Full-Stack Web Server Monitoring & Security Dashboard

Project Overview

This project demonstrates the end-to-end process of building a comprehensive monitoring and security solution using Splunk. The goal was to collect, analyze, and visualize data from a web server and its underlying operating system to gain real-time insights into system health and potential security threats.

Technologies Used

- **Log & Analytics Platform:** Splunk Enterprise (Host) & Splunk Universal Forwarder (Agent)
- **Operating Systems:** Kali Linux (Host OS), Parrot OS (Virtual Machine)
- **Virtualization:** VirtualBox
- **Web Server:** Apache2
- **Network Utilities:** SSH

Key Project Phases & Accomplishments

1. Data Ingestion & Collection

- **Environment Setup:** Configured a virtual lab using VirtualBox, with **Kali Linux** as the host for Splunk Enterprise and **Parrot OS** as a web server to generate log data.
- **Log Forwarding:** Installed and configured the **Splunk Universal Forwarder** on the Parrot OS VM to collect and forward logs from the Apache web server and the Linux OS.
- **Troubleshooting:** Successfully troubleshooted an issue where auth.log data was not being ingested due to file path differences on Parrot OS. The solution involved verifying file locations and configuring a custom log file for successful data forwarding, demonstrating strong problem-solving skills.
- **Receiver Configuration:** Configured a receiving port (**TCP 9997**) on the Splunk Enterprise instance to listen for incoming data from the Universal Forwarder.

Configuration Commands:

- **On Parrot OS (UF):**

  Bash

  ```
  sudo /opt/splunkforwarder/bin/splunk add forward-server <your_kali_ip>:9997
  sudo /opt/splunkforwarder/bin/splunk add monitor /var/log/apache2/access.log -sourcetype apache:access
  sudo /opt/splunkforwarder/bin/splunk add monitor /var/log/custom_auth.log -sourcetype custom:auth
  ```

- **On Kali Linux (Splunk Enterprise):**

  Bash

  ```
  sudo /opt/splunk/bin/splunk enable listen 9997 -auth admin:your_splunk_password
  ```

2. Data Parsing & Visualization

- **Field Extraction:** Utilized Splunk's **GUI Field Extractor** to parse raw log data from Apache and custom logs, creating structured fields such as client_ip, status_code, and request_url.

