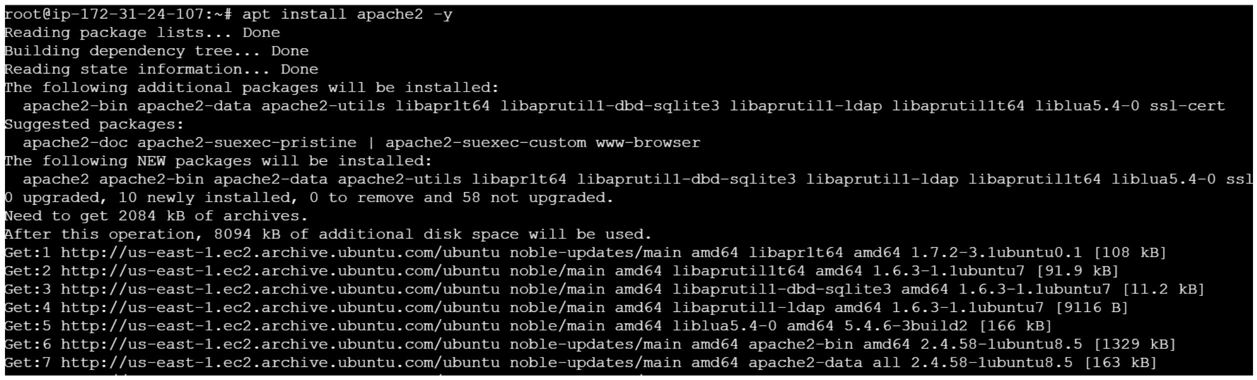
Login_failed_attempts

Create a Python script to:
- Parse logs from a web server.
- Identify IPs causing the most failed login attempts.
- Block those IPs by dynamically updating firewall rules.

1. Install Apache Web Server

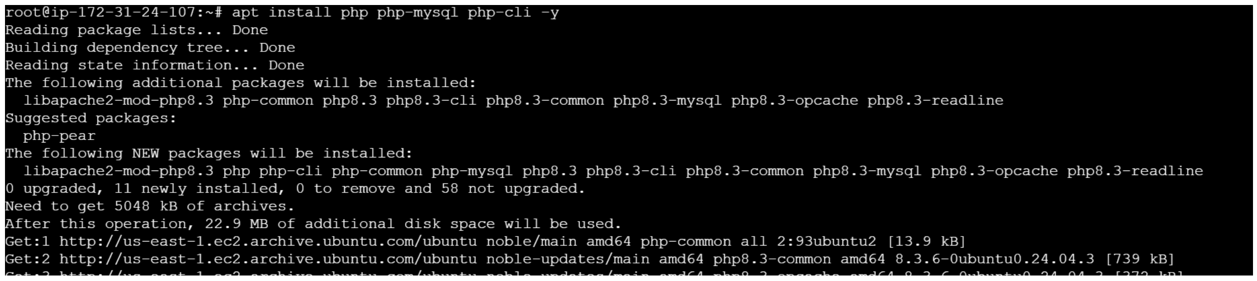
```
sudo apt update
sudo apt install apache2 -y
sudo systemctl start apache2
sudo systemctl enable apache2
```



```
root@ip-172-31-24-107:~# apt install apache2 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-sqlite3 libaprutil1-ldap libaprutil1t64 liblua5.4-0 ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-sqlite3 libaprutil1-ldap libaprutil1t64 liblua5.4-0 ssl
0 upgraded, 10 newly installed, 0 to remove and 58 not upgraded.
Need to get 2084 kB of archives.
After this operation, 8094 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libapr1t64 amd64 1.7.2-3.1ubuntu0.1 [108 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1t64 amd64 1.6.3-1.1ubuntu7 [91.9 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1-dbd-sqlite3 amd64 1.6.3-1.1ubuntu7 [11.2 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libaprutil1-ldap amd64 1.6.3-1.1ubuntu7 [9116 B]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 liblua5.4-0 amd64 5.4.6-3build2 [166 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-bin amd64 2.4.58-1ubuntu8.5 [1329 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 apache2-data all 2.4.58-1ubuntu8.5 [163 kB]
```

