Innovation Center for Education

PHEMESOFT
Software That Matters
TRUSTED DELIVERY PARTNER for IBM ICE

YENEPOYA UNIVERSITY

THREAT HUNTING TOOLKIT

PROJECT SYNOPSIS

THREAT DETECTION SYSTEM

BACHELOR OF COMPUTER APPLICATIONS
cyber forensics, data analytics and
cyber security

SUBMITTED BY

MUHAMMED IRFAN AK - 22BCACDC44

JACOB JOY - 22BCACDC24

AKSHAY KRISHNA PS - 22BCACDC09

ABINAV K - 22BCACDC05

SAYANTH S - 22BCACDC63

GUIDED BY

SUMITH K SHUKLA

Table of Contents

1. Introduction

The Threat Detection System is an advanced, web-based cybersecurity tool designed to proactively identify, analyze, and report potential threats within an organization's IT infrastructure. As organizations increasingly rely on digital networks and web applications, the frequency and complexity of cyber-attacks such as unauthorized access, malware infections, and data breaches have significantly risen. This makes early threat detection a critical aspect of modern cybersecurity strategies.

This system integrates two key functionalities — a Log File Analyzer and an IP/Domain Reputation Checker. The Log File Analyzer examines system-generated logs, which are often the first indicators of suspicious or malicious activity. It parses these logs, identifies abnormal behavior patterns, unauthorized access attempts, and unusual login activities, flagging them for further action.

The IP/Domain Reputation Checker assesses the credibility of external entities by querying reputable cybersecurity services such as VirusTotal and AbuseIPDB. These services maintain extensive global databases of known malicious IP addresses and domains, allowing the system to evaluate whether interactions with specific addresses pose a threat.

The application is built using the Django framework, chosen for its scalability, security, and rapid development capabilities, with SQLite serving as the lightweight, embedded database system. Additionally, Chart.js is incorporated for data visualization, enabling administrators to review security analytics through interactive charts and dashboards. This project aims to deliver an efficient, automated, and easy-to-use platform for network administrators to strengthen cybersecurity defenses, reduce human effort, and minimize incident response times.

2. Methodology / Planning of Work

The development of the Threat Detection System follows a systematic, phased methodology to ensure that each component is carefully planned, designed, implemented, and tested for optimal performance and reliability. The following stages outline the project's workflow:

1. Requirement Analysis: Conduct detailed research to gather comprehensive functional and non-functional requirements. This involves interacting with potential end-users, studying similar existing systems, and consulting security standards to define system goals, performance benchmarks, and operational constraints.
2. UI and Database Design: Design a user-friendly and intuitive interface using HTML, CSS, and JavaScript, ensuring clear navigation and accessible controls for both technical and non-technical users. Simultaneously, develop a normalized SQLite database schema to securely store log entries, threat records, and API responses, maintaining data consistency and minimizing redundancy.
3. Development of Log File Analyzer: Build a module capable of reading various system and server log files, parsing entries, and analyzing them for anomalies such as failed login attempts, irregular traffic, or unauthorized access attempts. The analyzer flags any detected irregularities for administrative review.
4. API Integration: Integrate with VirusTotal and AbuseIPDB APIs, implementing secure and efficient methods to query the reputation of IP addresses and domains. Responses are parsed and categorized based on risk levels, and malicious entities are highlighted within the system.
5. Module Testing: Execute rigorous unit testing for each module independently to verify functionality, accuracy, and robustness under different simulated threat conditions. Emphasis is placed on error handling, response times, and data integrity.
6. Final Integration and Deployment: Merge all modules into a unified application, perform integration testing to confirm seamless interaction between components, and deploy the application on a local server or cloud environment. Comprehensive system testing ensures stability, scalability, and security before final presentation or operational use.

3. Facilities Required

The successful design, development, testing, and deployment of the Threat Detection System requires a combination of reliable software tools, hardware resources, and networking facilities. These are selected based on compatibility, ease of integration, and efficiency in supporting cybersecurity applications.

Software Requirements:
- Django Framework (Python): Robust web framework ideal for developing secure, scalable applications rapidly.
- SQLite Database: Lightweight, serverless, and reliable database for managing application data without complex setup.
- HTML, CSS, JavaScript: For designing responsive, user-friendly front-end interfaces accessible through any modern browser.
- Chart.js Library: JavaScript-based charting library for creating dynamic and interactive charts to visualize security data and activity logs.
- VirusTotal and AbuseIPDB APIs: Third-party cybersecurity services offering global databases for reputation checks of IP addresses and domains.

Hardware Requirements:
- Computer System: Minimum specifications of 4GB RAM, multi-core processor, and 500GB storage for development, testing, and local hosting.
- Stable Internet Connection: High-speed internet connectivity is vital for accessing external APIs, downloading updates, and synchronizing threat databases in real-time.

These facilities collectively ensure the system’s reliable operation during development and deployment, while supporting scalability and ease of maintenance.

4. References

[1] Django Documentation: https:

//docs.djangoproject.com/
[2] SQLite Documentation: https:

//www.sqlite.org/docs.html
[3] Chart.js Documentation:

https://www.chartjs.org/docs
[4] AbuseIPDB API Documentation:

https://www.abuseipdb.com/
[5] VirusTotal API Documentation: https:

//developers.virustotal.com/
[6] OWASP Secure Coding Practices:

https://owasp.org/www-project-secure-coding-practices/