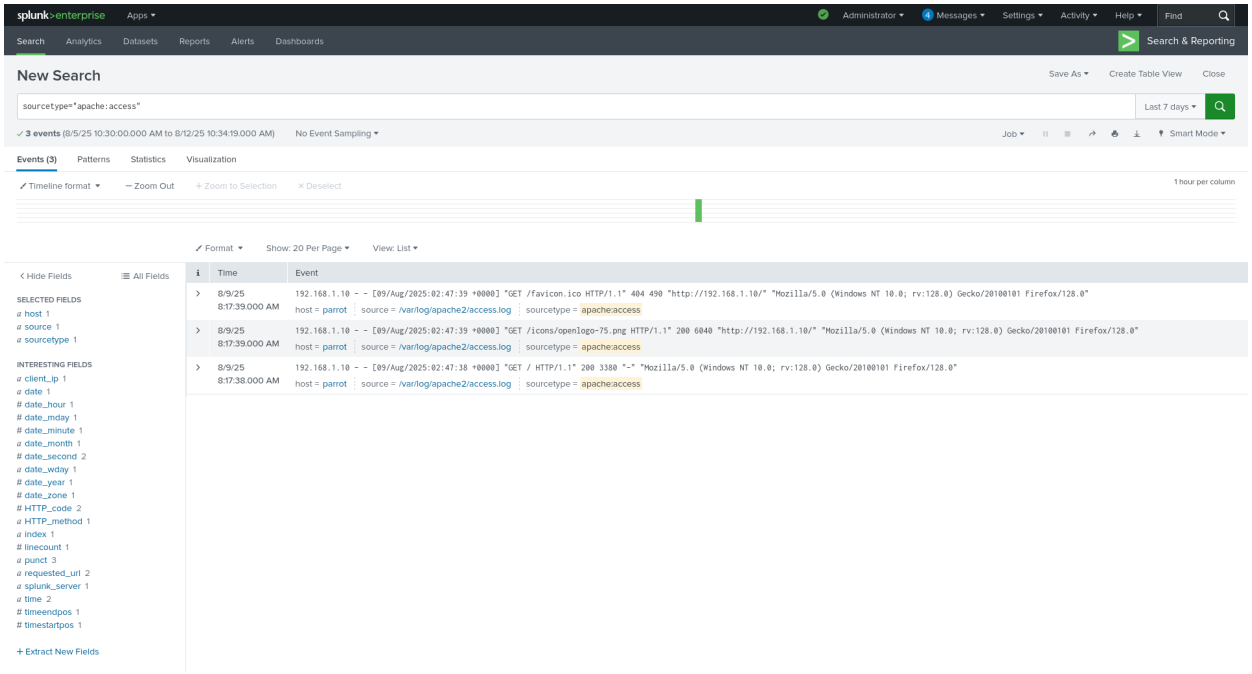
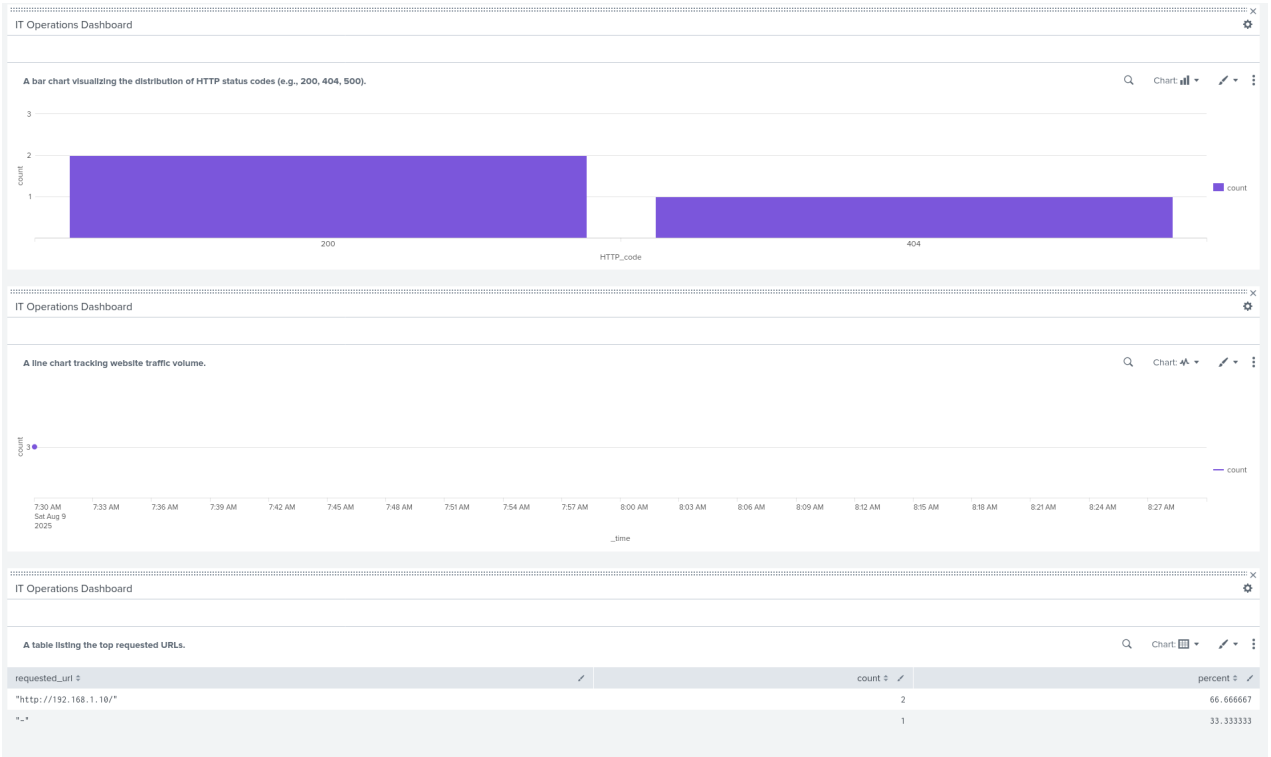


Figure 1: Search & Reporting

- **IT Operations Dashboard:** Built a dashboard to provide a real-time overview of web server performance.



- **Figure 2: IT Operations Dashboard**

- **Security Analytics Dashboard:** Developed a separate dashboard focused on security, featuring threat indicators like failed logins and scanning activity.