2. Install Prerequisites for WordPress

```
sudo apt install php php-mysql php-cli -y
```



```
root@ip-172-31-24-107:~# apt install php php-mysql php-cli -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libapache2-mod-php8.3 php-common php8.3 php8.3-cli php8.3-common php8.3-mysql php8.3-opcache php8.3-readline
Suggested packages:
  php-pear
The following NEW packages will be installed:
  libapache2-mod-php8.3 php php-cli php-common php-mysql php8.3 php8.3-cli php8.3-common php8.3-mysql php8.3-opcache php8.3-readline
0 upgraded, 11 newly installed, 0 to remove and 58 not upgraded.
Need to get 5048 kB of archives.
After this operation, 22.9 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 php-common all 2:93ubuntu2 [13.9 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 php8.3-common amd64 8.3.6-0ubuntu0.24.04.3 [739 kB]
```

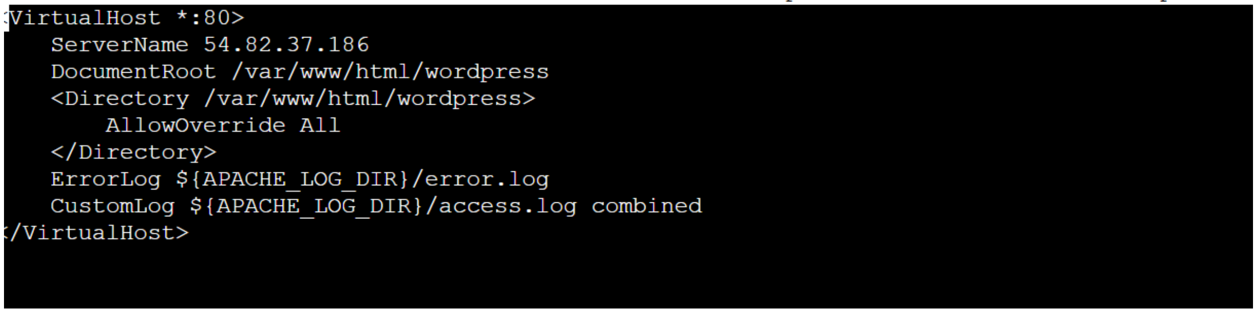
```
# Install MySQL
sudo apt install mysql-server -y
```

sudo systemctl start mysql
sudo systemctl enable mysql

3. Configure Apache for WordPress

Create a virtual host configuration file for your WordPress site.
Commands:

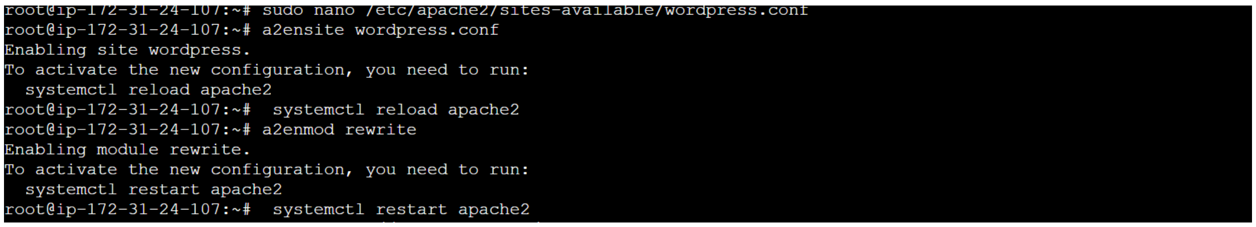
sudo nano /etc/apache2/sites-available/wordpress.conf



```
<VirtualHost *:80>
    ServerName 54.82.37.186
    DocumentRoot /var/www/html/wordpress
    <Directory /var/www/html/wordpress>
        AllowOverride All
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Enable the site and necessary modules:

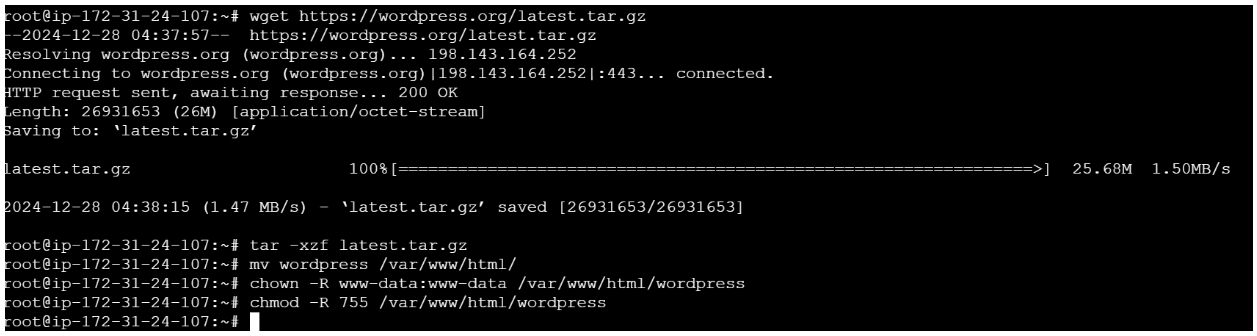
sudo a2ensite wordpress.conf
sudo a2enmod rewrite
sudo systemctl restart apache2



