Synopsis

MCAP11 - Mini Project 1A – Semester 1

1. Title of the Project:

Development of a URL-based Intrusion Detection System

2. Introduction:

In the modern digital landscape, the Hypertext Transfer Protocol (HTTP) is the foundation of the World Wide Web. While essential, its ubiquity also makes it a primary target for cyber threat actors. A significant number of these attacks are initiated by manipulating the Uniform Resource Locator (URL) field to exploit vulnerabilities in web applications. These attacks range from data theft to complete server compromise. This project is chosen to address the critical need for automated tools that can analyze web traffic and accurately identify malicious URLs, thereby providing a crucial layer of security intelligence for organizations.

3. Objectives of the Project:

- To design and develop a system capable of detecting a wide range of URL-based cyber-attacks, including SQL Injection, Cross-Site Scripting (XSS), Directory Traversal, Command Injection, and others.

- To implement a robust, rule-based detection engine utilizing regular expressions for high-speed pattern matching of known attack signatures.

- To develop a dynamic, web-based dashboard using a modern JavaScript framework to visualize detected threats, display analytics, and allow for interactive data exploration.

- To build a data processing pipeline that can ingest and analyze security data from two primary sources: structured CSV log files and raw PCAP network capture files.

- To provide functionality for users to filter detected security events by parameters such as attack type and source IP, and to export the filtered results into CSV and JSON formats for reporting.

4. Problem Definition:

Security analysts are often tasked with manually inspecting massive volumes of web server logs and network traffic data to identify threats. This process is not only time-consuming and labor-intensive but also highly prone to human error, meaning sophisticated attacks can go unnoticed. There is a lack of accessible tools that can automatically ingest various data formats (like raw PCAP captures and CSV logs), apply a comprehensive set of detection rules, and present the findings in an intuitive, actionable format. The core problem is the need for an efficient and accurate system that can detect a wide array of URL-based attack **attempts** and also help in identifying potentially **successful** breaches.

5. Scope of the Project:

- **In-Scope:**
  - The system will analyze offline data provided via file uploads (.csv, .pcap).
  - Detection will focus on a pre-defined list of common URL-based attacks.
  - The solution will include a fully functional web-based GUI for visualization and interaction.
  - The project includes the creation of a custom, simulated dataset for training and testing purposes.
- **Out-of-Scope:**
  - The system will not function as a real-time, inline firewall or actively block network traffic.
  - The system will not perform deep packet inspection beyond extracting necessary HTTP request information.
  - User authentication and multi-user support for the dashboard are not within the scope of this mini-project.

6. Proposed System / Solution:

The proposed solution is a full-stack web application with a two-tier architecture designed for efficient analysis and visualization of URL-based threats.

- **Backend (Analysis Engine):** Developed in **Python** using the **Flask** framework, the backend will serve as the core of the system. It will consist of three main modules:
  1. **Data Ingestion Module:** This module will handle file uploads and use libraries like **Pandas** for CSV parsing and **Scapy** for extracting HTTP requests from PCAP files.
  2. **Detection Module:** This is the central engine that will analyze each extracted URL against a library of pre-defined regex patterns to identify and classify attacks.
  3. **API Module:** A RESTful API will expose endpoints for the frontend to fetch analysis results, request data exports, and manage file processing.
- **Frontend (Visualization Dashboard):** A modern single-page application built with

React.js. The frontend will provide an intuitive user interface for:

1. **Dashboard View:** Displaying summary statistics and charts (e.g., a pie chart of attack types) using libraries like **Chart.js**.
2. **Events Table:** Showing a detailed, sortable, and filterable list of all detected malicious requests.
3. **Controls:** Allowing the user to upload files, apply filters, and trigger data exports

7. Technology Used

- **Backend Language: Python**
  - The core language for the analysis engine, chosen for its powerful data science and cybersecurity libraries.
- **API Framework: Flask**
  - A lightweight framework used to create the API that lets the frontend dashboard communicate with the backend.
- **Data Manipulation: Pandas**
  - Used for efficiently loading, reading, and manipulating the input data from .csv log files.
- **PCAP Analysis: Scapy**
  - A packet manipulation library used to open .pcap files and extract HTTP URLs from raw network traffic.
- **Frontend: JavaScript & React.js**
  - Used together to build the interactive and responsive user interface, creating the dashboard you see and interact with.
- **Data Visualization: Chart.js**
  - A JavaScript library that visualizes the final analysis data as interactive charts and graphs on the dashboard.
- **Database: SQLite**
  - A simple, file-based database used to store the analysis results, allowing for fast retrieval of data by the dashboard.
- **Development Tools: VS Code, Git, Postman**
  - Essential tools for the workflow: **VS Code** for writing code, **Git** for version control, and **Postman** for testing the API.

8. References

- [1] C. Opara, Y. Chen, and B. Wei, “Detecting Phishing Web Pages by Exploiting Raw URL and HTML Characteristics,” *arXiv*, 2020. [Online]. Available: https://arxiv.org/abs/2011.04412

- [2] OWASP Foundation, “Server-Side Request Forgery (SSRF),” 2021. [Online]. Available: https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_(SSRF)/

- [3] X. Wang, J. Zhai, and H. Yang, “Detecting Command Injection Attacks in Web Applications Using Deep Learning,” *Nature*, 2024. [Online]. Available: https://www.nature.com/articles/s41598-024-74350-3
- [4] OWASP Foundation, “Testing for Server-Side Request Forgery (SSRF),” 2021. [Online]. Available: https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/19-Testing_for_Server-Side_Request_Forgery

Name: *Vansh Shah*

Roll No.: *MCA2547*

Mentor: *Prof. Puja Devgun*