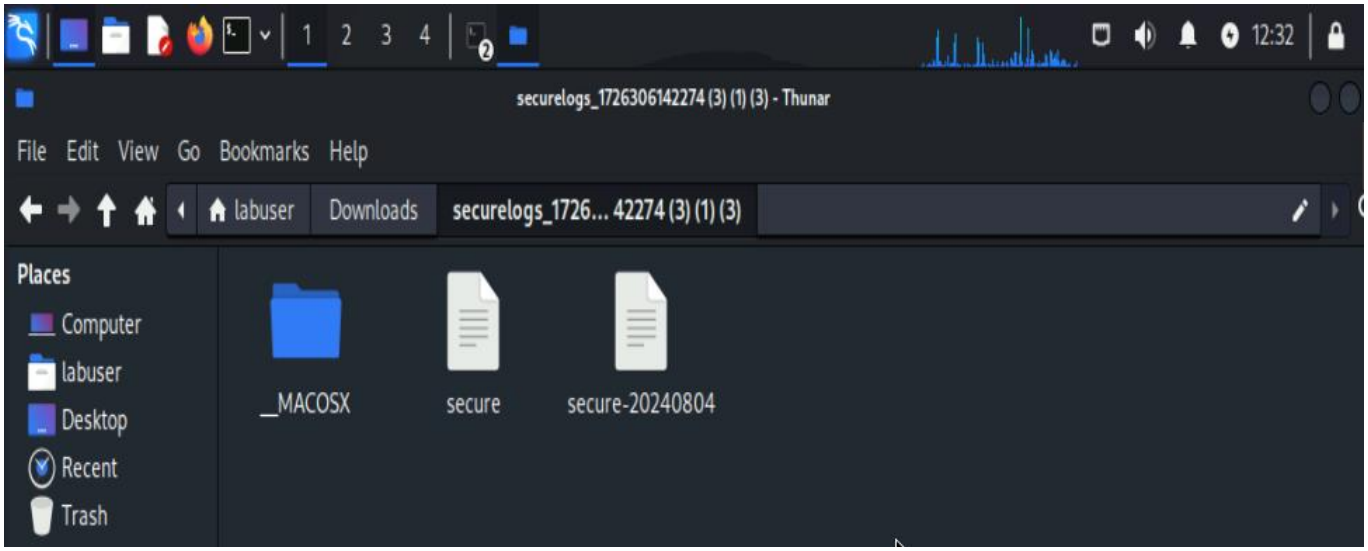
Brute Force Attack

Problem Statement:

Conduct a comprehensive security assessment and response for a CentOS VM under brute force attack, focusing on log analysis, user verification, and implementation of enhanced security measures to mitigate future threats.

Task 1: Download Authentication Logs: Access and download authentication logs from the provided URL. These logs contain critical evidence of brute force attacks, including access attempts and usernames.

Step 1: I access from the kali Linux machine the portal of “Simplilearn” and download the needed secure zip file. Step 2: I open the “Terminal” then write (CD \download path), (unzip “folder”).

