Log Monitoring & Alerting System for Linux Servers

by:

Abdul Hafees

Abdulhafeesph1@gmail.com

Introduction

The objective of this project was to design and implement a log monitoring and alerting system for Linux servers that can detect and respond to potential security threats in real time. The system focuses on identifying suspicious activities such as failed SSH login attempts, privilege escalation, and unauthorized access attempts, which are common indicators of malicious activity.

To enhance incident response, the system was integrated with multiple real-time alerting mechanisms, including Email, Slack, and Telegram, ensuring that alerts are immediately delivered to the security team. Logs were shipped using Filebeat and stored in Elasticsearch, while Kibana was used to create dashboards and visualizations for monitoring and analysis.

This solution provides a practical foundation for Security Operations Center (SOC) practices, enabling proactive detection and visualization of threats while improving overall situational awareness in Linux server environments.

Setup and Configuration

The following steps summarize the main setup and configuration performed during the project:

• Installed and configured Filebeat to collect log data.

• Configured Kibana and Elasticsearch for indexing and visualization.

• Edited configuration YAML files to define inputs, outputs, and alert log locations.

• Created scripts to detect suspicious activities and log them into a dedicated file (/var/log/security_alerts.log).

• Integrated Email (SMTP), Slack, and Telegram APIs to send alerts.

Important Commands Used

sudo filebeat modules enable system

sudo filebeat setup

curl -k -u elastic https://localhost:9200/_cat/indices?v

curl -k -u elastic:password "https://localhost:9200/filebeat-*/_search?q=message:ALERT&size=5" | jq .

GET _cat/indices

DELETE filebeat-* (to reset indices when needed)

sudo nano /etc/filebeat/filebeat.yml (to configure Filebeat inputs and outputs)

