InferSec: Explainable AI for Intelligent Threat Detection - Consolidated Report

Author: Manus AI **Date:** November 05, 2025 **Source Documents:** InferSec Project Reports (v2 & main), Infersec.docx, PNG Image (Architecture Diagram)

Abstract

InferSec is an **Explainable AI (XAI)** driven Intrusion Detection System (IDS) designed to detect network threats and provide human-interpretable explanations for each detection. The project integrates attack simulation (Kali Linux), data capture and preprocessing, machine learning (XGBoost, Random Forest, DNN), and explainability methods (SHAP, LIME, Captum) into a robust, end-to-end pipeline. The system operates in a hybrid environment, primarily utilizing **WSL2 (Ubuntu)** for development and analysis, a Kali VM for attack generation, and optionally Google Colab for GPU-accelerated training. This report consolidates all available documentation, detailing the project's objectives, architecture, technical stack, implementation steps, and future extensions.

1. Project Overview

Field	Detail
Project Name	InferSec
Full Form	Inference-driven Explainable Security System
Domains	Cybersecurity, Artificial Intelligence, Explainable AI (XAI)
Objective	Detect and interpret network intrusions using explainable ML models.
Deployment Context	Simulated attack/defense lab using Kali (attacks) + Ubuntu (XAI detection).
Nature	Real-time prototype (MVP) for research or Security Operations Center (SOC) use-case.
Tagline	An Explainable AI–powered Intrusion Detection System (XAI-IDS) built for transparency, trust, and interpretability in cybersecurity.

2. Core Objectives

The primary goals of the InferSec project are to:

1. **Build an isolated lab environment** for realistic attack and defense simulation.
2. **Generate labeled network traffic** (benign + malicious) for model training.
3. **Train ML and DL models** for intrusion detection and evaluate performance.
4. **Integrate Explainable AI techniques** (SHAP, LIME, Captum) to interpret and visualize model decisions.
5. **Develop an interactive dashboard** (Streamlit) for analysts to inspect predictions and feature attributions.
6. **Evaluate the models and explanations**, and prepare deliverables for academic submission.

3. System Architecture and Components

The system employs a **Dual-VM Hybrid Lab Setup**, optimized for a Windows host using WSL2.

3.1. Architecture Diagram

The conceptual architecture involves three main nodes:

Component	Role	Tools/Context
Kali Linux VM	Attack Source & Traffic Capture	nmap, hping3, tcpdump, Wireshark, Metasploit
Ubuntu (WSL2)	InferSec AI Node (Primary)	ML, SHAP, LIME, Streamlit Dashboard
Google Colab (Optional)	GPU-accelerated Training	PyTorch, extensive hyperparameter tuning

Network Topology: An isolated host-only network or WSL2 internal networking is used to contain experiments. Communication between the Kali VM and Ubuntu/WSL2 is achieved via shared folders or internal IP addresses (e.g., `172.24.x.x`).

3.2. Technical Stack

Layer	Tools/Frameworks
Language	Python 3.11
AI Frameworks	scikit-learn, PyTorch, XGBoost
XAI Frameworks	SHAP, LIME, Captum
Visualization	Streamlit, Plotly
Networking Tools	Zeek, tcpdump, nmap
Data Transformation	pandas, CICFlowMeter

4. Dataflow Pipeline

The project follows a seven-stage dataflow pipeline:

Stage	Description	Tools/Action
1. **Capture Traffic**	Kali generates attacks and captures traffic to `.pcap` files.	`sudo tcpdump -i eth0 -s 0 -w /shared/infersec_attack.pcap`
2. **Transfer PCAPs**	Move PCAPs to the Ubuntu/WSL2 environment.	Shared folder or `scp`
3. **Convert to Flows**	Convert PCAP files into flow-based CSVs, extracting features.	CICFlowMeter (Java) or pyshark scripts
4. **Preprocessing & Labeling**	Label flows (by attacker IP/time), clean data, handle missing values, normalize, and encode features.	pandas, custom scripts
5. **Model Training**	Train models on the processed data.	XGBoost, Random Forest, DNN (Colab)
6. **Explainability**	Apply XAI techniques to produce global and local explanations.	SHAP, LIME, Captum
7. **Visualization**	Load models and explanations into an interactive dashboard.	Streamlit

4.1. Feature Extraction

Key features computed from network flows include: `duration`, `src_bytes`, `dst_bytes`, `packet_count`, `pkt_rate`, `flags`, `protocol`, `src_port`, and `dst_port`.

