“Expert Cloud Consulting”

SOP | Block Failed Login IPs for Grafana

20 Jun 2025

—

Contributed by: Akshata Ujawane
Approved by : Akshay (In Review)
Expert Cloud Consulting
Office #811, Gera Imperium Rise,
Hinjewadi Phase-II Rd, Pune, India – 411057

Block Failed Login IPs for Grafana

Expert Cloud Consulting

Enhance Optimise & Scale

ASCP GPUonCLOUD Pvt Ltd

Objective

Automatically detect and block IPs that attempt to log in to Grafana with incorrect credentials more than 3 times within 5 minutes. Blocked IPs will be unblocked automatically after 5 minutes

Step-by-Step: Install Grafana on Ubuntu

Step 1: Update System **sudo apt update &&**

sudo apt upgrade -y

```
sudo apt-get install -y apt-transport-https
sudo apt-get install -y software-properties-common wget
sudo wget -q -O /usr/share/keyrings/grafana.key https://apt.grafana.com/gpg.key
```

Stable release
```
echo "deb [signed-by=/usr/share/keyrings/grafana.key]
https://apt.grafana.com
stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
```

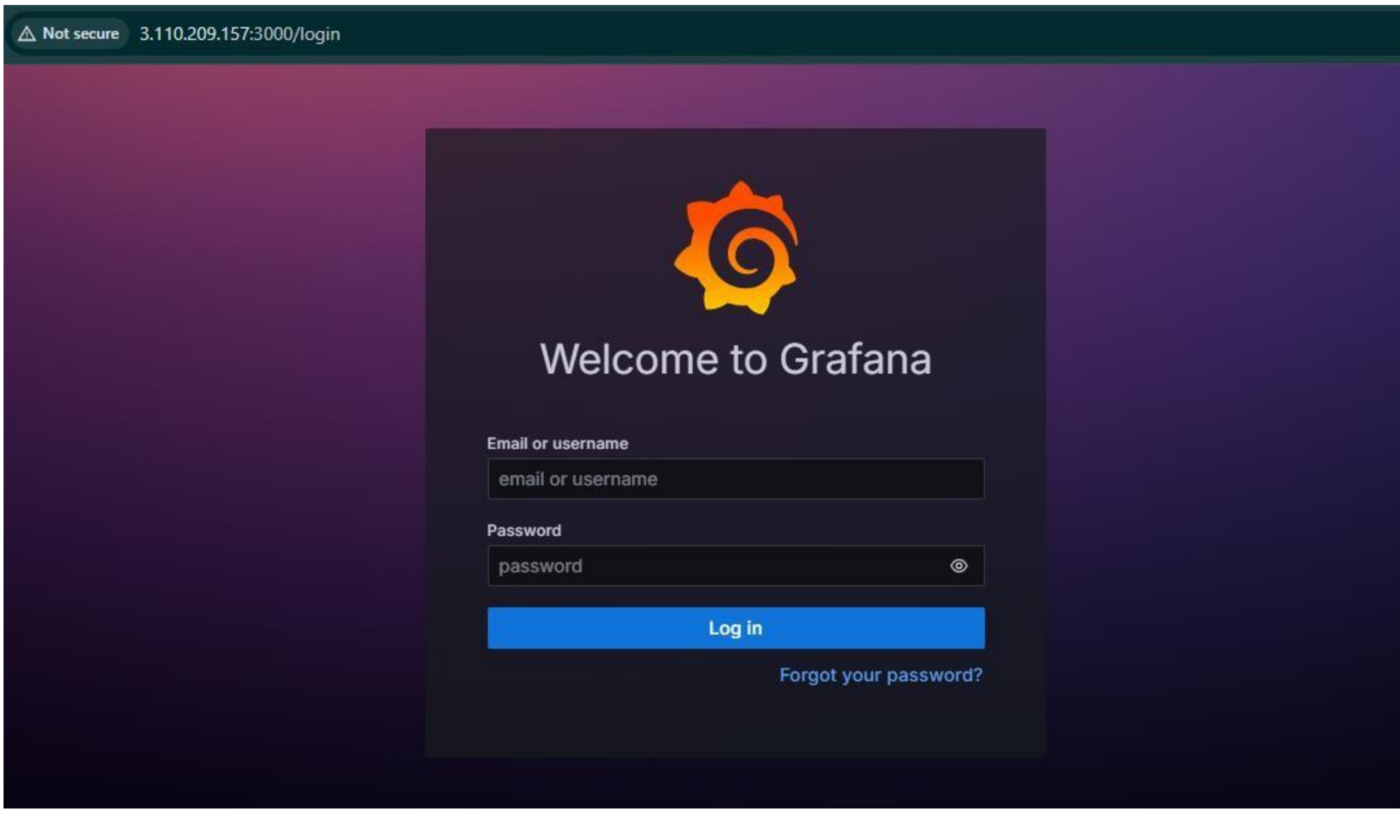
```
# Update the list of available packages
sudo apt-get update
```

```
# Install the latest OSS release:
sudo apt-get install grafana
#To start Grafana Server sudo

/bin/systemctl status grafana-server
```