```
root@ip-172-31-24-107:~# sudo nano /etc/apache2/sites-available/wordpress.conf
root@ip-172-31-24-107:~# a2ensite wordpress.conf
Enabling site wordpress.
To activate the new configuration, you need to run:
  systemctl reload apache2
root@ip-172-31-24-107:~#  systemctl reload apache2
root@ip-172-31-24-107:~# a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
  systemctl restart apache2
root@ip-172-31-24-107:~#  systemctl restart apache2
```

4. Download and Set Up WordPress

wget https://wordpress.org/latest.tar.gz
tar -xzf latest.tar.gz
sudo mv wordpress /var/www/html/
sudo chown -R www-data:www-data /var/www/html/wordpress
sudo chmod -R 755 /var/www/html/wordpress

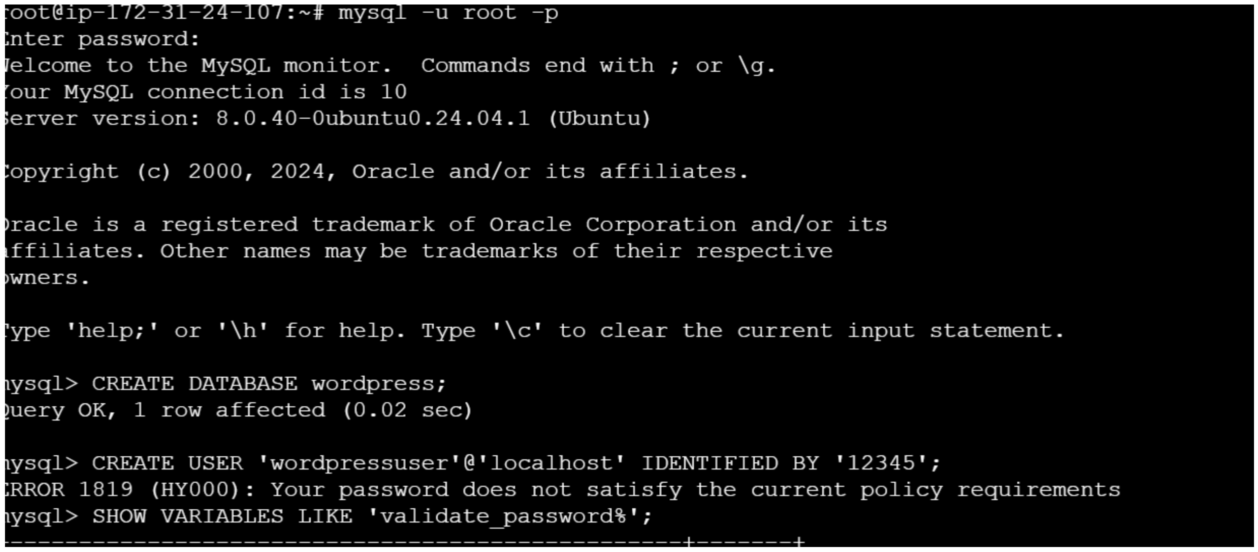