Python Monitoring Script

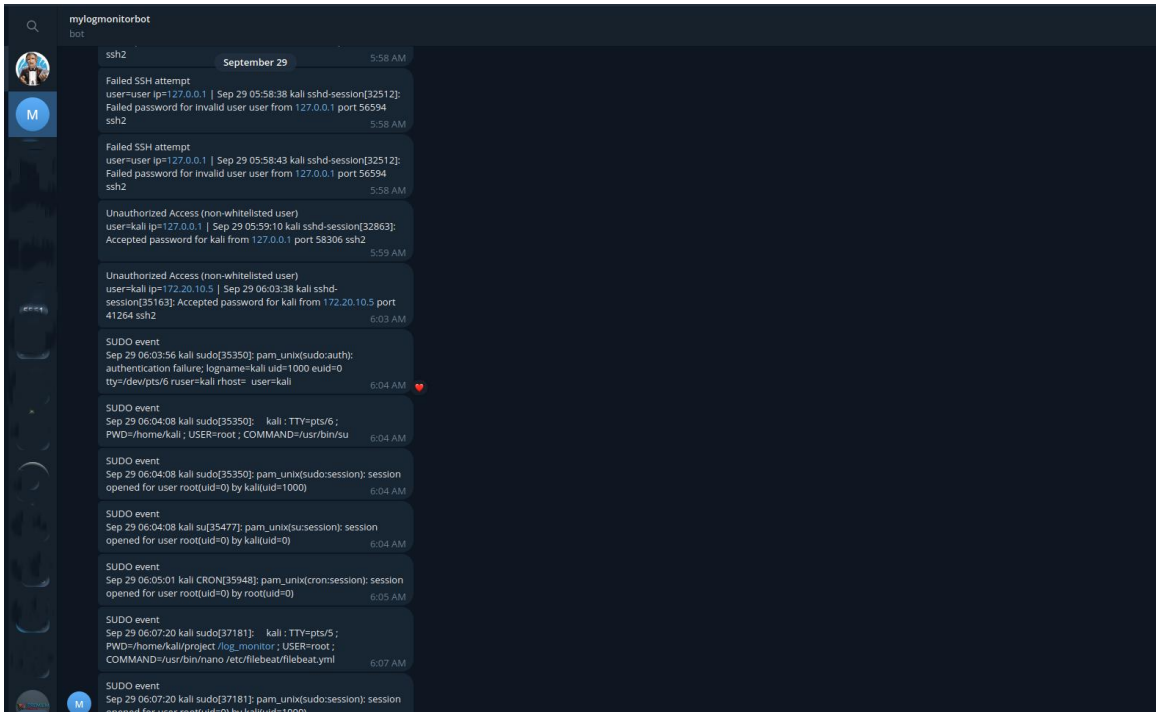
```
#!/usr/bin/env python3
import os
import re
import smtplib
import requests
import subprocess
from email.mime.text import MIMEText

# Log file to store alerts
ALERT_LOG = "/var/log/security_alerts.log"

# Slack, Telegram, Email configs (tokens, SMTP etc.)
SLACK_WEBHOOK = "https://hooks.slack.com/services/XXX/YYY/ZZZ"
TELEGRAM_TOKEN = "your-telegram-token"
TELEGRAM_CHAT_ID = "your-chat-id"
SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587
EMAIL_USER = "your_email@gmail.com"
EMAIL_PASS = "your_password"
EMAIL_TO = "receiver_email@gmail.com"

def send_slack(msg):
    requests.post(SLACK_WEBHOOK, json={"text": msg})

def send_telegram(msg):
    url = f"https://api.telegram.org/bot{TELEGRAM_TOKEN}/sendMessage"
    requests.post(url, data={"chat_id": TELEGRAM_CHAT_ID, "text": msg})

def send_email(msg):
    mime = MIMEText(msg)
    mime["From"] = EMAIL_USER
    mime["To"] = EMAIL_TO
    mime["Subject"] = "Security Alert"
    with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
        server.starttls()
        server.login(EMAIL_USER, EMAIL_PASS)
        server.sendmail(EMAIL_USER, EMAIL_TO, mime.as_string())
```

```
def log_alert(message):
    with open(ALERT_LOG, "a") as f:
        f.write(message + "\n")
    send_slack(message)
    send_telegram(message)
    send_email(message)

def monitor_logs():
    ssh_pattern = re.compile(r"Failed password")
    sudo_pattern = re.compile(r"incorrect password|authentication failure")
