Log Parser & Alert Script with Python

Developer: Ansh Saini

Objective

To build an automated Python script that parses a list of authorized IP addresses, removes blacklisted IPs, generates real-time alert messages, and logs all removal activity. This project simulates a common responsibility of a SOC (Security Operations Center) Analyst in maintaining secure access control.

Project Description

The project focuses on **automated IP allowlist management**. It works by reading a text file containing allowed IP addresses and removing those IPs that have been blacklisted. The script also prints real-time alerts for every removed IP and logs the removal event in a timestamped log file.

This simulation reflects the routine operations in cybersecurity where analysts are required to continuously audit and clean IP access logs.

Scope of the Project

- Understand and simulate real-world SOC analyst duties.
- Demonstrate basic file parsing, alerting, and automation.
- Gain foundational skills in IP validation and logging using Python.
- Build a script that can evolve into a full-scale log monitoring or alert system.

Technologies Used

Component	Description
Language	Python 3.x
Modules	`re` (regex), `datetime` (logging)
Input	`allow_list.txt`
Output	`removed_ips.log`, terminal alerts

Methodology (How It Works)

Step 1: Load the Input

- Reads the file `allow_list.txt` which contains a list of IP addresses.

Step 2: Define a Blacklist

- A `remove_list` contains IPs that should no longer have access.

Step 3: IP Validation

- Each IP is validated using a regular expression to ensure correct IPv4 format.

Step 4: Alert Generation

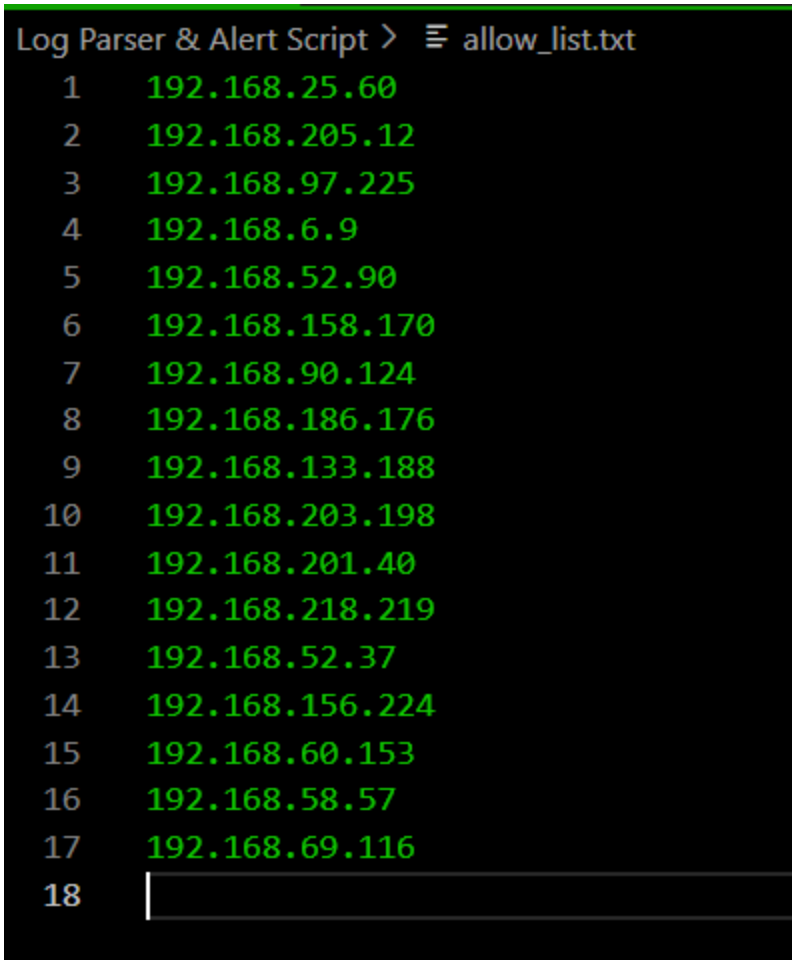
- If an IP from the blacklist is found in the allow list:
    - It is removed.
    - A real-time alert is printed to the terminal.
    - A timestamped entry is logged in `removed_ips.log`.