4.2. Recommended Datasets

The system is designed to be dataset-agnostic, but the following are recommended:

Dataset	Size/Features	Attack Types	Suitability
UNSW-NB15	~2.5M records, 49 features	Fuzzers, Backdoors, DoS, Exploits, Worms (9 types)	**Main Training:** Modern, manageable size, good for XAI.
CICIDS2017	~2.8M instances, ~79 features	Botnet, Heartbleed, Brute Force, Infiltration, etc.	**Benchmark:** Rich features, realistic traffic, includes PCAPs.
KDD99	Classic, 41 features	Outdated attack types	**Baseline:** Lightweight, good for quick initial prototypes.

5. Machine Learning and Explainability

5.1. Model Choices

Category	Model	Rationale
CPU-Optimized	XGBoost (Primary)	High performance, robust, integrates well with SHAP.
	Random Forest	Robust, interpretable baseline.
	Logistic Regression	Lightweight sanity-check baseline.
Deep Learning	DNN (MLP), LSTM	Used for sequential flows; requires GPU (Colab).

5.2. Explainability Tools

Tool	Purpose	Integration
SHAP	Global and local feature attributions.	`shap.TreeExplainer` for tree models.
LIME	Local surrogate explanations for instance insights.	Provides model-agnostic local explanations.
Captum	Integrated Gradients, gradient-based attributions.	Used for PyTorch/DNN models.

6. Evaluation and Deliverables

6.1. Evaluation Metrics

Category	Metrics	Checks
Detection	Accuracy, Precision, Recall, F1-score, ROC-AUC, Confusion Matrix.	Evaluate misclassification patterns and per-class performance.
Explainability	Feature importance consistency (SHAP vs LIME), Fidelity, Sanity Checks.	Do explanations highlight intuitive attack features? Human analyst feedback (optional).
Robustness	Test on unseen synthetic attacks from Kali VM.	Optionally run adversarial perturbations to evaluate resilience.

6.2. Deliverables

The final project deliverables include:

- Source code repository (GitHub) with clear README.
- Processed datasets (CSV) and original PCAPs (sanitized).
- Trained model artifacts and explainability outputs (SHAP values, plots).
- Streamlit dashboard for demoing InferSec.
- Final PDF report, presentation slides, and optional demo video.

7. Future Extensions

The project is designed for future expansion, including:

- **LLM Integration:** Add LLM summarization for human-readable explanations of SHAP outputs ("why was this flagged?" in plain English).
- **Deployment:** Deploy as a Dockerized microservice for enterprise environments.
- **Real-time Processing:** Implement continual learning and real-time packet sniffing/stream-processing pipelines (Kafka, Flink).
- **SIEM Integration:** Integrate with Security Information and Event Management (SIEM) tools like Splunk, Wazuh, or ELK Stack.

8. GitHub Repository Layout

The suggested repository structure ensures modularity and clarity:

Plain Text

```
InferSec/
├── data/
│ ├── raw/          # PCAP files from Kali
│ ├── processed/    # Flow CSVs
│ └── models/       # Trained model binaries
├── notebooks/      # Jupyter notebooks for development and analysis
├── src/
│ ├── infersec_model.py
│ ├── explainers/   # SHAP and LIME implementations
│ └── utils/        # pcap_to_csv, data_loader, metrics
├── app/
│ └── dashboard.py  # Streamlit interface
├── docs/           # Architecture, setup guide, results
└── requirements.txt
```

9. Quick Commands and Snippets

Category	Command/Snippet	Purpose
Environment	`python3 -m venv ~/infersec/venv`	Create Python virtual environment.
	`source ~/infersec/venv/bin/activate`	Activate virtual environment.
Capture (Kali)	`sudo tcpdump -i eth0 -s 0 -w /shared/infersec_attack.pcap`	Capture all traffic on `eth0` to a PCAP file.
PCAP to Flows	`java -jar CICFlowMeter.jar -f /path/to/pcap -o flows.csv`	Convert PCAP to flow CSV using CICFlowMeter.
XGBoost Train	`model.fit(X_train, y_train)`	Sample training command.
SHAP Plot	`shap.summary_plot(shap_values, X_test)`	Generate global feature importance plot.
Dashboard	`streamlit run app/dashboard.py`	Run the Streamlit web application.

References

[1] InferSec Project Report v2.pdf [2] InferSec Project Report.pdf [3] Infersec.docx [4] Architecture Diagram (PNG Image)