

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.

Install nginx that acts as proxy server.

```

azureuser@validate: ~
azureuser@validate: $ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Fri 2024-12-27 05:13:36 UTC; 2min 18s ago
     Docs: man:nginx(8)
  Process: 904 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 920 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 927 (nginx)
    Tasks: 3 (limit: 9458)
   Memory: 3.8M (peak: 3.9M)
      CPU: 27ms
   CGroup: /system.slice/nginx.service
           └─927 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─928 "nginx: worker process"
               └─929 "nginx: worker process"

Dec 27 05:13:36 validate systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
Dec 27 05:13:36 validate systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
azureuser@validate: $

```

#Install the prerequisite for WordPress as PHP and MySQL.

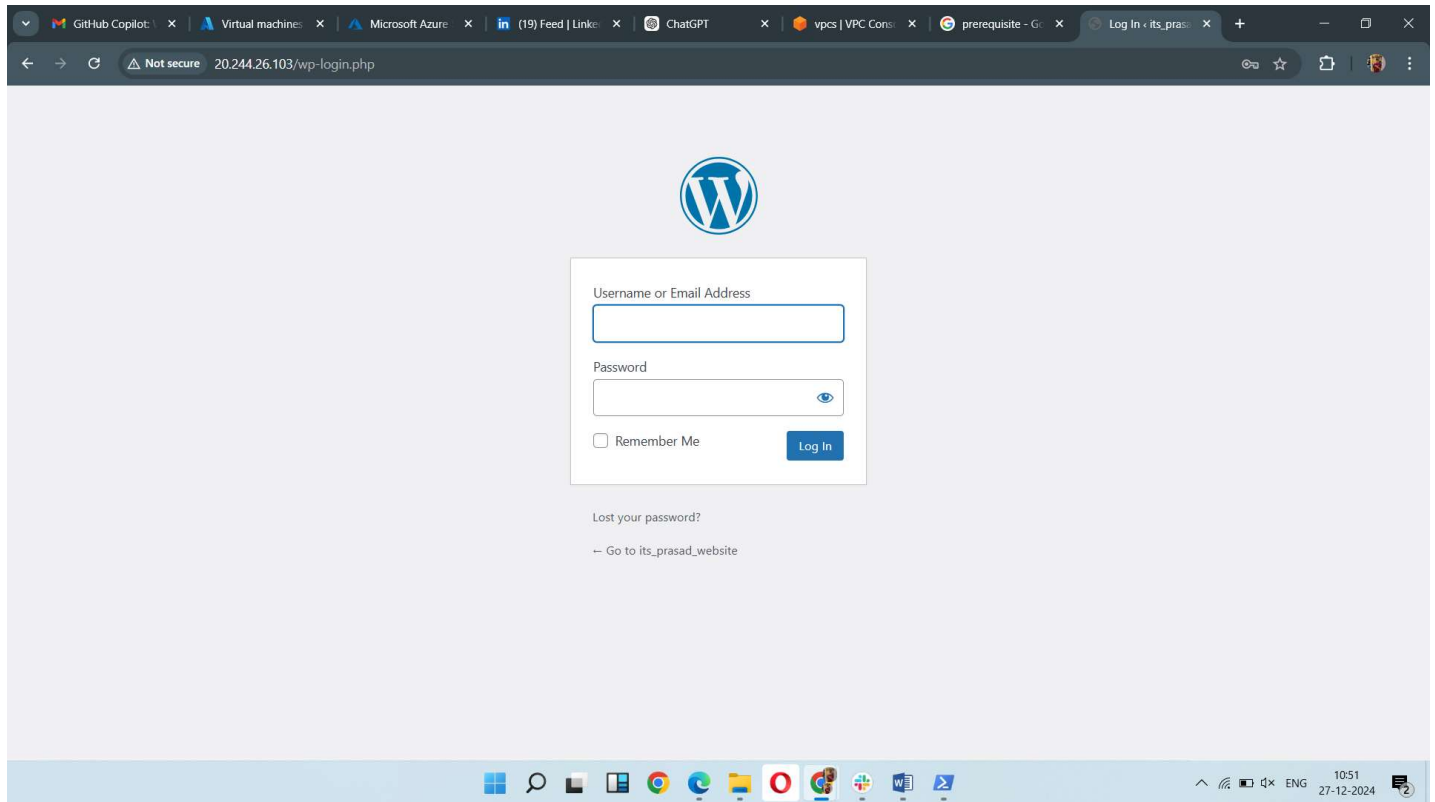
```

azureuser@validate: ~
azureuser@validate: $ systemctl status mysql
● mysql.service - MySQL Community Server
   Loaded: loaded (/usr/lib/systemd/system/mysql.service; enabled; preset: enabled)
   Active: active (running) since Fri 2024-12-27 05:13:39 UTC; 6min ago
   Process: 777 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
 Main PID: 1024 (mysqld)
    Status: "Server is operational"
   Tasks: 37 (limit: 9458)
  Memory: 448.8M (peak: 450.8M)
      CPU: 3.014s
   CGroup: /system.slice/mysql.service
           └─1024 /usr/sbin/mysqld

Dec 27 05:13:35 validate systemd[1]: Starting mysql.service - MySQL Community Server...
Dec 27 05:13:39 validate systemd[1]: Started mysql.service - MySQL Community Server.
azureuser@validate: $ php -v
PHP 8.0.30 (cli) (built: Dec 24 2024 07:19:59) ( NTS )
Copyright (c) The PHP Group
Zend Engine v4.0.30, Copyright (c) Zend Technologies
    with Zend OPcache v8.0.30, Copyright (c), by Zend Technologies
azureuser@validate: $