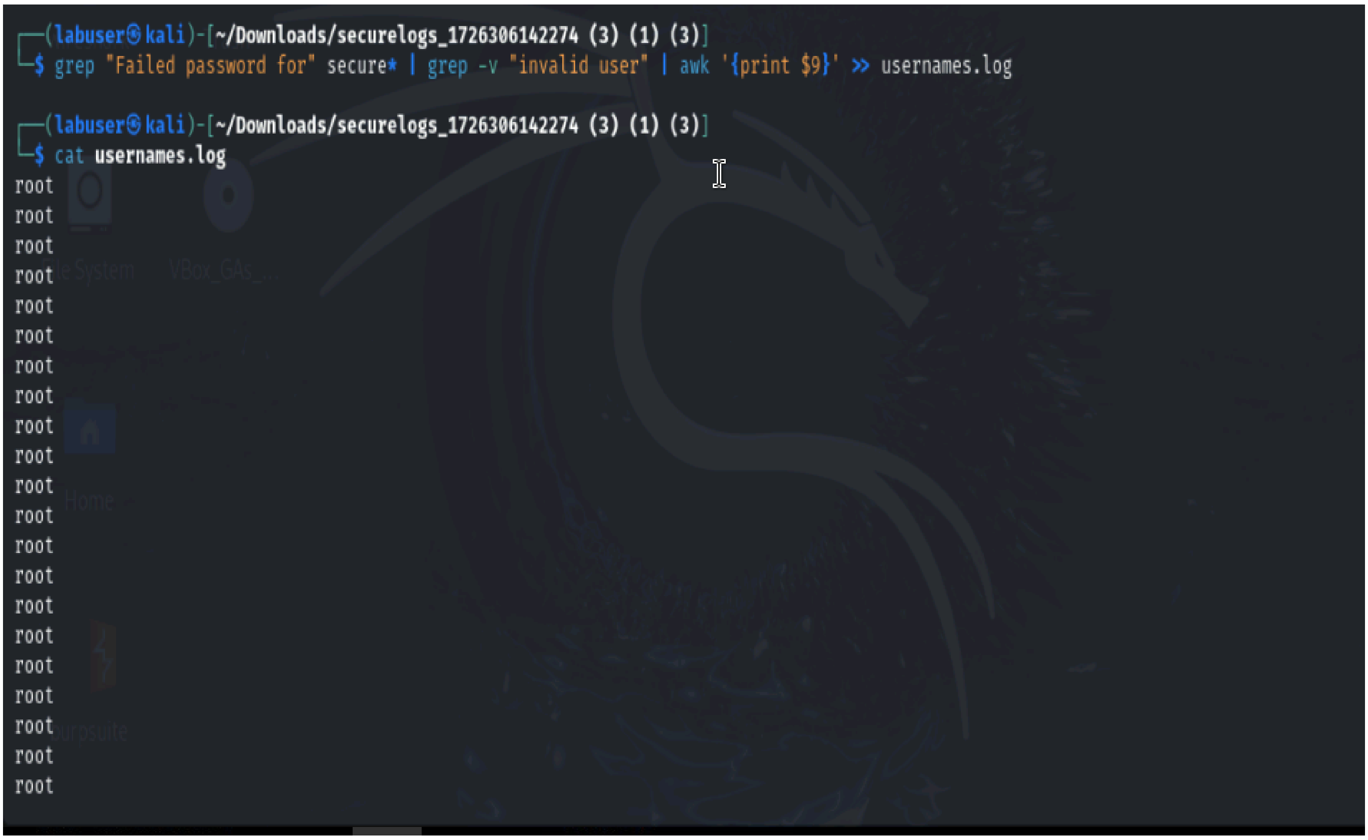


Task 2: Analyze the Logs for Usernames: Use log analysis tools or scripts to extract all usernames attempted during the attack, identifying the extent and specific entry points targeted.

