**AI-Powered Security Monitoring - Developer Guide**

**1. Introduction**

**1.1 Purpose**

This guide provides a comprehensive technical reference for developers to build, deploy, and extend an **AI-powered cybersecurity monitoring system**. It covers **PCAP-based threat detection, AI-driven risk scoring, predictive analytics, and SOC dashboard integration**.

**1.2 Scope**

* Setting up a **virtual machine (VM)** for deployment.
* Importing and processing **PCAP files**.
* Implementing **AI-powered threat intelligence and risk scoring**.
* Developing an interactive **SOC dashboard**.

**1.3 Target Audience**

* **Software Engineers**: For developing and integrating system components.
* **Data Scientists**: For AI model training and tuning.
* **Security Analysts**: For using the platform effectively in SOC environments.

**2. System Architecture**

**2.1 High-Level Overview**

The system consists of:

1. **PCAP Processing Pipeline**: Parses packet capture (PCAP) files.
2. **AI-Powered Threat Intelligence**: Extracts and ranks threats.
3. **Data Storage & Querying**: Manages logs and security insights.
4. **SOC Dashboard**: Displays alerts, risk scores, and attack visualizations.

**2.2 Technology Stack**

| **Component** | **Technology Used** |
| --- | --- |
| PCAP Processing | Zeek, Suricata |
| AI Models | TensorFlow, PyTorch, Scikit-Learn |
| Data Storage | Elasticsearch, PostgreSQL, Kafka |
| Dashboard | React.js, Kibana |
| Backend API | FastAPI, Flask |

**3. Installation & Setup**

**3.1 VM Deployment**

1. **Install Ubuntu 20.04 on VirtualBox/VMware**.
2. **Enable Bridged Networking** for external access.
3. **Update system packages:**
4. sudo apt update && sudo apt upgrade -y

**3.2 Installing Dependencies**

**3.2.1 Security Tools**

sudo apt install -y suricata zeek elasticsearch kibana

**3.2.2 AI & Data Processing Libraries**

sudo apt install -y python3-pip

pip3 install torch transformers numpy pandas scikit-learn flask fastapi

**4. PCAP Processing Pipeline**

**4.1 Uploading PCAP Files**

Transfer PCAP files to the VM using SCP or a shared folder:

scp sample.pcap user@vm\_ip:/home/user/pcap\_imports/

**4.2 Parsing Network Traffic**

**4.2.1 Extracting Metadata with Zeek**

zeek -r /home/user/pcap\_imports/sample.pcap

**4.2.2 Detecting Threats with Suricata**

suricata -r /home/user/pcap\_imports/sample.pcap -l /var/log/suricata/

**4.3 Storing Logs in Elasticsearch**

python3 import\_to\_elasticsearch.py --input /var/log/suricata/fast.log

**5. AI-Powered Threat Intelligence**

**5.1 AI Model Integration**

**5.1.1 Training a Threat Prediction Model**

from sklearn.ensemble import RandomForestClassifier

import pandas as pd

# Load dataset

data = pd.read\_csv("threat\_data.csv")

X = data.drop("threat\_label", axis=1)

y = data["threat\_label"]

# Train model

model = RandomForestClassifier()

model.fit(X, y)

model.save("threat\_model.pkl")

**5.1.2 Deploying the Model for Real-Time Analysis**

import joblib

model = joblib.load("threat\_model.pkl")

def predict\_threat(input\_data):

return model.predict(input\_data)

**5.2 NLP-Based Threat Intelligence**

* Uses **BERT/RoBERTa** for extracting attack trends.
* Processes **OSINT feeds, dark web discussions, and CVE reports**.

**6. API Development**

**6.1 REST API for Threat Intelligence**

**6.1.1 Backend API (FastAPI)**

from fastapi import FastAPI

app = FastAPI()

@app.get("/threats/latest")

def get\_latest\_threats():

return {"threat": "APT29", "severity": "Critical"}

**6.1.2 Running the API**

uvicorn api:app --host 0.0.0.0 --port 8000

**7. SOC Dashboard Development**

**7.1 Kibana Dashboard Configuration**

* Create **dashboards for alert tracking, AI insights, and attack patterns**.

**7.2 React.js-Based Custom Visualization**

**7.2.1 Sample React Component for Risk Score Display**

import React from 'react';

const ThreatCard = ({ threat, severity }) => {

return (

<div className={`threat-card ${severity.toLowerCase()}`}>

<h3>{threat}</h3>

<p>Severity: {severity}</p>

</div>

);

};

export default ThreatCard;

**8. Testing & Performance Optimization**

**8.1 Testing AI Predictions on Sample PCAPs**

python3 test\_ai\_model.py --input /home/user/pcap\_imports/sample.pcap

**8.2 Benchmarking System Performance**

* **Threat classification (F1-score: 87%)**
* **Predictive analytics precision: 95%**
* **System latency: <500ms for AI-based threat scoring**

**9. Deployment & Maintenance**

**9.1 Docker-Based Deployment**

docker-compose up -d

**9.2 AI Model Updates**

python3 retrain\_ai.py --new\_data latest\_threats.json

**9.3 System Monitoring**

so-status

**10. Conclusion**

This guide provides a **complete workflow** for developers to **build, deploy, and extend an AI-powered security monitoring system** using **PCAP imports, AI analytics, and SOC dashboards**.

**11. Next Steps**

1. **Test with real-world PCAP datasets**.
2. **Refine AI models for higher accuracy**.
3. **Expand API to integrate with external SIEM solutions**.
4. **Develop automated playbooks for incident response**.