```
root@ip-172-31-24-107:~# wget https://wordpress.org/latest.tar.gz
--2024-12-28 04:37:57--  https://wordpress.org/latest.tar.gz
Resolving wordpress.org (wordpress.org)... 198.143.164.252
Connecting to wordpress.org (wordpress.org)|198.143.164.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26931653 (26M) [application/octet-stream]
Saving to: ‘latest.tar.gz’

latest.tar.gz                  100%[==============================================================>]  25.68M  1.50MB/s

2024-12-28 04:38:15 (1.47 MB/s) - ‘latest.tar.gz’ saved [26931653/26931653]

root@ip-172-31-24-107:~# tar -xzf latest.tar.gz
root@ip-172-31-24-107:~# mv wordpress /var/www/html/
root@ip-172-31-24-107:~# chown -R www-data:www-data /var/www/html/wordpress
root@ip-172-31-24-107:~# chmod -R 755 /var/www/html/wordpress
root@ip-172-31-24-107:~#
```

5. Create a Database for WordPress

Commands:

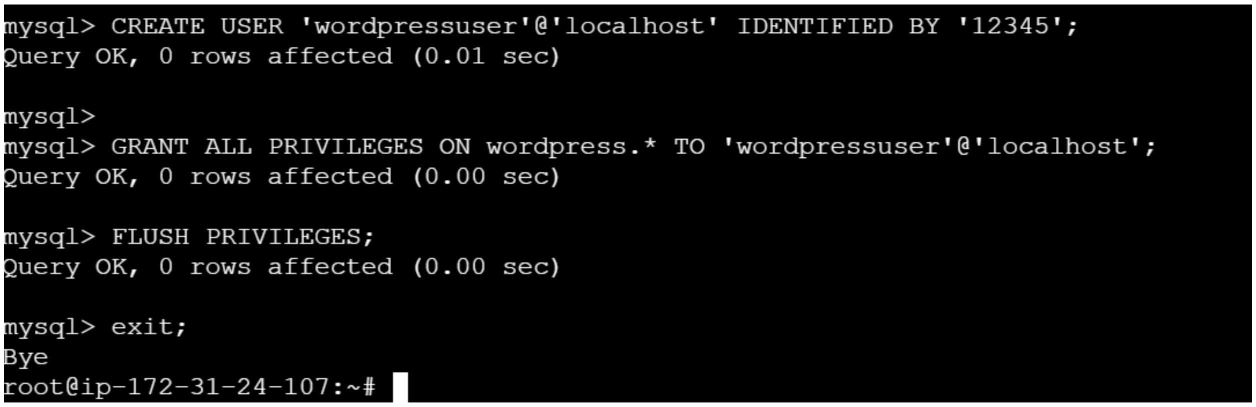
sudo mysql -u root -p



```
oot@ip-172-31-24-107:~# mysql -u root -p
nter password:
elcome to the MySQL monitor.  Commands end with ; or \g.
our MySQL connection id is 10
erver version: 8.0.40-0ubuntu0.24.04.1 (Ubuntu)

opyright (c) 2000, 2024, Oracle and/or its affiliates.

racle is a registered trademark of Oracle Corporation and/or its
ffiliates. Other names may be trademarks of their respective
wners.

ype 'help;' or '\h' for help. Type '\c' to clear the current input statement.

ysql> CREATE DATABASE wordpress;
uery OK, 1 row affected (0.02 sec)

ysql> CREATE USER 'wordpressuser'@'localhost' IDENTIFIED BY '12345';
RROR 1819 (HY000): Your password does not satisfy the current policy requirements
ysql> SHOW VARIABLES LIKE 'validate_password%';
-------------------------------------------------+--------+
```

SQL Commands:

CREATE DATABASE wordpress;
CREATE USER 'wordpressuser'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON wordpress.* TO
'wordpressuser'@'localhost';
FLUSH PRIVILEGES;
EXIT;