```

Installing wordpress



By checking the access log of the nginx I understood the HTTP status code

- 2XX:success
 - 200: OK (The request was success)
 - 201: Created (A new resource was successfully created)
 - 204: No Content (Request succeeded, but there's no content to send back)
- 4xx: Client Errors
 - 400: Bad Request (The request could not be understood)
 - 401: Unauthorized (Authentication is required)
 - 403: Forbidden (The server understood the request but refuses to authorize it)
 - 404: Not Found (The requested resource is not found)

```
azureuser@validate: ~  
azureuser@validate: $ sudo tail -f /var/log/nginx/access.log  
20.244.26.103 - - [26/Dec/2024:11:25:46 +0000] "POST /wp-cron.php?doing_wp_cron=1735212346.7472679615020751953125 HTTP/1.1" 499 0 "-" "WordPress/6.7.1; http://141.98.11.155 - - [26/Dec/2024:11:25:46 +0000] "GET / HTTP/1.1" 301 5 "-" "-"  
20.244.26.103 - - [27/Dec/2024:05:21:27 +0000] "POST /wp-cron.php?doing_wp_cron=1735276887.0670690536499023437500 HTTP/1.1" 499 0 "-" "WordPress/6.7.1; http://182.156.140.38 - - [27/Dec/2024:05:21:27 +0000] "GET / HTTP/1.1" 200 12014 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36"  
182.156.140.38 - - [27/Dec/2024:05:21:27 +0000] "GET /wp-includes/blocks/navigation/style.min.css?ver=6.7.1 HTTP/1.1" 200 16384 "http://20.244.26.103/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36"  
182.156.140.38 - - [27/Dec/2024:05:21:27 +0000] "GET /wp-content/themes/twentytwentyfive/style.css?ver=1.0 HTTP/1.1" 200 2503 "http://20.244.26.103/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36"  
182.156.140.38 - - [27/Dec/2024:05:21:27 +0000] "GET /wp-includes/js/dist/script-modules/block-library/navigation/view.min.js?ver=8ff192874fc8910a284c HTTP/1.1" 200 366 "http://20.244.26.103/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36"  
182.156.140.38 - - [27/Dec/2024:05:21:27 +0000] "GET /wp-includes/js/dist/script-modules/interactivity/index.min.js?ver=06b8f695ef48ab2d9277 HTTP/1.1" 200 366 "http://20.244.26.103/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36"  
182.156.140.38 - - [27/Dec/2024:05:21:27 +0000] "GET /wp-content/themes/twentytwentyfive/assets/fonts/manrope/Manrope-VariableFont_wght.woff2 HTTP/1.1" 200 53 "http://20.244.26.103/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36"  
182.156.140.38 - - [27/Dec/2024:05:21:35 +0000] "GET /wp-login.php HTTP/1.1" 200 1938 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36"
```

Python script that :

Accessed the nginx logs

Created the blocked_ip.txt file if not there and use to store the history of blocked IP addresses.