Step 5: Overwrite File

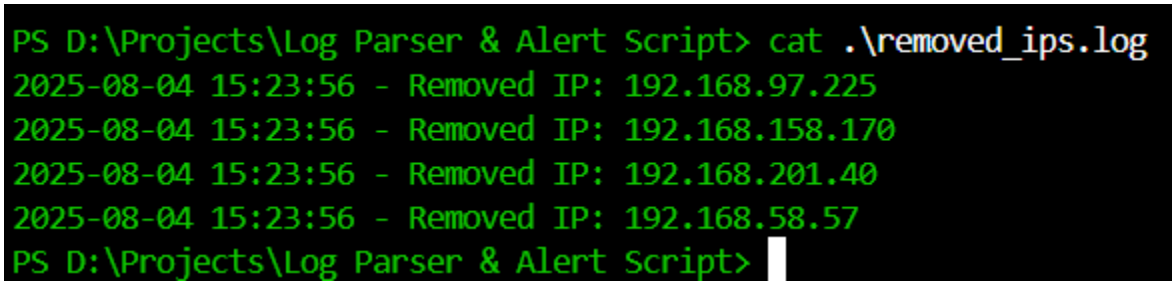
- The updated list of IPs is written back to the original `allow_list.txt` file.

Sample Input

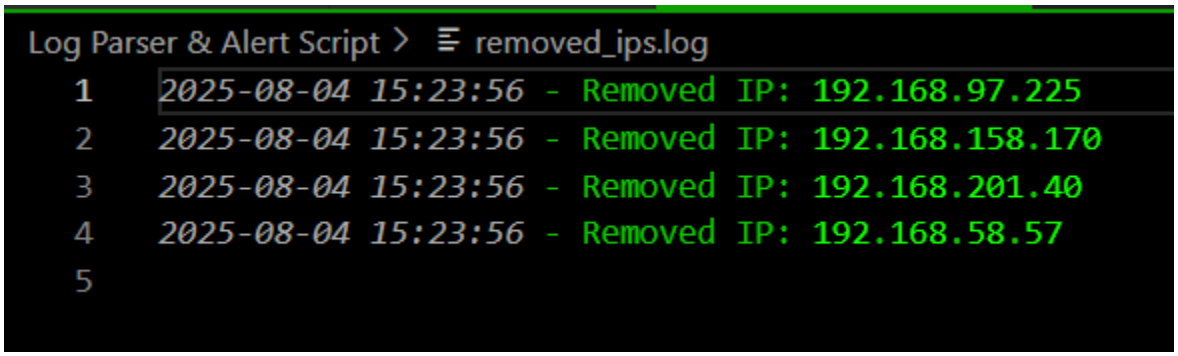
File: `allow_list.txt` (before cleaning)



Sample Output



Log File (`removed_ips.log`)



Final Code Snapshot

Log Parser & Alert Script > log_parser.py > ...

```python
import re
from datetime import datetime

# File name containing the allowed IPs
import_file = "allow_list.txt"

# File to log removed IPs
log_file = "removed_ips.log"

# List of IPs to remove
remove_list = [
    "192.168.97.225",
    "192.168.158.170",
    "192.168.201.40",
    "192.168.58.57"
]

# Function to validate IPv4 format
def is_valid_ip(ip):
    pattern = re.compile(r"^(?:\d{1,3}\.){3}\d{1,3}$")
    return pattern.match(ip) is not None

# Step 1: Read the file contents
with open(import_file, "r") as file:
    ip_addresses = file.read().split()

# Step 2: Open log file for appending
with open(log_file, "a") as log:
    for ip in remove_list:
        if not is_valid_ip(ip):
            print(f"[WARNING] Skipped invalid IP format: {ip}")
            continue

        if ip in ip_addresses:
            ip_addresses.remove(ip)
            print(f"[ALERT] Blacklisted IP removed: {ip}")

            # Log the removal with timestamp
            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            log.write(f"{timestamp} - Removed IP: {ip}\n")

# Step 3: Write updated IP list back to the file
with open(import_file, "w") as file:
    file.write("\n".join(ip_addresses))

print("✅ IP list updated and alerts logged.")

```

Applications & Future Scope

- Integrate with email or Slack for external alerts.
- Convert into a command-line utility.
- Extend to handle CSV or JSON log formats.
- Add backup functionality for the original allow list.

Conclusion

This project is a foundational example of how simple scripts can automate critical security tasks like access control list management. With minimal resources and code, this tool simulates the role of a SOC analyst by performing parsing, validation, alerting, and logging of access logs.

It serves as a stepping stone to building more advanced tools for log monitoring, intrusion detection, and network security management.