Step 1: looking at contents of log files

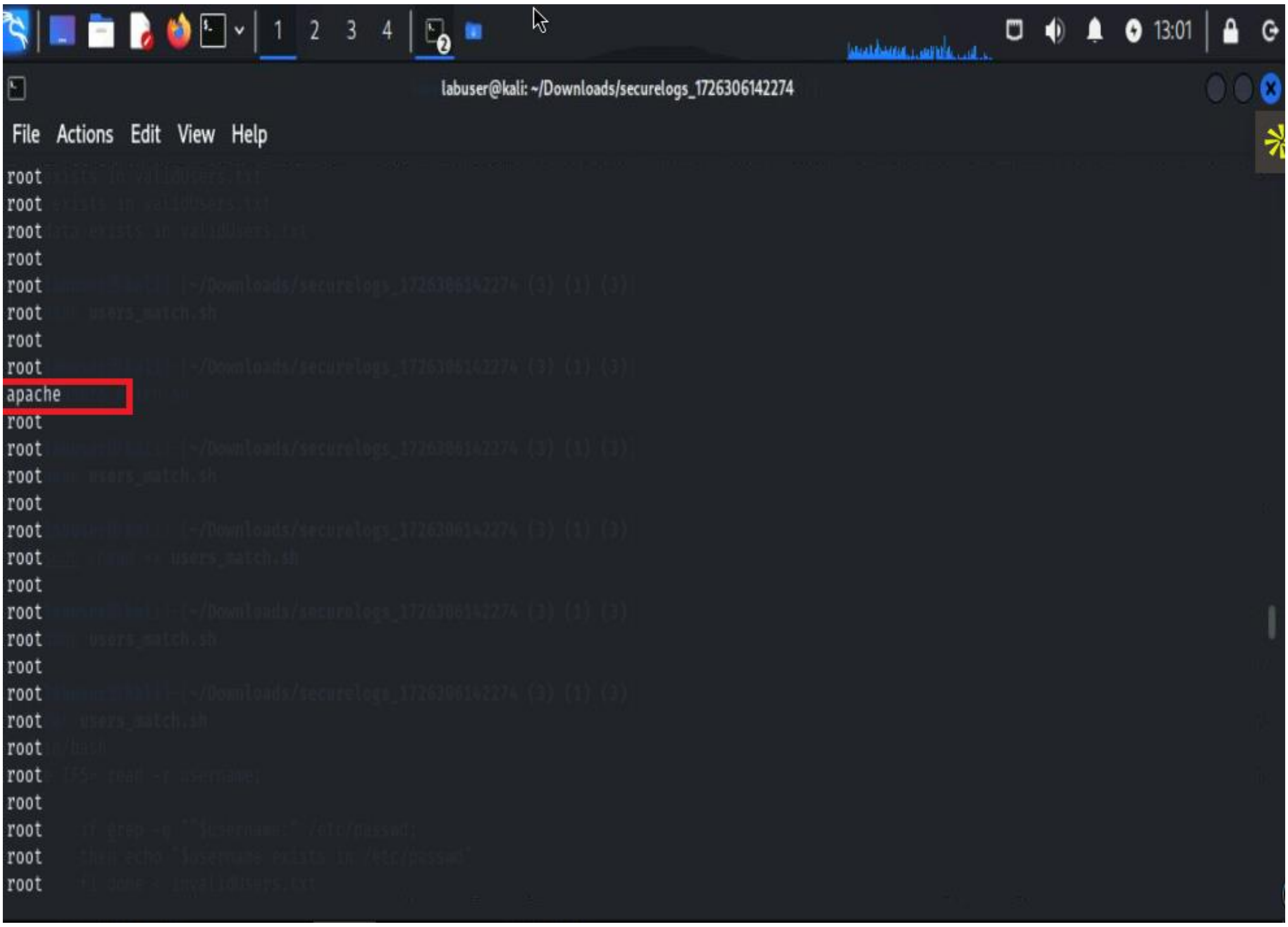
```
labuser@kali: ~/Downloads/securelogs_1726306142274
File  Actions  Edit  View  Help
└─$ cat secure-20240804
Jul 28 07:02:46 honeypot-vicesolutions-sevayan sshd[6810]: Failed password for root from 221.122.88.121 port 59086 ssh2
Jul 28 07:02:46 honeypot-vicesolutions-sevayan sshd[6810]: Connection closed by 221.122.88.121 port 59086 [preauth]
Jul 28 07:02:49 honeypot-vicesolutions-sevayan sshd[6846]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh rus
er= rhost=221.122.88.121  user=root
Jul 28 07:02:50 honeypot-vicesolutions-sevayan sshd[6846]: Failed password for root from 221.122.88.121 port 33114 ssh2
Jul 28 07:02:51 honeypot-vicesolutions-sevayan sshd[6846]: Connection closed by 221.122.88.121 port 33114 [preauth]
Jul 28 07:02:54 honeypot-vicesolutions-sevayan sshd[6855]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh rus
er= rhost=221.122.88.121  user=root
Jul 28 07:02:56 honeypot-vicesolutions-sevayan sshd[6855]: Failed password for root from 221.122.88.121 port 35120 ssh2
Jul 28 07:02:56 honeypot-vicesolutions-sevayan sshd[6855]: Connection closed by 221.122.88.121 port 35120 [preauth]
Jul 28 07:02:58 honeypot-vicesolutions-sevayan sshd[6861]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh rus
er= rhost=221.122.88.121  user=root
Jul 28 07:03:00 honeypot-vicesolutions-sevayan sshd[6861]: Failed password for root from 221.122.88.121 port 37522 ssh2
Jul 28 07:03:01 honeypot-vicesolutions-sevayan sshd[6861]: Connection closed by 221.122.88.121 port 37522 [preauth]
Jul 28 07:03:03 honeypot-vicesolutions-sevayan sshd[6867]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh rus
er= rhost=221.122.88.121  user=root
Jul 28 07:03:06 honeypot-vicesolutions-sevayan sshd[6867]: Failed password for root from 221.122.88.121 port 39724 ssh2
Jul 28 07:03:06 honeypot-vicesolutions-sevayan sshd[6867]: Connection closed by 221.122.88.121 port 39724 [preauth]
Jul 28 07:03:08 honeypot-vicesolutions-sevayan sshd[6873]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh rus
er= rhost=221.122.88.121  user=root
Jul 28 07:03:10 honeypot-vicesolutions-sevayan sshd[6873]: Failed password for root from 221.122.88.121 port 41954 ssh2
Jul 28 07:03:11 honeypot-vicesolutions-sevayan sshd[6873]: Connection closed by 221.122.88.121 port 41954 [preauth]
Jul 28 07:03:13 honeypot-vicesolutions-sevayan sshd[6881]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh rus
er= rhost=221.122.88.121  user=root
Jul 28 07:03:15 honeypot-vicesolutions-sevayan sshd[6881]: Failed password for root from 221.122.88.121 port 44178 ssh2
Jul 28 07:03:16 honeypot-vicesolutions-sevayan sshd[6881]: Connection closed by 221.122.88.121 port 44178 [preauth]
```