Access Grafana

Open your browser and go to:

http://3.110.209.157:3000/login

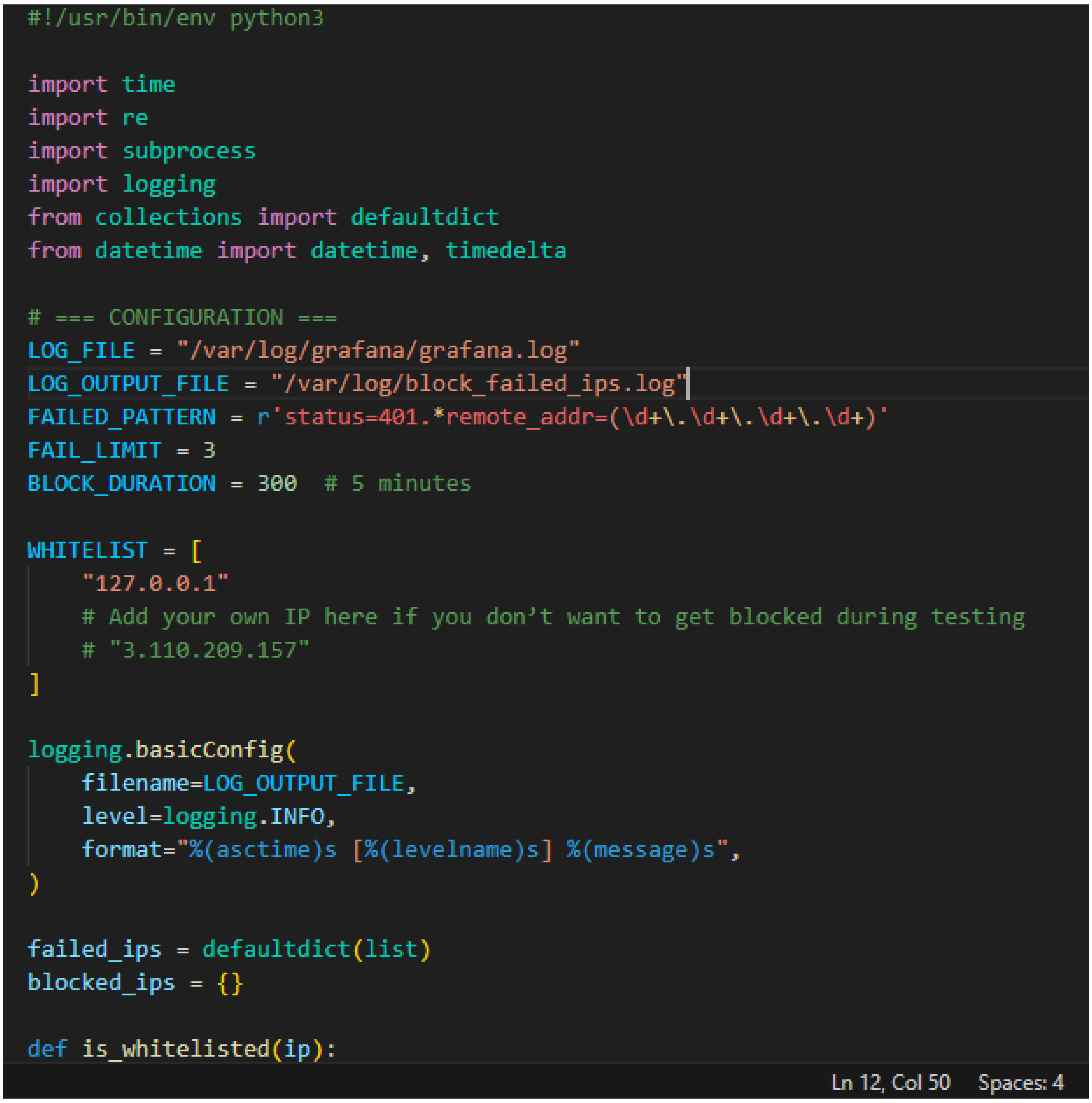


