# Python Developer Assignment Document

**Organization:** C3iHub, IIT Kanpur

## Overview

This document contains two detailed Python development assignments related to cybersecurity data analysis. The goal is to create automated tools capable of parsing, analyzing, and reporting on network security datasets (Firewall Logs and PCAP captures). Both tools should be production-ready, efficient, and include a graphical user interface (GUI) for usability.

## Assignment Files

- **Firewall Log File:** sample_firewall.log
- **PCAP File:** sample_traffic.pcap

## Project 1: Firewall Log Analysis

### Objective

Develop a Python-based tool to automatically parse, enrich, and analyze firewall logs to detect suspicious, malicious, or policy-violating traffic patterns.

### Tasklist

1. **Log Preparation & Parsing**
    1.1. Collect and review logs from Fortigate, Palo Alto, and Sophos.
    1.2. Parse essential fields: Timestamp, Source IP, Destination IP, Ports, Protocol, Action, Rule Name, App, Category.
    1.3. Build a universal parser to handle multiple log formats (regex or delimiter-based).
    1.4. Handle malformed entries gracefully.
2. **Data Normalization & Enrichment**
    2.1. Normalize timestamps into a single consistent format.
    2.2. Add GeoIP lookup for Source/Destination IPs.
    2.3. Integrate Threat Intelligence APIs (AbuseIPDB, VirusTotal, AlienVault OTX).
    2.4. Tag internal vs external IPs.
3. **Detection Logic**
    3.1. Detect repeated connection attempts or port scans.
    3.2. Identify brute-force or denial-of-service attempts.
    3.3. Detect suspicious outbound traffic or data exfiltration.
    3.4. Highlight blocked critical ports (22, 23, 3389, 445).
    3.5. Flag lateral movement and policy violations.
    3.6. Track newly seen IPs and anomalies.
4. **Reporting & Visualization**
    4.1. Generate CSV/Excel summary tables for connections and anomalies.
    4.2. Create visual graphs (Top Source/Destination IPs, Protocol Distribution).
    4.3. Include GeoIP world map visualization.
    4.4. Highlight malicious IPs and suspicious activities.
5. **Automation & GUI Development**
    5.1. Build GUI using CustomTkinter for file selection and output directory.
    5.2. Enable multi-log batch processing with progress bar.
    5.3. Save each output in a folder named after the input file.
    5.4. Implement error logs, summary logs, and runtime exception handling.

**Expected Deliverables**

- Python script: `firewall_analyzer.py`
- GUI-enabled tool with file and folder browsing.
- Excel + HTML summary reports with charts.
- Output tested with the provided `sample_firewall.csv`.

## Project 2: PCAP Analysis

### Objective

Develop a Python-based PCAP analyzer capable of identifying network anomalies, command-and-control behavior, and OWASP Top-10 web application attacks.

### Tasklist

1. **PCAP Parsing & Session Reconstruction**
   - 1.1. Parse PCAP files using PyShark or Scapy.
   - 1.2. Extract Source/Destination IP, Ports, Protocol, Bytes, and Payload.
   - 1.3. Reconstruct TCP and HTTP sessions.
2. **Traffic Classification & Enrichment**
   - 2.1. Categorize traffic: DNS, HTTP, HTTPS, ICMP, FTP, SMTP, SSH, etc.
   - 2.2. Decode Base64 and URL-encoded payloads.
   - 2.3. Perform GeoIP and Threat Intelligence lookups.
3. **Suspicious Network Behavior Detection**
   - 3.1. Detect port scans, SYN floods, and DDoS patterns.
   - 3.2. Identify data exfiltration or beaconing communication.
   - 3.3. Detect malicious DNS queries or fast-flux domains.
4. **OWASP Web Attack Detection**
   - 4.1. **SQL Injection:** Detect keywords like `' OR 1=1 --`, `UNION SELECT`, `xp_cmdshell`.
   - 4.2. **XSS:** Detect HTML/JS injection like `<script>`, `onerror=`.
   - 4.3. **Command Injection:** Identify `; ls`, , `| cat /etc/passwd`.
   - 4.4. **Path Traversal:** Detect `../`, `/etc/passwd`.
   - 4.5. **File Upload Attacks:** Detect attempts to upload scripts (`.php`, `.jsp`, `.asp`).
   - 4.6. **Broken Authentication:** Identify repeated failed logins or session hijacks.
   - 4.7. **Server Misconfiguration:** Detect exposures like `/admin`, `/test`, `/backup`.
5. **Reporting & Visualization**
   - 5.1. Generate Excel + HTML reports summarizing detected anomalies.
   - 5.2. Include visual graphs (Top Talkers, Protocols, Attack Types).
   - 5.3. Highlight IPs and payloads linked to OWASP findings.
6. **Automation & GUI Development**
   - 6.1. Build CustomTkinter GUI for file browsing.
   - 6.2. Support multiple PCAPs with progress bar.
   - 6.3. Implement error handling for large/corrupt PCAPs.
   - 6.4. Save outputs in dedicated folders with timestamps.

**Expected Deliverables**

- Python script: `pcap_analyzer.py`
- GUI-enabled tool for interactive analysis.
- Excel + HTML reports summarizing network anomalies and OWASP findings.
- Output tested with the provided `sample_traffic.pcap`.

## General Notes for Developer

- Maintain modular and well-documented code structure.
- Ensure GUI usability and error tolerance.
- Provide a `requirements.txt` file listing all dependencies.
- Each project should generate self-contained output directories.
- Submit test reports, screenshots, and sample outputs for review.