Step 2: getting usernames from secure logs and storing it to file named ‘username.log’



```
┌──(labuser㉿kali)-[~/Downloads/securelogs_1726306142274 (3) (1) (3)]
└─$ grep "Failed password for" secure* | grep -v "invalid user" | awk '{print $9}' >> usernames.log

┌──(labuser㉿kali)-[~/Downloads/securelogs_1726306142274 (3) (1) (3)]
└─$ cat usernames.log
root
root
root
root
root
root
root
root
root
root
root
root
root
root
root
root
root
root
root
root
root
```



```
root
root
root
root
root
root
root
root
apache
root
root
root
root
root
root
root
root
root
root
root
root
root
root
root
root
root
root
```

Step 3: listing the content of securelog folder to see if the files created

```
┌──(labuser㉿kali)-[~/Downloads/securelogs_1726306142274 (3) (1) (3)]
└─$ ls -la
total 13644
drwxrwxr-x 3 labuser labuser     4096 Aug 10 11:15 .
drwxr-xr-x 7 labuser labuser     4096 Aug 10 10:35 ..
-rw-rw-r-- 1 labuser labuser     1024 Aug 10 11:09 .secure.swp
drwxrwxr-x 2 labuser labuser     4096 Aug 10 10:35 __MACOSX
-rw-r--r-- 1 labuser labuser  3731304 Sep 14  2024 secure
-rw-r--r-- 1 labuser labuser 10092544 Sep 14  2024 secure-20240804
-rw-rw-r-- 1 labuser labuser   130076 Aug 10 11:15 usernames.log

┌──(labuser㉿kali)-[~/Downloads/securelogs_1726306142274 (3) (1) (3)]
└─$
```