Step 1: Create the Python Script

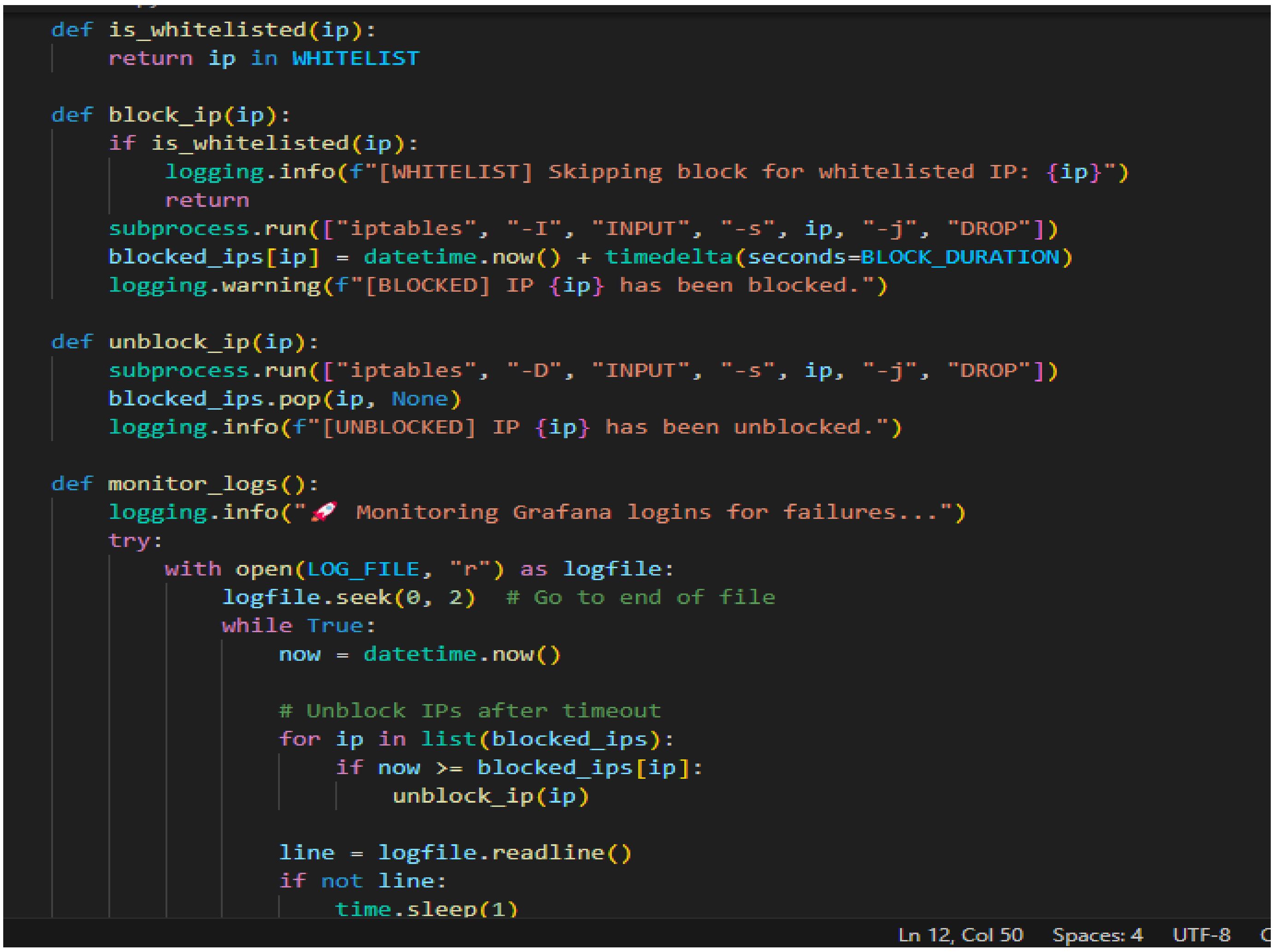
Createa directory to store the script:

sudo mkdir -p /opt/scripts

Create the Python script file:

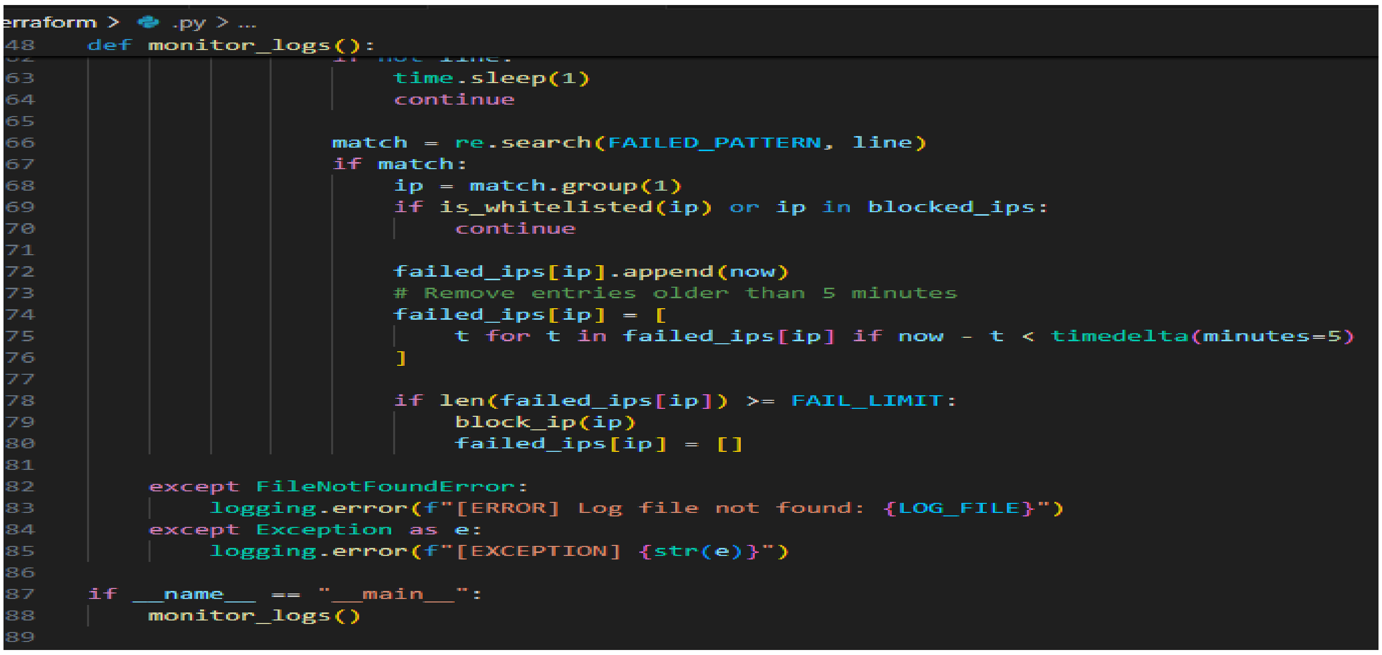


```
#!/usr/bin/env python3

import time
import re
import subprocess
import logging
from collections import defaultdict
from datetime import datetime, timedelta

# === CONFIGURATION ===
LOG_FILE = "/var/log/grafana/grafana.log"
LOG_OUTPUT_FILE = "/var/log/block_failed_ips.log"
FAILED_PATTERN = r'status=401.*remote_addr=(\d+\.\d+\.\d+\.\d+)'
FAIL_LIMIT = 3
BLOCK_DURATION = 300  # 5 minutes

WHITELIST = [
    "127.0.0.1"
    # Add your own IP here if you don’t want to get blocked during testing
    # "3.110.209.157"
]

logging.basicConfig(
    filename=LOG_OUTPUT_FILE,
    level=logging.INFO,
    format="%(asctime)s [%(levelname)s] %(message)s",
)

failed_ips = defaultdict(list)
blocked_ips = {}

def is_whitelisted(ip):
```



```
def is_whitelisted(ip):
    return ip in WHITELIST

def block_ip(ip):
    if is_whitelisted(ip):
        logging.info(f"[WHITELIST] Skipping block for whitelisted IP: {ip}")
        return
    subprocess.run(["iptables", "-I", "INPUT", "-s", ip, "-j", "DROP"])
    blocked_ips[ip] = datetime.now() + timedelta(seconds=BLOCK_DURATION)
    logging.warning(f"[BLOCKED] IP {ip} has been blocked.")

def unblock_ip(ip):
    subprocess.run(["iptables", "-D", "INPUT", "-s", ip, "-j", "DROP"])
    blocked_ips.pop(ip, None)
    logging.info(f"[UNBLOCKED] IP {ip} has been unblocked.")

def monitor_logs():
    logging.info("🚀 Monitoring Grafana logins for failures...")
    try:
        with open(LOG_FILE, "r") as logfile:
            logfile.seek(0, 2)  # Go to end of file
            while True:
                now = datetime.now()

                # Unblock IPs after timeout
                for ip in list(blocked_ips):
                    if now >= blocked_ips[ip]:
                        unblock_ip(ip)

                line = logfile.readline()
                if not line:
                    time.sleep(1)
```