```
mysql> CREATE USER 'wordpressuser'@'localhost' IDENTIFIED BY '12345';
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpressuser'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> exit;
Bye
root@ip-172-31-24-107:~#
```

If browse http://ip-address/wp-login.php

6. Modify the Python Script for Apache Logs

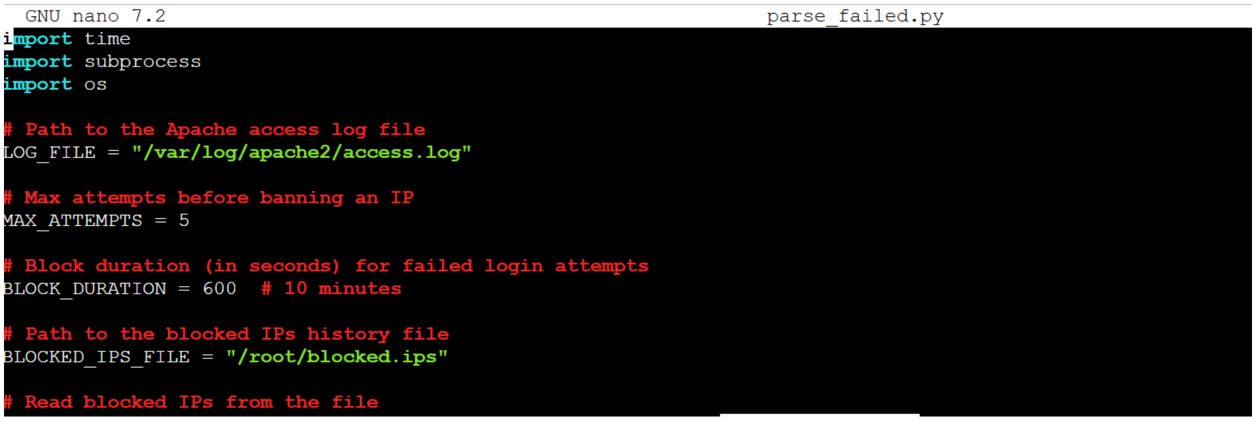
Adapt the provided Python script to monitor Apache logs (/var/log/apache2/access.log).