Step 4: extracting the invalid usernames form secure logs and storing it into invalidUsers.txt file

```
┌──(labuser㉿kali)-[~/Downloads/securelogs_1726306142274 (3) (1) (3)]
└─$ cat secure* | grep ssh | grep user | grep Invalid | cut -d " " -f 8 | sort -u >> invalidUsers.txt

┌──(labuser㉿kali)-[~/Downloads/securelogs_1726306142274 (3) (1) (3)]
└─$ ls -la
total 13652
drwxrwxr-x 3 labuser labuser     4096 Aug 10 11:28 .
drwxr-xr-x 7 labuser labuser     4096 Aug 10 10:35 ..
-rw-rw-r-- 1 labuser labuser     1024 Aug 10 11:09 .secure.swp
drwxrwxr-x 2 labuser labuser     4096 Aug 10 10:35 __MACOSX
-rw-rw-r-- 1 labuser labuser     5361 Aug 10 11:28 invalidUsers.txt
-rw-r--r-- 1 labuser labuser  3731304 Sep 14  2024 secure
-rw-r--r-- 1 labuser labuser 10092544 Sep 14  2024 secure-20240804
-rw-rw-r-- 1 labuser labuser   130076 Aug 10 11:15 usernames.log

┌──(labuser㉿kali)-[~/Downloads/securelogs_1726306142274 (3) (1) (3)]
└─$
```

Step 5: Counting the invalid users in invalidUsers.txt

```
┌──(labuser㉿kali)-[~/Downloads/securelogs_1726306142274]
└─$ cat invalidUsers.txt | wc -l
802
```

- There is total 802 invalid users

Task 3: Cross-Reference Usernames with Company Records: Cross-reference extracted usernames with the internal user database to check if any correspond to actual user accounts, indicating potential insider threats.

Step 1: Creating script named ‘users_match.sh’ to check if any of the invalid users are present in /etc/passwd where all the valid users are listed

```
GNU nano 8.0                                        users_match.sh
#!/bin/bash
while IFS= read -r username;
do
        if grep -q "^$username:" /etc/passwd;
        then echo "$username exists in validUsers.txt"
        fi done < invalidUsers.txt
```

Step 2: setting ‘user_match.sh’ execution permission and executing the script

```
┌──(labuser㉿kali)-[~/Downloads/securelogs_1726306142274 (3) (1) (3)]
└─$ sudo chmod +x users_match.sh

┌──(labuser㉿kali)-[~/Downloads/securelogs_1726306142274 (3) (1) (3)]
└─$ cat users_match.sh
#!/bin/bash
while IFS= read -r username;
do
        if grep -q "^$username:" /etc/passwd;
        then echo "$username exists in validUsers.txt"
        fi done < invalidUsers.txt

┌──(labuser㉿kali)-[~/Downloads/securelogs_1726306142274 (3) (1) (3)]
└─$ ./users_match.sh
backup exists in validUsers.txt
news exists in validUsers.txt
proxy exists in validUsers.txt
sys exists in validUsers.txt
uucp exists in validUsers.txt
www-data exists in validUsers.txt

┌──(labuser㉿kali)-[~/Downloads/securelogs_1726306142274 (3) (1) (3)]
└─$
```