Ln 12, Col 50   Spaces: 4   UTF-8



erraform > .py > ...

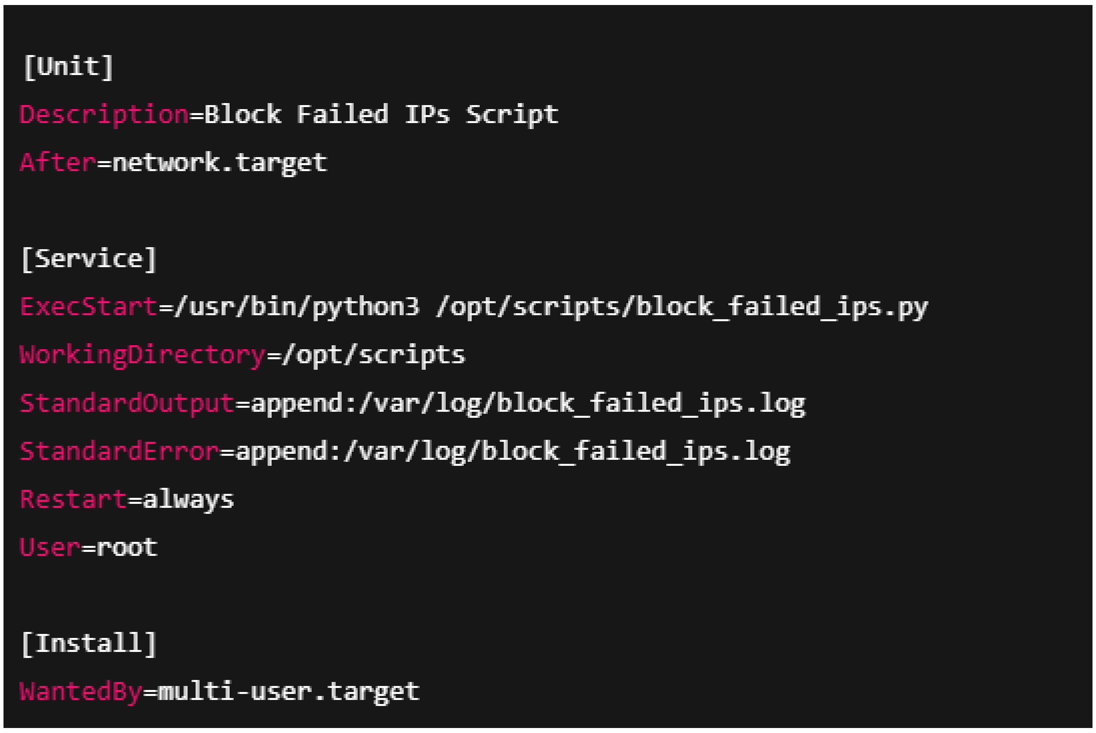
```
    def monitor_logs():
                    time.sleep(1)
                    continue

                match = re.search(FAILED_PATTERN, line)
                if match:
                    ip = match.group(1)
                    if is_whitelisted(ip) or ip in blocked_ips:
                        continue

                    failed_ips[ip].append(now)
                    # Remove entries older than 5 minutes
                    failed_ips[ip] = [
                        t for t in failed_ips[ip] if now - t < timedelta(minutes=5)
                    ]

                    if len(failed_ips[ip]) >= FAIL_LIMIT:
                        block_ip(ip)
                        failed_ips[ip] = []

    except FileNotFoundError:
        logging.error(f"[ERROR] Log file not found: {LOG_FILE}")
    except Exception as e:
        logging.error(f"[EXCEPTION] {str(e)}")

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

Make the script executable

sudo chmod +x/opt/scripts/block_failed_ips.py

Step 2: Create a systemd Service

sudo nano/etc/systemd/system/block_failed_ips.service



```
[Unit]
Description=Block Failed IPs Script
After=network.target

[Service]
ExecStart=/usr/bin/python3 /opt/scripts/block_failed_ips.py
WorkingDirectory=/opt/scripts
StandardOutput=append:/var/log/block_failed_ips.log
StandardError=append:/var/log/block_failed_ips.log
Restart=always
User=root

[Install]
WantedBy=multi-user.target
```

Step 3: Enable and Start the Service

sudo systemctl daemon-reexec
sudo systemctl daemon-reload

sudo systemctl enable block_failed_ips.service
sudo systemctl start block_failed_ips.service

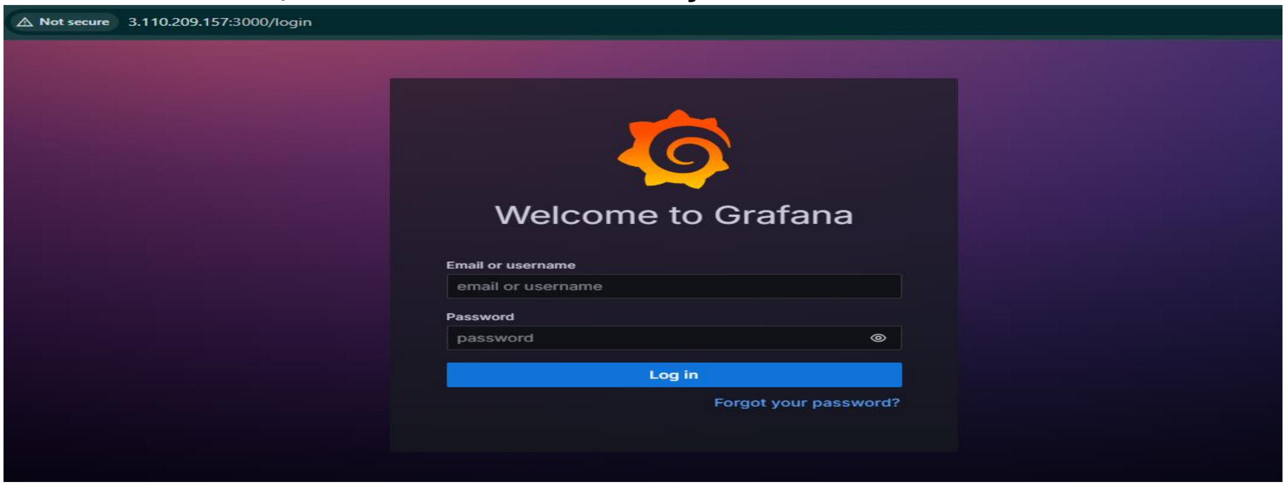
Check service status:

sudo systemctl status block_failed_ips.service

