy file:
    a. Run the following command
        i. dd if=/dev/zero of=dummy.disk bs=1M count=100
2. Edit /etc/fstab file"
    a. Add the following line change file dummy.disk path, and mount points according to your requirements, and options like defaults,ro,x-systemd.automount, dumps

    i. /home/bhairav0001/filesystem_Lab/dummy.disk /mnt/dummy ext4 defaults,loop,ro,x-systemd.automount 0 0
3. Test if any error:
    a. Use ‘sudo mount -a’ to test any configuration error up to this point
4. Reload the system daemon:
    i. sudo systemctl daemon-reload OR daemon-reexec
5. Finally reboot the system and verify the mount
    i. sudo reboot
    ii. mount |grep dummy

KeyPoints:

Config in /etc/fstab file, options sections can be modified as per our requirement. In my case, defaults option sets default options, ro sets Kernel Level ReadOnly Permission, x-systemd.automount mounts on demand (meaning it only mounts when the FS is touched or accessed) and this option also avoids bootfailure.

Note:

1. Use sudo privilege if permission denied
2. Don’t forget to test before running systemctl daemon-reload, otherwise, you might be locked out of the system during boot.

Section: Adding Encryption to the Disk:

Adding Encryption to the Disk is couple of steps extra before we mount the Encrypted Filesytstem to the mountpoint. Hence, in this section I have followed the following steps and done encryption manually and Parmanent:

i. Manual Encryption:

Mount-→unmount→ comment /etc/fstab-→attach img to block--→luksFormat-→open crypt container-→assign FS -→mount

ii. Encryption and mount at boot:

Unmount crypt container-→edit /etc/crypttab--→edit /etc/fstab --→add path to cryptcontainer-→test if error-→daemon-reload-→reboot-→enterpassphrase-→touchFS & Verify.

Note: Mounts Created during the boot attaches img file to blockdevice on its own and create FS using /etc/fstab configuration. When we unmount the mount point, we have to manually attach it to the block device and create FS into it. Similarly, it's same for the encrypted container as well.

After the first passphrase setup, during bootup, it automatically creates encrypted container in /dev/mapper/dummy_encrypt. Only difference this time is, in /etc/fstab file, we are mapping the dummy_encrypt path to the img file instead of loop device as in our previous attempts.

a. /etc/crypttab file config:

- # <target name> <source device> <key file> <options>

  dummy_encrypt /home/bhairav0001/filesystem_Lab/dummy.disk none luks,loop

b./etc/fstab file config:

#/home/bhairav0001/filesystem_Lab/dummy.disk /mnt/dummy ext4 defaults,loop,ro 0 0 ---> **just Filesystem**

/dev/mapper/dummy_encrypt /mnt/dummy ext4 ro,defaults,x-systemd.automount 0 0 ----->**Encrypted FileSystem**

c.daemon -reload, verify and reboot:

Daemon reload is essential because the system uses the current or I would say, previously activated daemon processes. In order to bring up the new changes in fstab, or crypttab, daemon reload is very important.

- sudo systemctl daemon-reload
- sudo reboot
- mount |grep dummy

d. Output must show autofs in the systemd-1 fs.

Since I have added, x-systemd.automount option in /etc/fstab file. It does not create the FS until the mount point is accessed or touched. Until then, 'mount |grep dummy' command shows the temporarily not active mount with 'autofs' into it.

On successful test:

Prompt asks for the passphrase and creates the mount point

Section: Break the FileSystem and Lock yourself out of the Linux Machine

In this exciting part of the lab, I followed the following steps:

/etc/fstab file ------→broke path to crypt_container------→removed x-systemd.automount-----→test error -→system daemon reload-→reboot→entered passphrase prompt-→bootfailure---→entered emergencyshell --→remounted FS as rw, -→opened changed /etc/fstab----→ exit and ----→the system boots on

Important Things to Remember:

i. access to the root file system during boot is read-only mode.
ii. So, remount this Filesystem with read-write permission
    a. mount -o remount,rw /

Note: While using encrypted Container during reboot, the boot console asks for the passphrase. And it won't let you skip it. If you're doing it on remote machine, you might not get access back to your machine, because it breaks ssh. So I suggest using VM consoles or local sandbox.