    with subprocess.Popen(["tail", "-F", "/var/log/auth.log"], stdout=subprocess.PIPE,
stderr=subprocess.PIPE) as p:
        for line in iter(p.stdout.readline, b""):
            decoded = line.decode("utf-8").strip()
            if ssh_pattern.search(decoded):
                log_alert(f"ALERT: Failed SSH attempt | {decoded}")
            elif sudo_pattern.search(decoded):
                log_alert(f"ALERT: SUDO event | {decoded}")

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

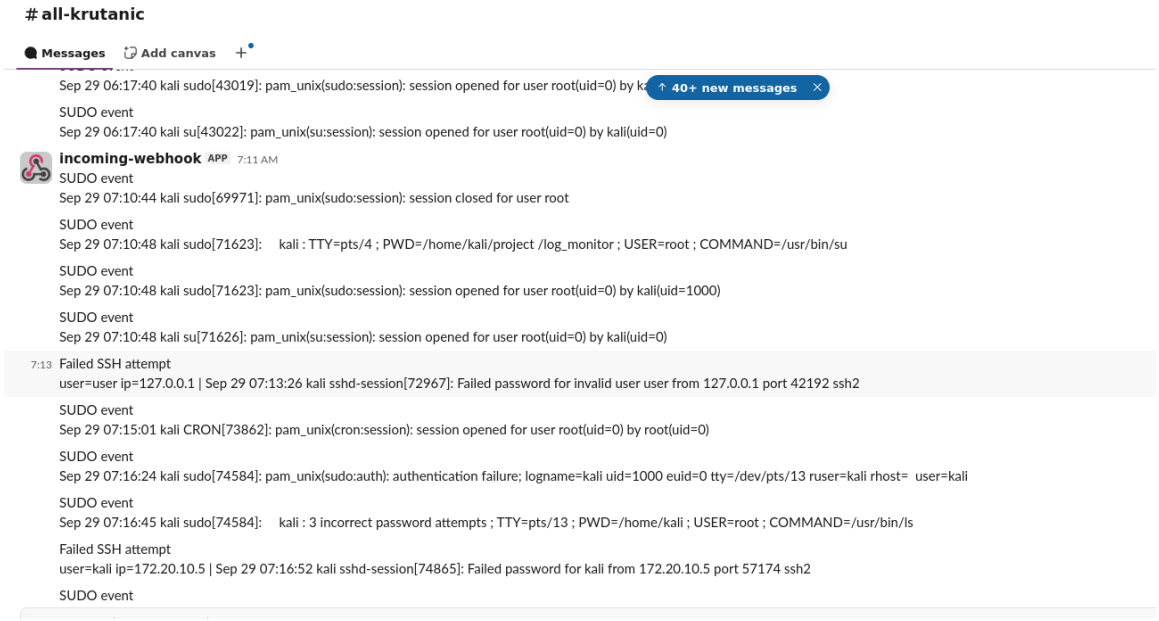
Alerting Integrations

Alerts were successfully integrated and tested across the following channels:

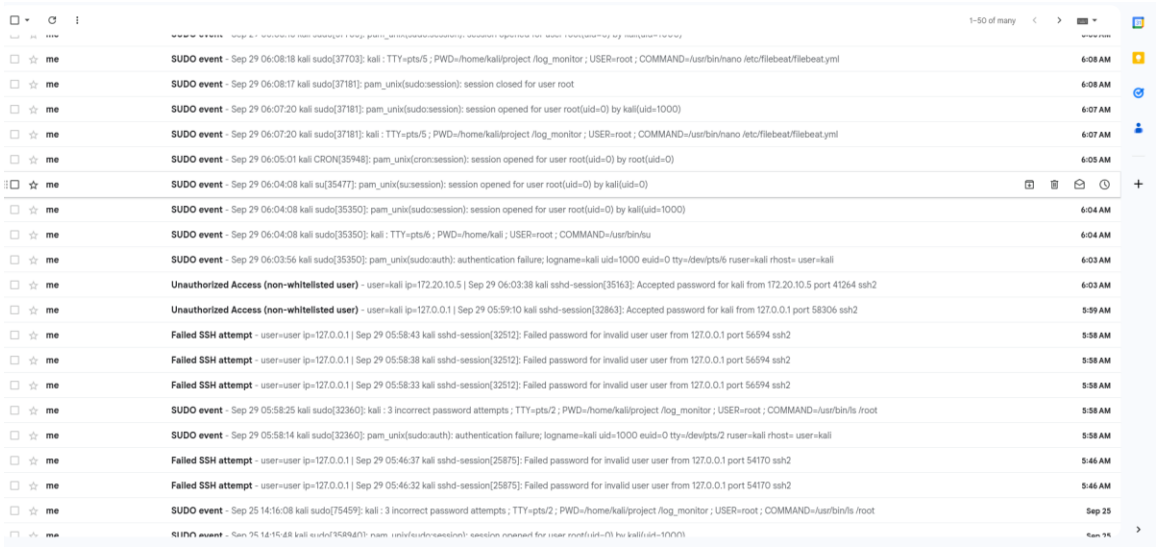
• Telegram



• Slack

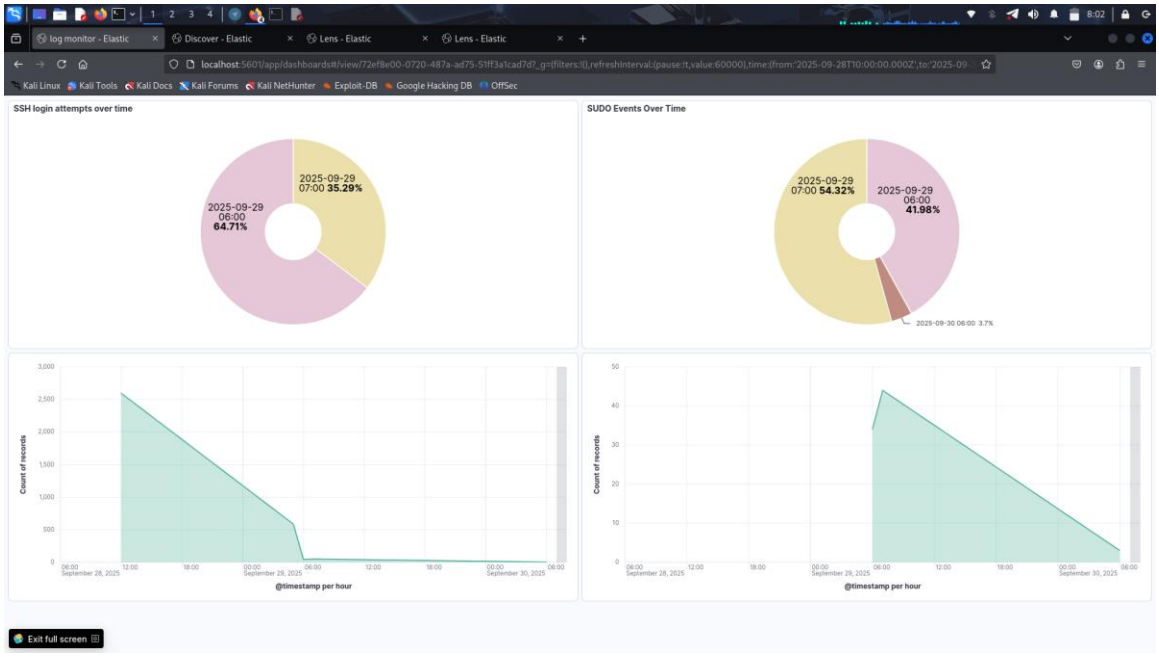


• Email



Visualization in Kibana

Kibana was used to visualize the collected alerts. The following dashboards and visualizations were created:



Conclusion

This project successfully demonstrated the implementation of a real-time security monitoring and alerting system. Logs were collected, processed, and sent to Elasticsearch for visualization in Kibana. Alerts for suspicious activities such as failed SSH login attempts and privilege escalation were generated and delivered via multiple communication channels.

This project successfully implemented a real-time security monitoring system with

- Log monitoring and parsing through a custom Python script.
- Alerts to multiple channels (Telegram, Slack, Gmail).
- Log ingestion using Filebeat into Elasticsearch.
- Visualization using Kibana dashboards.