Python Script Overview:

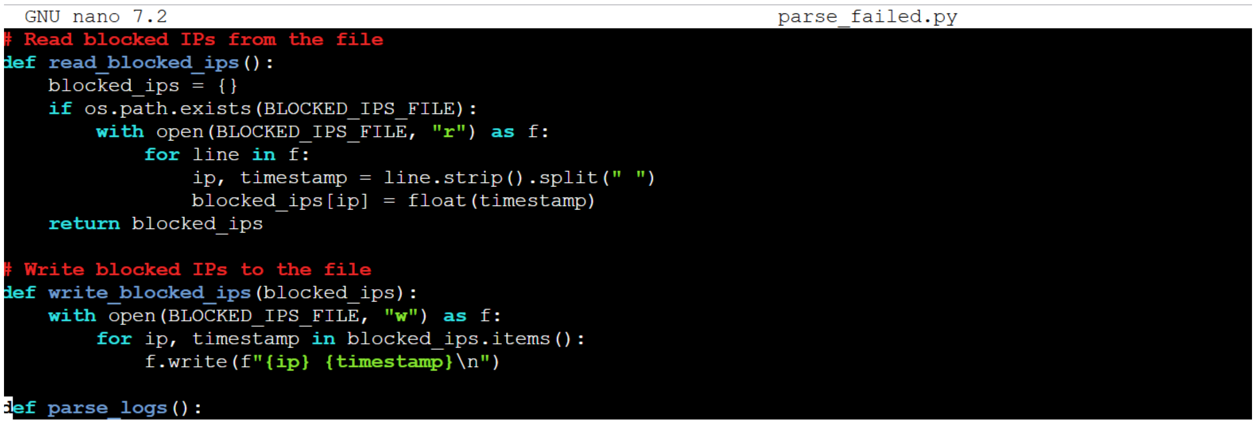
- Parse logs from Apache.
- Identify IPs with excessive failed login attempts.
- Dynamically block these IPs using iptables.

Create file for python script

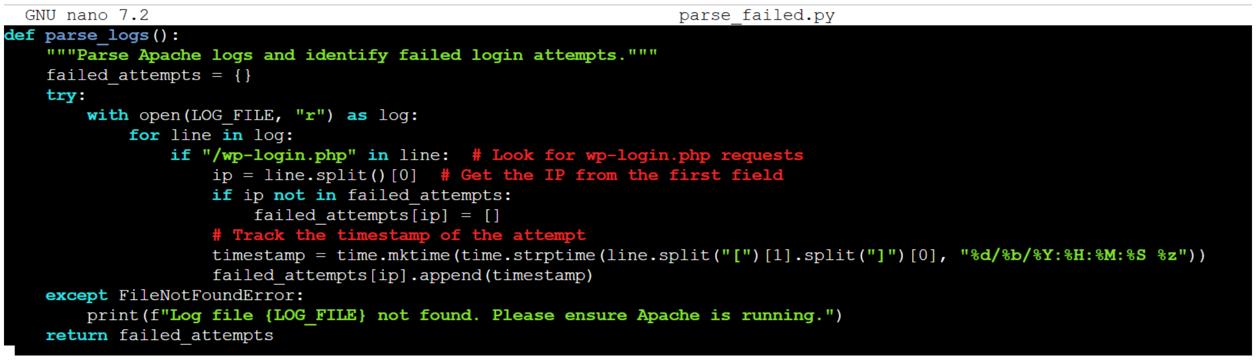
sudo nano parse_failed.py



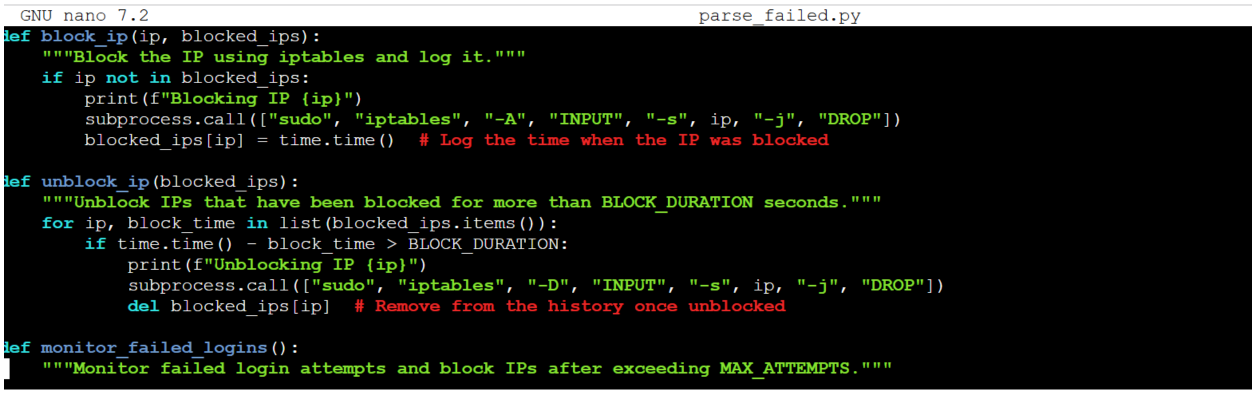
```
  GNU nano 7.2                                                    parse_failed.py
import time
import subprocess
import os