Max attempted per user is set as 5 & Max duration that IP will Remain blocked is 5 minutes.

Blocking the User IP using iptables and stored it blocked_ip.txt file with date and time.

Use tail -f for continuously monitoring the access.log file

```
GNU nano 7.2 block_ip_failed_logins.py
import subprocess
import time
import re
from datetime import datetime

# Configuration
LOG_FILE = "/var/log/nginx/access.log"
BLOCKED_IPS_FILE = "/etc/nginx/sites-available/blocked_ips.txt"
MAX_FAILED_ATTEMPTS = 5
BLOCK_DURATION = 300 # in seconds
BLOCKED_IPS = set()

def load_blocked_ips():
    """
    Load already blocked IPs from the file to avoid redundant blocking.
    """
    try:
        with open(BLOCKED_IPS_FILE, "r") as file:
            for line in file:
                # Extract IP from the line
                parts = line.strip().split(" ")
                if parts:
                    ip = parts[0]
                    BLOCKED_IPS.add(ip)
    except FileNotFoundError:
        print(f"{BLOCKED_IPS_FILE} not found. Creating a new one...")
        open(BLOCKED_IPS_FILE, "w").close()

def block_ip(ip_address):
    """
    Block the given IP using iptables and add it to the blocked IPs file with timestamp.
    """
    if ip_address in BLOCKED_IPS:
        print(f"IP {ip_address} is already blocked.")
        return

    try:
        # Block IP via iptables
        subprocess.run(["sudo", "iptables", "-A", "INPUT", "-s", ip_address, "-j", "DROP"], check=True)
        print(f"IP {ip_address} has been blocked.")

        # Add to blocked IPs list and write to file with timestamp
        BLOCKED_IPS.add(ip_address)
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        with open(BLOCKED_IPS_FILE, "a") as file:
            file.write(f"{ip_address} - Blocked at: {timestamp}\n")

        # Schedule unblock
        time.sleep(BLOCK_DURATION)
        unblock_ip(ip_address)
    except subprocess.CalledProcessError as e:
        print(f"Error blocking IP {ip_address}: {e}")