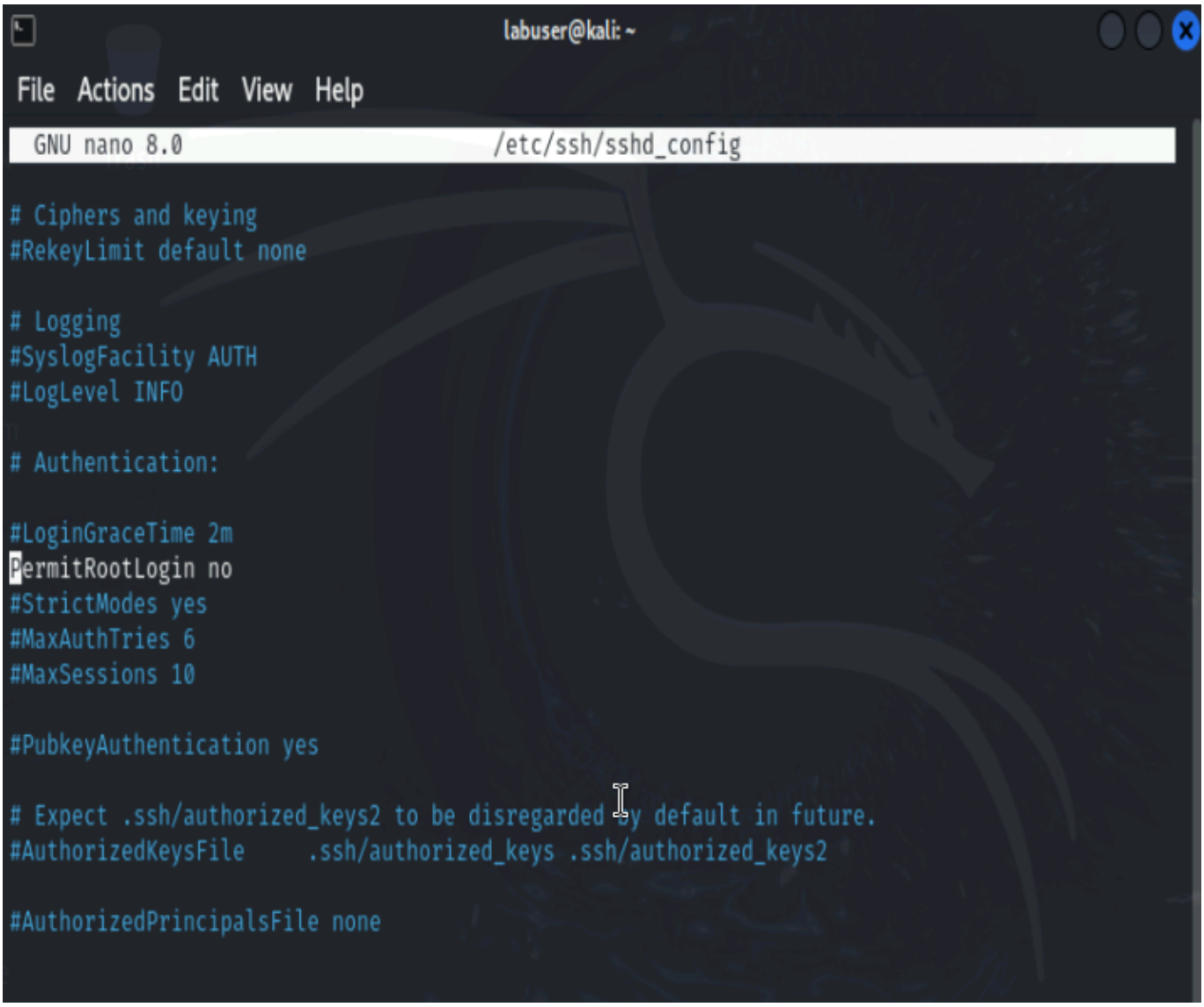
- As we can see that only few of the users listed in ‘invalidUsers.txt’ have matched with users listed in /etc/passwd and rest of the usernames are in ‘invalidUsers.txt’ are invalid.
- So we can conclude that a Brute force attack was carried.

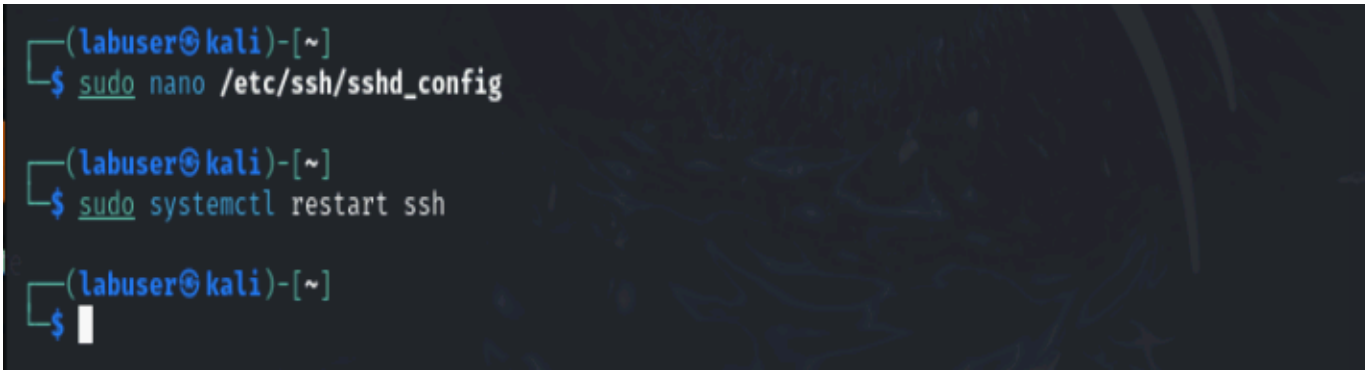
Task 4: Implement Security Enhancements: Based on findings, enhance security by enforcing stricter password policies, implementing multifactor authentication, and possibly changing SSH ports.

