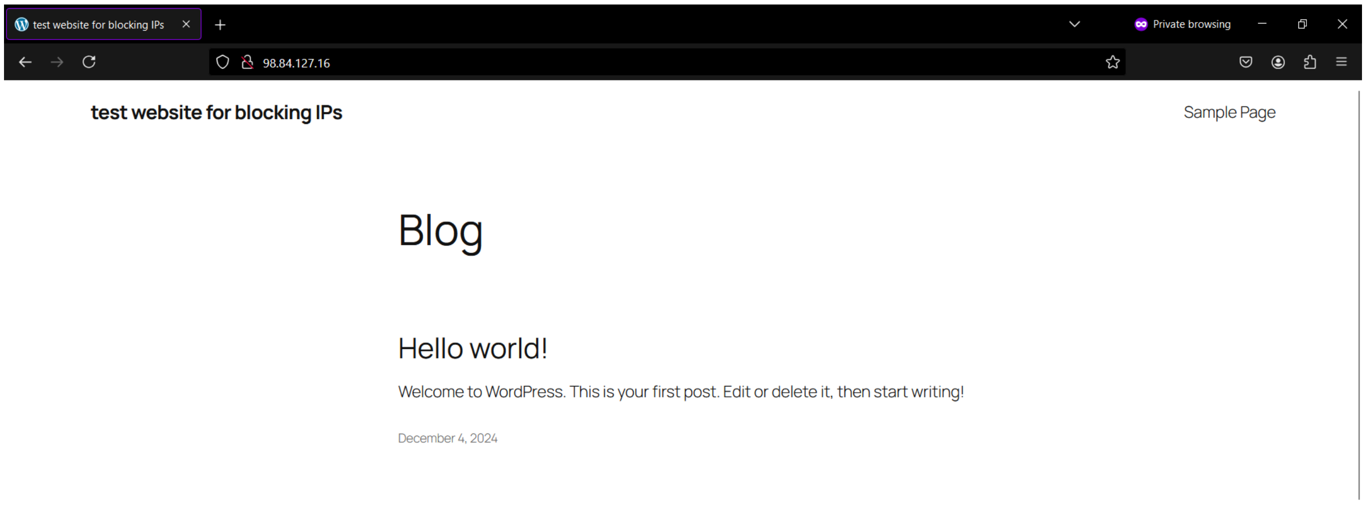
Python script for blocking IPs

Problem Statement :
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.

For this script we require a web server with login so for that we will install wordpress application.



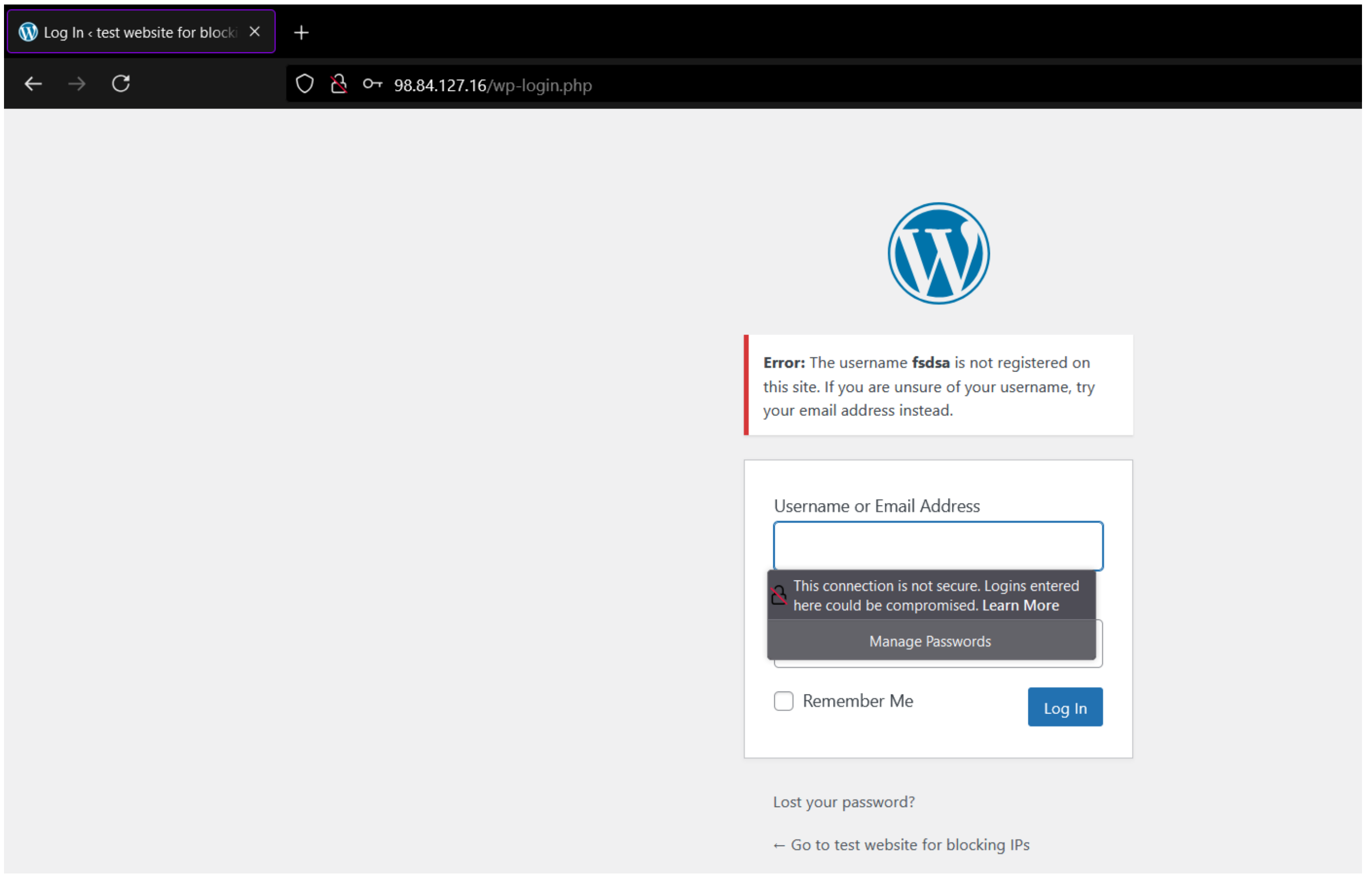
Blog

Hello world!