def unblock_ip(ip_address):
    """
    Unblock the given IP after the block duration expires.
    """
    try:
        subprocess.run(["sudo", "iptables", "-D", "INPUT", "-s", ip_address, "-j", "DROP"], check=True)
        print(f"IP {ip_address} has been unblocked.")
        BLOCKED_IPS.remove(ip_address)

        # Remove the IP from the file
        with open(BLOCKED_IPS_FILE, "r") as file:
            lines = file.readlines()
        with open(BLOCKED_IPS_FILE, "w") as file:
            for line in lines:
                if not line.startswith(ip_address):
                    file.write(line)
    except subprocess.CalledProcessError as e:
        print(f"Error unblocking IP {ip_address}: {e}")

def main():
    """
    Continuously monitor the access log for failed login attempts.
    """
    failed_attempts = {}

    # Open log file with tail
    with subprocess.Popen(['tail', '-n', '0', '-F', LOG_FILE], stdout=subprocess.PIPE) as log_file:
        for line in log_file.stdout:
            line = line.decode('utf-8').strip()

            # Match POST requests to /wp-login.php
            if 'POST /wp-login.php' in line:
                parts = line.split()
                ip_address = parts[0]

                # Track failed attempts per IP
                if ip_address not in failed_attempts:
                    failed_attempts[ip_address] = 0
                failed_attempts[ip_address] += 1

                print(f"IP {ip_address} has {failed_attempts[ip_address]} failed attempts.")

                # Block IP if it exceeds the threshold
                if failed_attempts[ip_address] >= MAX_FAILED_ATTEMPTS:
                    print(f"Blocking IP {ip_address} for {BLOCK_DURATION} seconds due to repeated failed logins.")
                    block_ip(ip_address)
                    failed_attempts[ip_address] = 0 # Reset the counter
```

```
GNU nano 7.2 block_ip_failed_logins.py
print(f"Error blocking IP {ip_address}: {e}")