Step 1: Disabling Root login by using ‘PermitRootLogin no’ in /etc/ssh/sshd_config and then restarting the ssh service.

- sudo nano /etc/ssh/sshd_config
  - PermitRootLogin no

- sudo systemctl restart ssh





Step 2: Enforcing stronger password policy in /etc/security/pwquality.conf

- setting up password policy of setting:
  - minimum total password length = 16
  - Require atleast 1 digit
  - Require atleast 1 uppercase letter
  - Require atleast 1 special character
  - Require at least 1 lowercase letter

```
labuser@kali: ~
File  Actions  Edit  View  Help
  GNU nano 8.0                    /etc/security/pwquality.conf *
minlen = 16
#
# 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.
dcredit = -1
#
# 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.
ucredit = -1
#
# 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.
lcredit = -1
#
# 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.
ocredit = -1
lcredit = -1
# The minimum number of required classes of characters for the new
# password (digits, uppercase, lowercase, others).
```

Step 3: Implementing multifactor authentication:

- installing google authenticator
  - sudo apt install libpam-google-authenticator
- Run: google-authenticator
  - This will generate a QR code, on scanning it will get Google Authenticator app on your phone

```
┌──(user1㉿kali)-[~]
└─$ sudo apt install libpam-google-authenticator
Installing:
  libpam-google-authenticator

Summary:
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 459
  Download size: 44.5 kB
  Space needed: 134 kB / 18.0 GB available

Get:1 http://kali.download/kali kali-rolling/main amd64 libpam-google-authenticator amd64 20191231-2.1 [44.5 kB]
Fetched 44.5 kB in 6s (7,227 B/s)
Selecting previously unselected package libpam-google-authenticator.
(Reading database ... 412166 files and directories currently installed.)
Preparing to unpack .../libpam-google-authenticator_20191231-2.1_amd64.deb ...
Unpacking libpam-google-authenticator (20191231-2.1) ...
Setting up libpam-google-authenticator (20191231-2.1) ...
Processing triggers for kali-menu (2025.2.7) ...
Processing triggers for man-db (2.13.1-1) ...

┌──(user1㉿kali)-[~]
└─$ auth required pam_google_authenticator.so
Command 'auth' not found, did you mean:
  command 'oauth' from deb ruby-oauth
  command 'iauth' from deb ircd-irc2
  command 'xauth' from deb xauth
Try: sudo apt install <deb name>

┌──(user1㉿kali)-[~]
└─$ google-authenticator

Do you want authentication tokens to be time-based (y/n) y
Warning: pasting the following URL into your browser exposes the OTP secret to Google:
```

Scannig the above QR code will open an google authenticator app on mobile, and adds current user account to app. and promps a 6 digit OTP valid for 30 secs, which is used for login.