# Path to the Apache access log file
LOG_FILE = "/var/log/apache2/access.log"

# Max attempts before banning an IP
MAX_ATTEMPTS = 5

# Block duration (in seconds) for failed login attempts
BLOCK_DURATION = 600   # 10 minutes

# Path to the blocked IPs history file
BLOCKED_IPS_FILE = "/root/blocked.ips"

# Read blocked IPs from the file
```



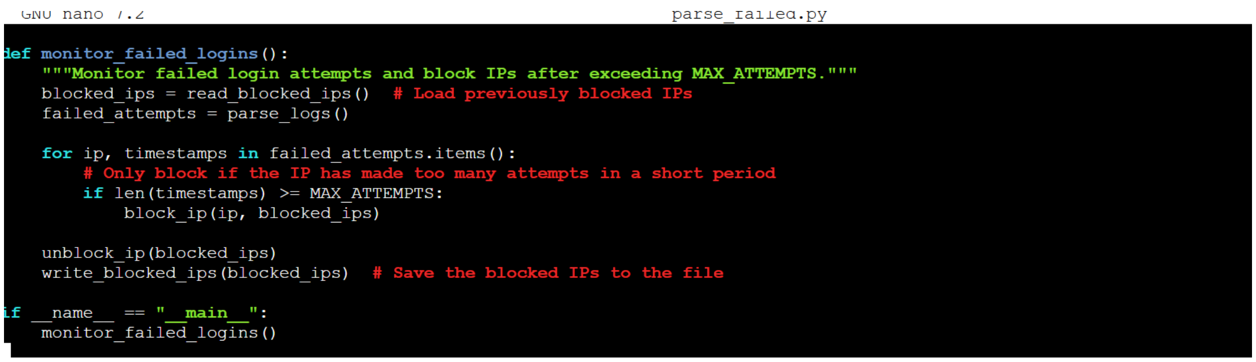
```
  GNU nano 7.2                                          parse_failed.py
# Read blocked IPs from the file
def read_blocked_ips():
    blocked_ips = {}
    if os.path.exists(BLOCKED_IPS_FILE):
        with open(BLOCKED_IPS_FILE, "r") as f:
            for line in f:
                ip, timestamp = line.strip().split(" ")
                blocked_ips[ip] = float(timestamp)
    return blocked_ips

# Write blocked IPs to the file
def write_blocked_ips(blocked_ips):
    with open(BLOCKED_IPS_FILE, "w") as f:
        for ip, timestamp in blocked_ips.items():
            f.write(f"{ip} {timestamp}\n")

def parse_logs():
```



```
  GNU nano 7.2                                          parse_failed.py
def parse_logs():
    """Parse Apache logs and identify failed login attempts."""
    failed_attempts = {}
    try:
        with open(LOG_FILE, "r") as log:
            for line in log:
                if "/wp-login.php" in line:  # Look for wp-login.php requests
                    ip = line.split()[0]  # Get the IP from the first field
                    if ip not in failed_attempts:
                        failed_attempts[ip] = []
                    # Track the timestamp of the attempt
                    timestamp = time.mktime(time.strptime(line.split("[")[1].split("]")[0], "%d/%b/%Y:%H:%M:%S %z"))
                    failed_attempts[ip].append(timestamp)
    except FileNotFoundError:
        print(f"Log file {LOG_FILE} not found. Please ensure Apache is running.")
    return failed_attempts
```



```
  GNU nano 7.2                                          parse_failed.py
def block_ip(ip, blocked_ips):
    """Block the IP using iptables and log it."""
    if ip not in blocked_ips:
        print(f"Blocking IP {ip}")
        subprocess.call(["sudo", "iptables", "-A", "INPUT", "-s", ip, "-j", "DROP"])
        blocked_ips[ip] = time.time()  # Log the time when the IP was blocked

def unblock_ip(blocked_ips):
    """Unblock IPs that have been blocked for more than BLOCK_DURATION seconds."""
    for ip, block_time in list(blocked_ips.items()):
        if time.time() - block_time > BLOCK_DURATION:
            print(f"Unblocking IP {ip}")
            subprocess.call(["sudo", "iptables", "-D", "INPUT", "-s", ip, "-j", "DROP"])
            del blocked_ips[ip]  # Remove from the history once unblocked

def monitor_failed_logins():
    """Monitor failed login attempts and block IPs after exceeding MAX_ATTEMPTS."""
```



```
  GNU nano 7.2                                          parse_failed.py

def monitor_failed_logins():
    """Monitor failed login attempts and block IPs after exceeding MAX_ATTEMPTS."""
    blocked_ips = read_blocked_ips()  # Load previously blocked IPs
    failed_attempts = parse_logs()

    for ip, timestamps in failed_attempts.items():
        # Only block if the IP has made too many attempts in a short period
        if len(timestamps) >= MAX_ATTEMPTS:
            block_ip(ip, blocked_ips)

    unblock_ip(blocked_ips)
    write_blocked_ips(blocked_ips)  # Save the blocked IPs to the file