def unblock_ip(ip_address):
    """
    Unblock the given IP after the block duration expires.
    """
    try:
        subprocess.run(["sudo", "iptables", "-D", "INPUT", "-s", ip_address, "-j", "DROP"], check=True)
        print(f"IP {ip_address} has been unblocked.")
        BLOCKED_IPS.remove(ip_address)

        # Remove the IP from the file
        with open(BLOCKED_IPS_FILE, "r") as file:
            lines = file.readlines()
        with open(BLOCKED_IPS_FILE, "w") as file:
            for line in lines:
                if not line.startswith(ip_address):
                    file.write(line)
    except subprocess.CalledProcessError as e:
        print(f"Error unblocking IP {ip_address}: {e}")

def main():
    """
    Continuously monitor the access log for failed login attempts.
    """
    failed_attempts = {}

    # Open log file with tail
    with subprocess.Popen(['tail', '-n', '0', '-F', LOG_FILE], stdout=subprocess.PIPE) as log_file:
        for line in log_file.stdout:
            line = line.decode('utf-8').strip()

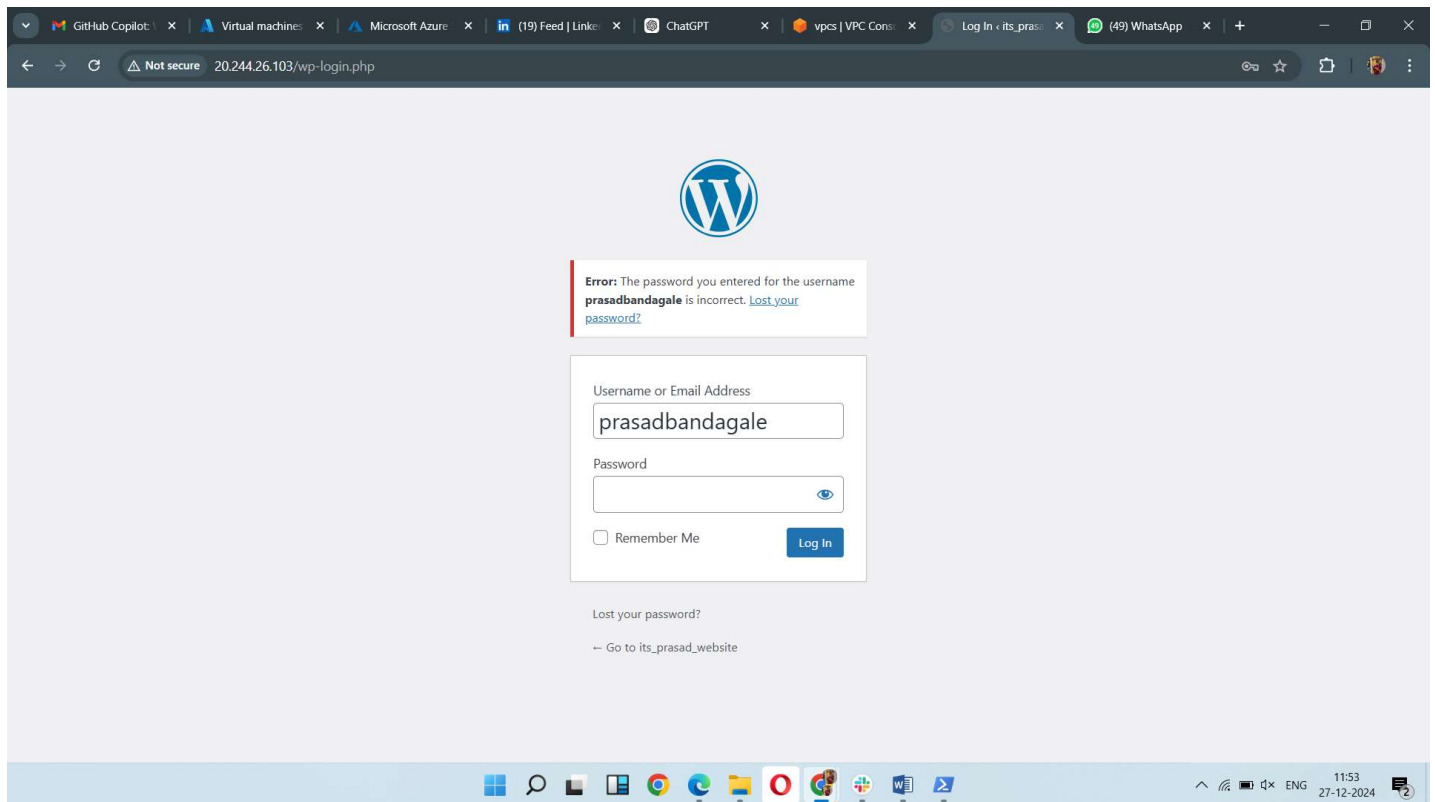
            # Match POST requests to /wp-login.php
            if 'POST /wp-login.php' in line:
                parts = line.split()
                ip_address = parts[0]

                # Track failed attempts per IP
                if ip_address not in failed_attempts:
                    failed_attempts[ip_address] = 0
                failed_attempts[ip_address] += 1

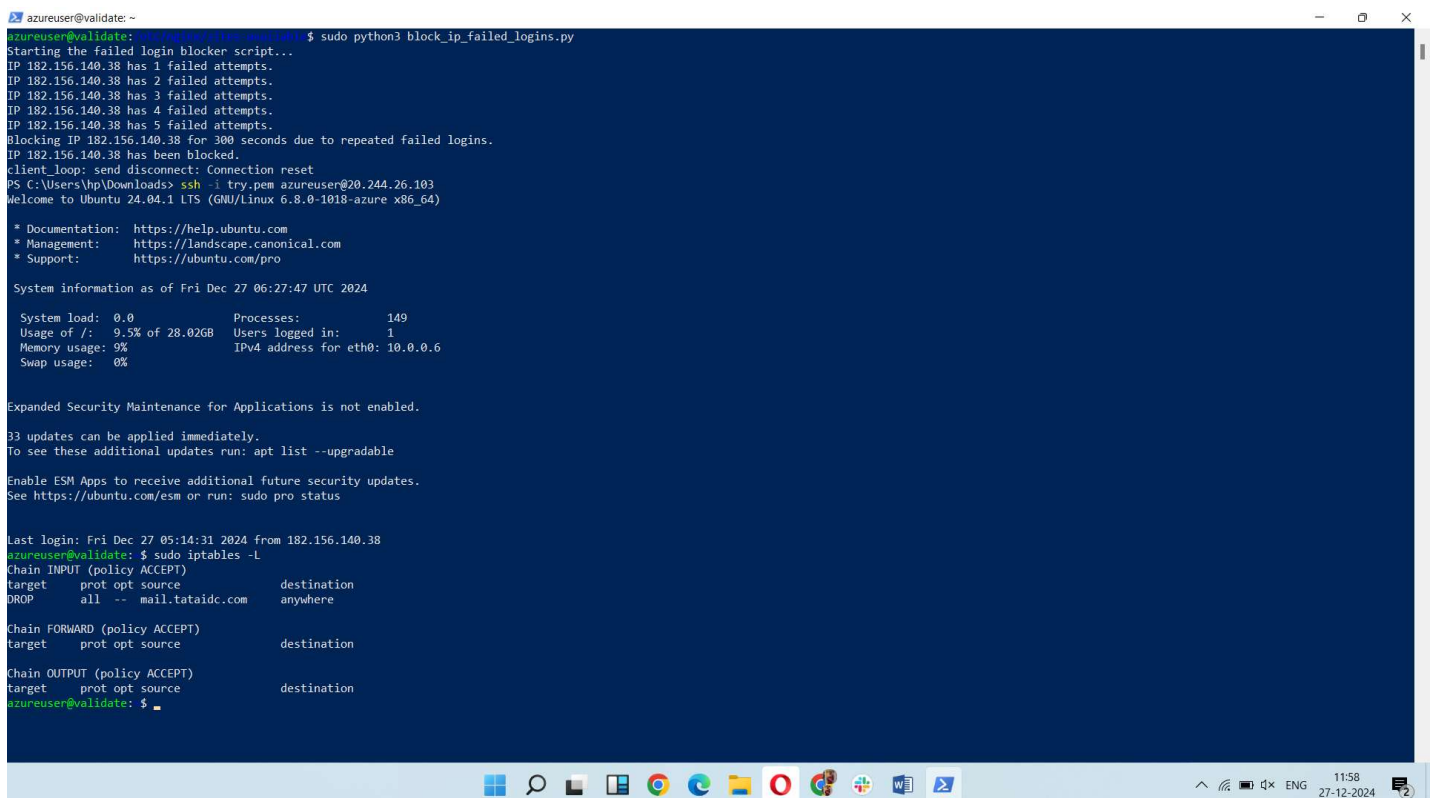
                print(f"IP {ip_address} has {failed_attempts[ip_address]} failed attempts.")

                # Block IP if it exceeds the threshold
                if failed_attempts[ip_address] >= MAX_FAILED_ATTEMPTS:
                    print(f"Blocking IP {ip_address} for {BLOCK_DURATION} seconds due to repeated failed logins.")
                    block_ip(ip_address)
                    failed_attempts[ip_address] = 0 # Reset the counter
```