Welcome to WordPress. This is your first post. Edit or delete it, then start writing!

December 4, 2024

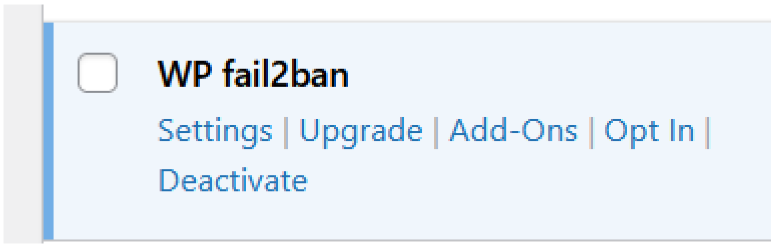
After installing the application go to wp-admin and try to login with incorrect password



Error: The username **fsdsa** is not registered on this site. If you are unsure of your username, try your email address instead.

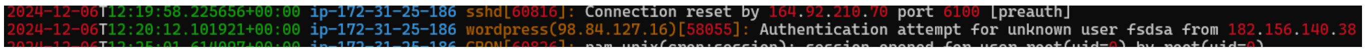
Log In ‹ test website for blocki
98.84.127.16/wp-login.php
Username or Email Address
This connection is not secure. Logins entered here could be compromised. Learn More
Manage Passwords
Remember Me
Log In
Lost your password?
← Go to test website for blocking IPs

these types of logs are stored in /var/log/auth.log but by default wordpress doesn't store authentication logs so we need to install fail2ban plugin



install the above plugin and activate it

after activating we will be able to see unknown user in auth.log



```
2024-12-06T12:19:58.225656+00:00 ip-172-31-25-186 sshd[60816]: Connection reset by 164.92.210.70 port 6100 [preauth]
2024-12-06T12:20:12.101921+00:00 ip-172-31-25-186 wordpress(98.84.127.16)[58055]: Authentication attempt for unknown user fsdsa from 182.156.140.38
2024-12-06T12:25:01.614097+00:00 ip-172-31-25-186 CRON[60826]: pam_unix(cron:session): session opened for user root(uid=0) by root(uid=0)
```

for the python script add the path of auth.log file

for running the script enter the command

```
python3 ipblocker.py
```

```python
import re
import subprocess
```

```
from collections import Counter
import os

# Path to the WordPress log file
LOG_FILE = "/var/log/auth.log"  # Update this path if needed

# Define the threshold for failed login attempts
FAILURE_THRESHOLD = 5

# Regular expression to match failed login attempts in WordPress logs
FAILED_LOGIN_PATTERN = r"Authentication attempt for unknown user .* from
(\d+\.\d+\.\d+\.\d+)"

def parse_logs(file_path):
    """
    Parse the log file and extract IP addresses causing failed login attempts.
    """
    failed_ips = []
    try:
        with open(file_path, 'r') as log_file:
            for line in log_file:
                match = re.search(FAILED_LOGIN_PATTERN, line)
                if match:
                    failed_ips.append(match.group(1))
    except FileNotFoundError:
        print(f"Log file not found: {file_path}")
    except PermissionError:
        print(f"Permission denied: Unable to read {file_path}")
    return failed_ips

def update_firewall(blocked_ips):
    """
    Add firewall rules to block IP addresses using iptables.
    """
    for ip in blocked_ips:
        try:
            # Check if the IP is already blocked
            result = subprocess.run(
                ["iptables", "-L", "-n"],
                stdout=subprocess.PIPE,
                stderr=subprocess.PIPE,
                text=True
            )
            if ip in result.stdout:
                print(f"IP {ip} is already blocked.")
                continue
            # Add a rule to block the IP
            print(f"Blocking IP: {ip}")
            subprocess.run(
                ["iptables", "-A", "INPUT", "-s", ip, "-j", "DROP"],
                check=True
            )
        except subprocess.CalledProcessError as e:
            print(f"Error updating firewall for IP {ip}: {e}")

def main():
    # Ensure the script is run as root
    if os.geteuid() != 0:
        print("This script must be run as root.")
        return

    # Parse logs to get IPs with failed login attempts
    failed_ips = parse_logs(LOG_FILE)
    if not failed_ips:
        print("No failed login attempts found.")
        return

    # Count the occurrences of each IP
    ip_counts = Counter(failed_ips)
    print("Failed login attempts per IP:")
    for ip, count in ip_counts.items():
        print(f"{ip}: {count} attempts")

    # Identify IPs exceeding the threshold
    blocked_ips = [ip for ip, count in ip_counts.items() if count >
FAILURE_THRESHOLD]
    if not blocked_ips:
        print("No IPs exceed the failure threshold.")
        return

    # Block IPs
    update_firewall(blocked_ips)

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

for unblocking ip

```
sudo iptables -D INPUT -s <IP> -j DROP
```

```

- created by:- Chandrakant
```