if __name__ == "__main__":
    monitor_failed_logins()
```

Make Script executable

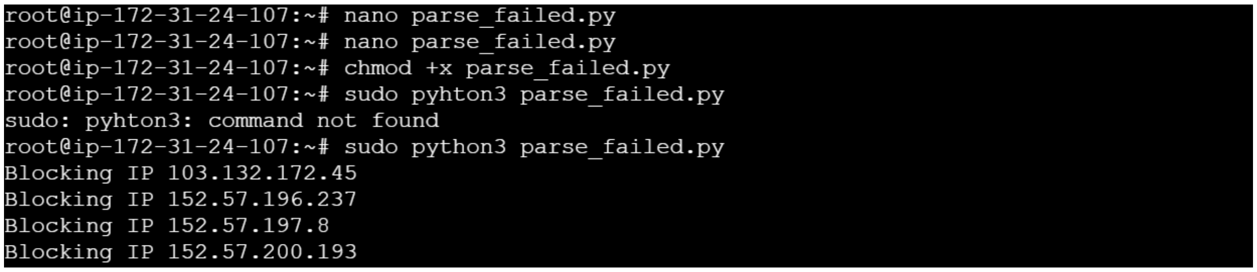
```
sudo chmod +x parse_failed.py
```

7. Test the Script

- Simulate failed login attempts more than 5 times to verify functionality.
- Monitor Apache logs using tail -f /var/log/apache2/access.log.
- Confirm blocked IPs using sudo iptables -L.

8. Run the Script

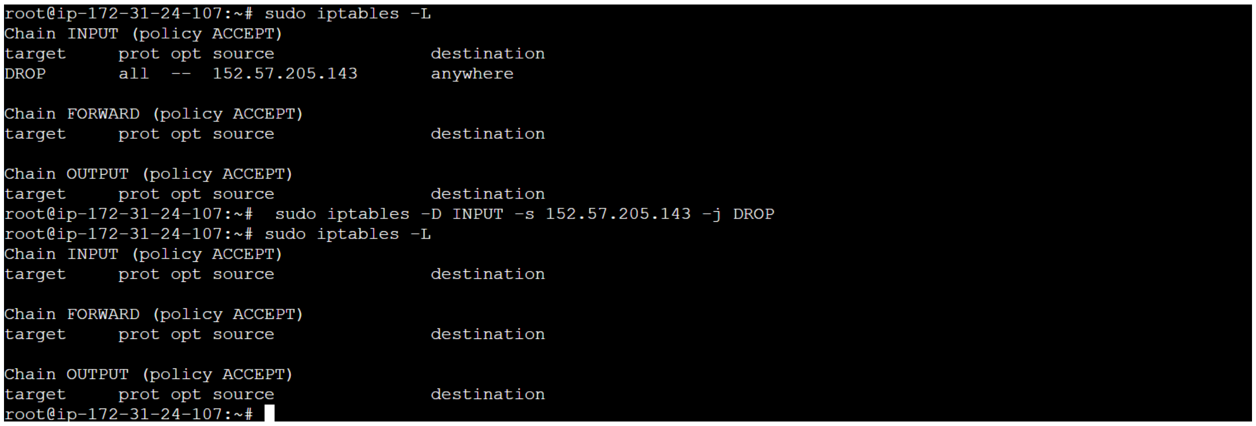
sudo python3 parse_failed.py



```
root@ip-172-31-24-107:~# nano parse_failed.py
root@ip-172-31-24-107:~# nano parse_failed.py
root@ip-172-31-24-107:~# chmod +x parse_failed.py
root@ip-172-31-24-107:~# sudo pyhton3 parse_failed.py
sudo: pyhton3: command not found
root@ip-172-31-24-107:~# sudo python3 parse_failed.py
Blocking IP 103.132.172.45
Blocking IP 152.57.196.237
Blocking IP 152.57.197.8
Blocking IP 152.57.200.193
```

9. For unblocked ip

sudo iptables -D INPUT -s <ip-address/> -j DROP



```
root@ip-172-31-24-107:~# sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  152.57.205.143       anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ip-172-31-24-107:~#  sudo iptables -D INPUT -s 152.57.205.143 -j DROP
root@ip-172-31-24-107:~# sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ip-172-31-24-107:~# 
```