Intentionally attempted to log in 5 times with incorrect credentials to check how the script would react.



The script blocked the IP after 5 failed login attempts, as verified in iptables. Since the IP was blocked, SSH access to the server using the same IP was not possible. Therefore, the network was changed, and the server was accessed again.



The Script also stored the blocked IP address with Date and time in the Blocked_ips.txt file

```
azureuser@validate: /etc/nginx/sites-available
Chain FORWARD (policy ACCEPT)
target      prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target      prot opt source                destination
azureuser@validate: $ cd /etc/nginx/sites-available/
azureuser@validate: /etc/nginx/sites-available$ ll
total 24
drwxr-xr-x 2 root root 4096 Dec 27 06:11 /
drwxr-xr-x 8 root root 4096 Dec 26 08:58 /
-rw-r--r-- 1 root root 3725 Dec 27 06:11 block_ip_failed_logins.py
-rw-r--r-- 1 root root   49 Dec 27 06:26 blocked_ips.txt
-rw-r--r-- 1 root root 2986 Dec 26 09:13 default
-rw----- 1 root root  324 Dec 26 09:40 nohup.out
azureuser@validate: /etc/nginx/sites-available$ cat blocked_ips.txt
182.156.140.38 - Blocked at: 2024-12-27 06:26:31
azureuser@validate: /etc/nginx/sites-available$
```

After 5 minutes, the IP address blocked by the script is automatically unblocked.

```
azureuser@validate: /etc/nginx/sites-available
368 clear
369 tail blocked_ips.txt
370 sudo iptables -L -v -n
371 rm blocked_ips.txt
372 sudo rm blocked_ips.txt
373 sudo python3 block_ip_failed_logins.py
374 sudo iptables -L
375 sudo iptables -D INPUT 1
376 sudo iptables -L
377 clear
378 sudo iptables -L
379 cd /etc/nginx/sites-available/
380 ll
381 cat blocked_ips.txt
382 clear
383 sudo iptables -L
384 clear
385 history
azureuser@validate: /etc/nginx/sites-available$ 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
azureuser@validate: /etc/nginx/sites-available$
```