Log Analyzer & Alert System – Mini SIEM in Python
Author: Samuele Sparno
Date: June 2025

---

1. Introduction

This project was created with the goal of simulating the behavior of a mini-SIEM (Security Information and Event Management), focusing on the analysis of system log files, the detection of suspicious behavior, and the automatic generation of alerts.
The main objective is to strengthen practical skills in cybersecurity detection through a modern and modular stack based on Python.
The system supports event visualization through a Grafana dashboard and integrates with Elasticsearch for alert indexing and search.

---

2. System Architecture

Execution Flow

1. The `auth.log` file is analyzed to extract suspicious events (e.g., failed logins)
2. YAML rules define the detection criteria (e.g., 5 failures from the same IP within 30 seconds)
3. Generated alerts are saved in `alerts.json` and sent via email
4. Each alert is indexed into Elasticsearch
5. Grafana visualizes the indexed data on a dashboard

---

3. Technical Analysis

Log Parsing

The script `analyze_logs.py` uses regex to identify failed login events in the `auth.log` file.
The data is structured into Python dictionaries and converted to JSON.

Detection Rules

Defined in `rules/detection_rules.yaml`, each rule specifies:
• Event type
• Field to monitor (e.g., IP)
• Trigger threshold
• Time window

Alert Generation

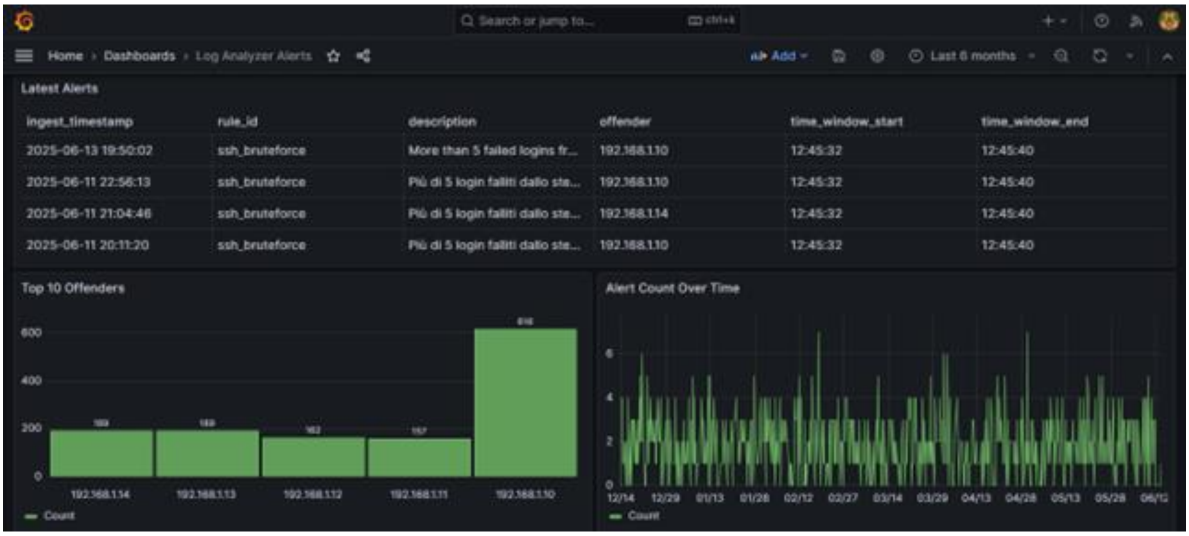
When a rule is matched, a structured alert is created with:
• Rule ID
• Description
• Involved user IP
• Time window

Email Sending

Each alert is notified via email if enabled in `config.yaml`.
The `smtplib` and `EmailMessage` libraries are used.

Elasticsearch
Alerts are indexed into the local Docker Elasticsearch cluster (port 9200).
Each document includes:
- Timestamp
- IP
- Description

Grafana
Grafana connects to Elasticsearch to display alerts:
- Table of recent alerts
- Bar chart of most active IPs
- Alert count timeline



ingest_timestamp	rule_id	description	offender	time_window_start	time_window_end
2025-06-13 19:50:02	ssh_bruteforce	More than 5 failed logins fr...	192.168.1.10	12:45:32	12:45:40
2025-06-11 22:56:13	ssh_bruteforce	Più di 5 login falliti dallo ste...	192.168.1.10	12:45:32	12:45:40
2025-06-11 21:04:46	ssh_bruteforce	Più di 5 login falliti dallo ste...	192.168.1.14	12:45:32	12:45:40
2025-06-11 20:11:20	ssh_bruteforce	Più di 5 login falliti dallo ste...	192.168.1.10	12:45:32	12:45:40

---

4. Technologies Used
- Python 3
- PyYAML for rule parsing
- Regex for event identification
- smtplib / EmailMessage for email notifications
- Elasticsearch on Docker
- Grafana on Docker
- YAML / JSON for config and output

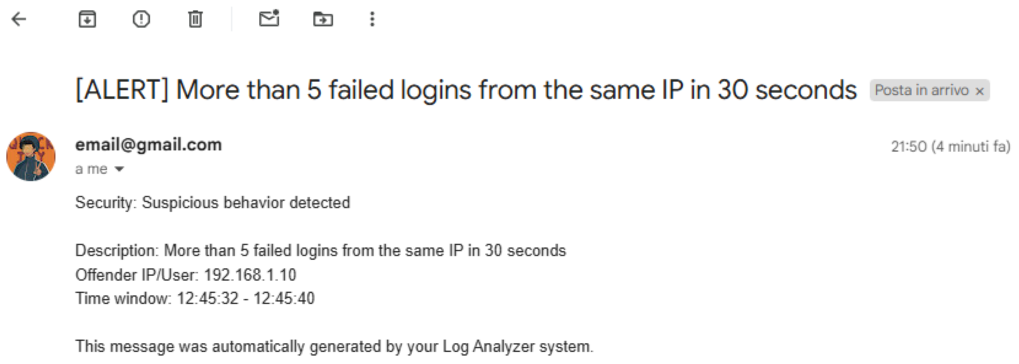
---

5. Output Examples

JSON Alert

```
json
CopiaModifica
{
  "rule_id": "ssh_bruteforce",
  "description": "More than 5 failed logins from the same IP in 30 seconds",
  "offender": "192.168.1.10",
  "count": 5,
  "time_window_start": "12:45:32",
  "time_window_end": "12:45:40"
}
```

Email
Subject: ALERTALERTALERT SSH brute-force detected
Body:
Suspicious activity detected.
Rule: SSH brute-force
IP: 192.168.1.10
Window: 12:45:32 - 12:45:40



[ALERT] More than 5 failed logins from the same IP in 30 seconds

email@gmail.com
a me
21:50 (4 minuti fa)

Security: Suspicious behavior detected

Description: More than 5 failed logins from the same IP in 30 seconds
Offender IP/User: 192.168.1.10
Time window: 12:45:32 - 12:45:40

This message was automatically generated by your Log Analyzer system.

---

6. Conclusion
This project represents a complete practical case of detection and alerting in the security field.
Despite its simple implementation, each component resembles real elements used in professional SIEMs.
I was able to consolidate knowledge in log parsing, regex, event handling, SMTP security, indexing, and dashboard visualization.