Step 4: enable PAM in /etc/pam.d/ssh

- auth required pam_google_authenticator.so
- Doing this when a user logs in via SSH, after entering their password, the system will also require a valid TOTP code from the user's Google Authenticator app.
- placement: above @include common-auth

```
user1@kali: ~
File  Actions  Edit  View  Help
  GNU nano 8.4                    /etc/pam.d/sshd
# PAM configuration for the Secure Shell service

auth required pam_google_authenticator.so

# Standard Un*x authentication.
@include common-auth
```

Screenshot taken

View image

Step 5: Edit /etc/ssh/sshd_config file with 'ChallengeResponseAuthentication yes'

- It allows SSH to to asks user for OTP. Without this, SSH won't ask for the Google Authenticator code at all, even if PAM is configured.

```
File  Actions  Edit  View  Help
  GNU nano 8.4                                  /etc/ssh/sshd_config
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to "no" here!
#PasswordAuthentication yes
#PermitEmptyPasswords no
ChallengeResponseAuthentication yes

# Change to "yes" to enable keyboard-interactive authentication.  Depending on
# the system's configuration, this may involve passwords, challenge-response,
# one-time passwords or some combination of these and other methods.
```

  - Restart ssh service:
    - sudo systemctl restart ssh

Step 6: Changing the ssh port from 22 (default) to 2222, in sshd_config file and then restarting the service

  - sudo nano /etc/ssh/sshd_config
  - sudo systemctl restart ssh

```
labuser@kali: ~
File  Actions  Edit  View  Help
  GNU nano 8.0                          /etc/ssh/sshd_config
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/local/bin:/usr/bin:/bin:/usr/games

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

Port 2222
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

                              [ Wrote 122 lines ]
^G Help      ^O Write Out   ^F Where Is    ^K Cut       ^T Execute     ^C Location
^X Exit      ^R Read File   ^\ Replace     ^U Paste     ^J Justify     ^/ Go To Line
```

```
┌──(labuser㉿kali)-[~]
└─$ sudo nano /etc/ssh/sshd_config

┌──(labuser㉿kali)-[~]
└─$ sudo systemctl restart ssh

┌──(labuser㉿kali)-[~]
└─$
```

Task 5: Continuous Monitoring and Reporting: Set up continuous monitoring to detect unusual access patterns and generate regular reports to inform the security team of any new threats.

Step 1:

- For continous monitoring we could install 'auditd' tool
  - sudo apt install auditd
  - sudo systemctl enable --now auditd
- can check for failed logins using:
  - ausearch -m USER_LOGIN --success no

```
┌──(user1㉿kali)-[~]
└─$ sudo apt install auditd
[sudo] password for user1:
auditd is already the newest version (1:4.0.2-2+b2).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 459

┌──(user1㉿kali)-[~]
└─$ sudo systemctl enable --now auditd
Synchronizing state of auditd.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable auditd

┌──(user1㉿kali)-[~]
└─$ sudo systemctl status auditd
● auditd.service - Security Audit Logging Service
     Loaded: loaded (/usr/lib/systemd/system/auditd.service; enabled; preset: disabled)
     Active: active (running) since Mon 2025-08-11 13:41:21 IST; 1min 43s ago
 Invocation: 51858a934c5e4400b26c1c657dbd57ae
       Docs: man:auditd(8)
             https://github.com/linux-audit/audit-documentation
   Main PID: 650 (auditd)
      Tasks: 2 (limit: 4501)
     Memory: 924K (peak: 1.9M)
        CPU: 40ms
     CGroup: /system.slice/auditd.service
             └─650 /usr/sbin/auditd

Aug 11 13:41:21 kali systemd[1]: Starting auditd.service - Security Audit Logging Service...
Aug 11 13:41:21 kali auditd[650]: No plugins found, not dispatching events
Aug 11 13:41:21 kali auditd[650]: Init complete, auditd 4.0.2 listening for events (startup state enable)
Aug 11 13:41:21 kali systemd[1]: Started auditd.service - Security Audit Logging Service.

┌──(user1㉿kali)-[~]
└─$ ausearch -m USER_LOGIN --success no
Error opening config file (Permission denied)
NOTE - using built-in end_of_event_timeout: 2
NOTE - using built-in logs: /var/log/audit/audit.log
<no matches>

┌──(user1㉿kali)-[~]
└─$ sudo ausearch -m USER_LOGIN --success no
<no matches>
```

Here, 'ausearch -m USER_LOGIN --success no' searches audit logs on a Linux system using the audit framework, filters only USER_LOGIN events (login attempts) and shows only events where the login was unsuccessful (--success no).