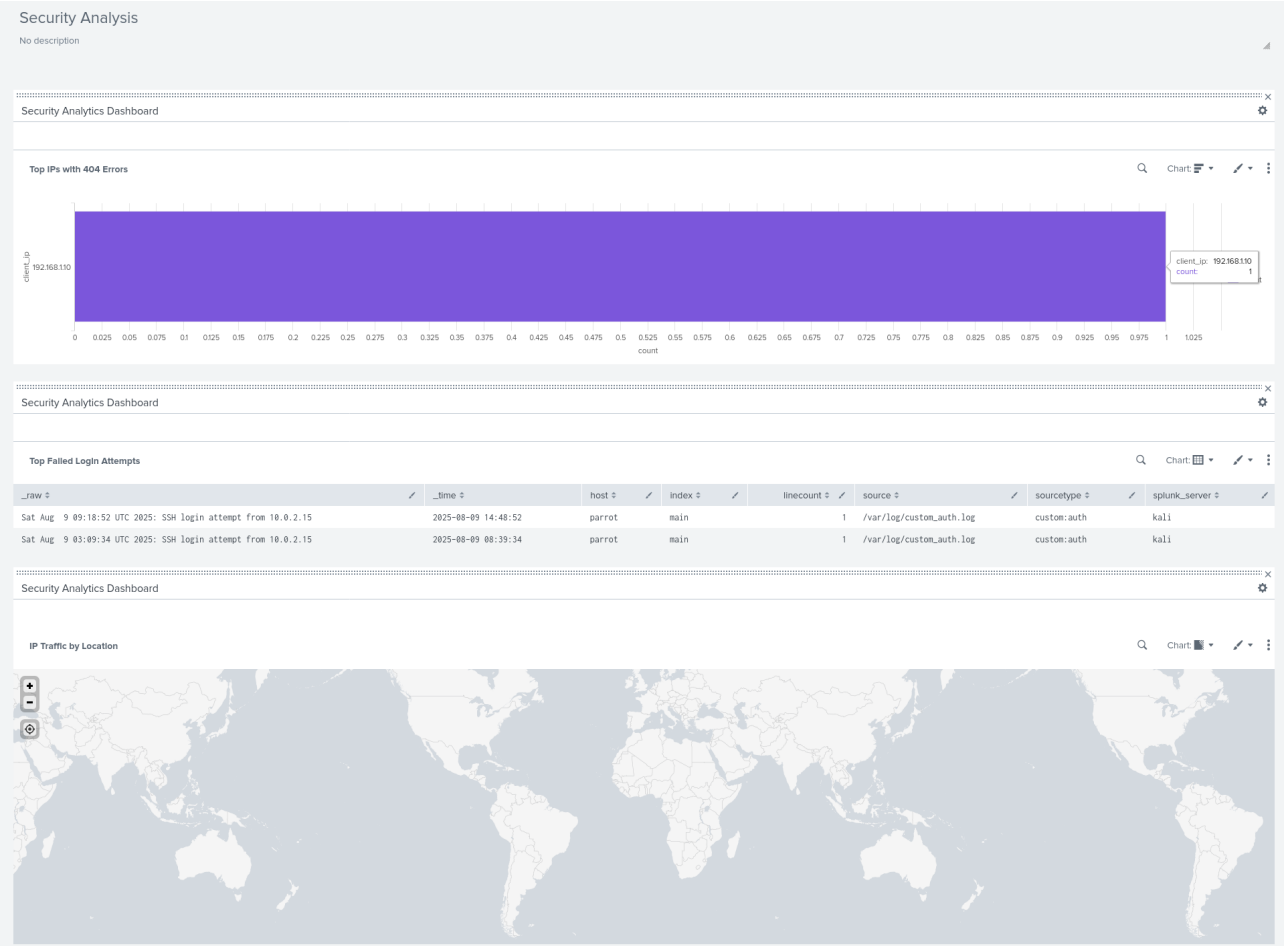


Figure 3: Security Analytics Dashboard

SPL Queries:

- Website Traffic Volume (Line Chart):

  sourcetype="apache:access" | timechart span=1h count

- HTTP Status Codes (Bar Chart):

  sourcetype="apache:access" | stats count by HTTP_code

- Top 404 Error IPs (Bar Chart):

  sourcetype="apache:access" HTTP_code=404 | stats count by client_ip | sort -count

- Failed Login Attempts (Table):

  sourcetype="custom:auth" | top client_ip

- Traffic by Location (Map Panel):

  sourcetype="apache:access" | iplocation client_ip | geostats count by client_ip

3. Automated Alerting

- **Threat Detection:** Configured **scheduled alerts** to provide proactive notifications of security incidents without requiring constant manual monitoring of the dashboards.
- **Brute-Force Alert:** Created a scheduled alert that triggers when a single IP address has more than 10 failed login attempts within a 5-minute period.
- **Scanning Alert:** Set up a scheduled alert to identify and notify about potential web server scanning by flagging a high number of 404 errors from a single IP address.

Alert SPL Queries:

- Brute-Force Alert:

  sourcetype="custom:auth" | stats count by client_ip | where count > 10

- Scanning Activity Alert:

  sourcetype="apache:access" HTTP_code=404 | stats count by client_ip | where count > 20