```
2025-06-20 11:42:19,193 [BLOCKED] IP 182.156.140.38 has been blocked for 120 seconds.
2025-06-20 11:42:19,194 [NOTICE] Refer to PDF notice: /opt/scripts/pdfs/blocked_notice.pdf
```

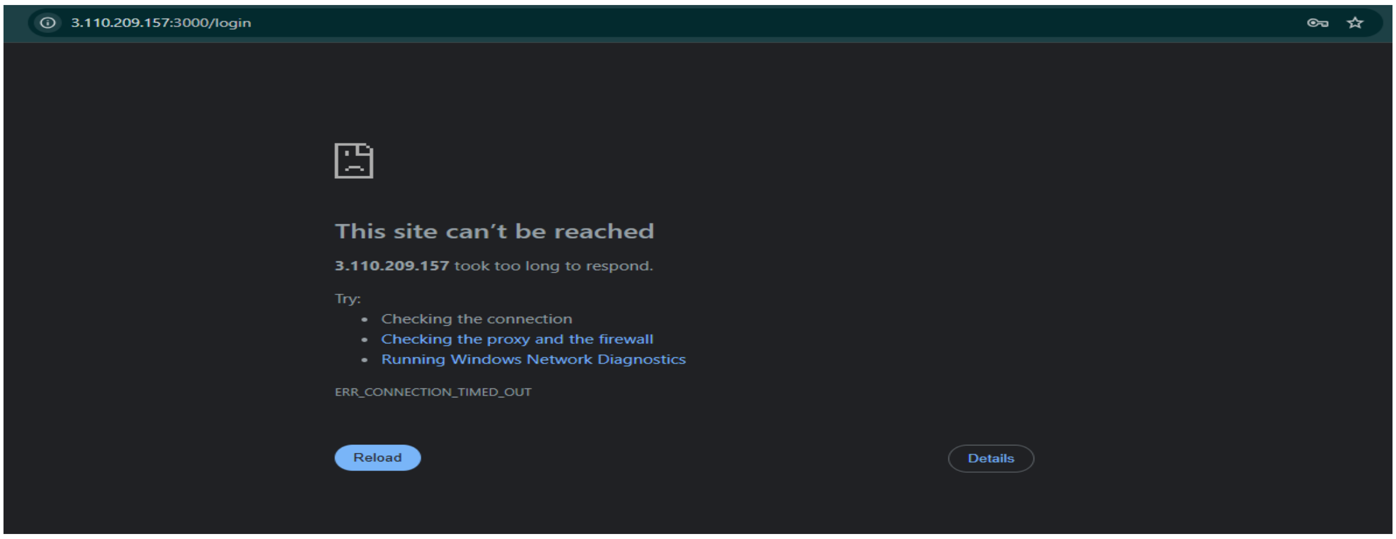
When a user attempts to log in with the wrong password 3 times, their IP is blocked for 120 seconds

After 120 seconds, the IP will be automatically unblocked



Step 4: Verify the Logs

sudo tail -f /var/log/block_failed_ips.log



This site can’t be reached

3.110.209.157 took too long to respond.

Try:

- Checking the connection
- Checking the proxy and the firewall
- Running Windows Network Diagnostics

ERR_CONNECTION_TIMED_OUT