POST-IMPLEMENTATION SECURITY REPORT

Subject: Distributed Intrusion Detection System using Snort, Kafka, and ELK
Date: 28th December 2025
Prepared By: Jyothi Samhitha Katragadda

Classification: Demonstration Document
Intended Audience: Security Analysts, SOC Engineers, Evaluators

---

1. EXECUTIVE SUMMARY

This project implements a **distributed network intrusion detection and monitoring system** using **Snort 3 sensors**, **Apache Kafka**, and the **Elastic Stack (Logstash, Elasticsearch, Kibana)**. The system is designed to monitor **multiple network segments** using independent sensors and forward alerts to a centralized analytics platform in real time.

Two Snort sensors were deployed on separate network interfaces, each monitoring a different segment. Alerts generated by Snort are forwarded through Kafka using custom Python forwarders, ingested by Logstash, indexed into Elasticsearch, and visualized using Kibana dashboards.

The project demonstrates:

- Multi-segment intrusion detection
- Decoupled alert ingestion using Kafka
- Centralized security visibility via ELK
- Practical SOC-style monitoring workflows

2. PROJECT METHODOLOGY

Phase I: Environment Setup

- Windows host system with VirtualBox
- WSL (Ubuntu) for Kafka, Logstash, Elasticsearch, Kibana
- Kali Linux virtual machines for sensors and traffic generation

Phase II: Snort Sensor Configuration

- Snort 3 installed on Kali Linux
- Custom ICMP detection rules added
- JSON alert output enabled

- Separate log directories per interface

Phase III: Kafka & Logstash Pipeline

- Apache Kafka deployed on WSL & topic created for Snort alerts
- Custom Python forwarder tails Snort alert files
- Logstash pipeline ingests Kafka messages into Elasticsearch

Phase IV: Multi-Segment Sensor Deployment

- Sensor 1: Host-only network (eth0)
- Sensor 2: Internal network (seg-b, eth1)
- Independent Snort processes and forwarders per interface

Phase V: Visualization & Validation

- Elasticsearch index patterns created
- Kibana dashboards built
- Live alert validation through traffic generation
- End-to-end alert flow verified

3. THREAT MODEL AND ASSUMPTIONS

Threats Simulated

- ICMP reconnaissance (ping sweeps)
- TCP SYN scanning
- Network probing activity

Assumptions

- Lab environment is trusted and controlled
- No encrypted payload inspection
- Alerts represent simulated attack behavior
- Sensors operate in passive IDS mode

Trust Boundaries

- Snort sensors generate alerts
- Kafka acts as a trusted transport layer
- ELK stack serves as centralized analysis plane

4. SYSTEM ARCHITECTURE

Core Components

- **Snort 3:** Network intrusion detection
- **Kafka:** Decoupled alert streaming
- **Logstash:** Data ingestion and transformation
- **Elasticsearch:** Alert indexing and storage
- **Kibana:** Visualization and analysis

Design Rationale

- Kafka prevents alert loss and allows scalability
- Independent sensors allow segment-specific monitoring
- Centralized ELK enables SOC-style analytics

The architecture supports **horizontal sensor scaling** and **real-time alert analysis**.

5. QUANTITATIVE ANALYSIS AND METRICS

Metric	**Observation**
Alerts Generated	ICMP and TCP events detected
Sensors Active	2
Network Segments	Host-only, Internal
Kafka Throughput	Real-time alert delivery
Elasticsearch Docs	Incremental growth verified
Visualization Latency	Near real-time

Document counts increased immediately upon traffic generation, validating the ingestion pipeline.

6. OBSERVATIONS AND TECHNICAL ANALYSIS

Sensor Behavior

- Host-only network generated consistent ICMP alerts
- Internal network traffic required external VM for visibility
- Self-ping traffic does not traverse interfaces

Pipeline Behavior

- Kafka successfully buffered and forwarded alerts
- Logstash downtime resulted in ingestion gaps
- Elasticsearch indexing confirmed via _cat/indices

Visualization Insights

- Clear alert peaks during active scans
- Protocol distribution dominated by ICMP
- Sensor activity validated dual-sensor operation

7. CHALLENGES FACED (STAR ANALOGY)

Situation

Initial alerts were not appearing in Kibana despite traffic generation.

Task

Identify whether the issue existed at Snort, Kafka, Logstash, or Elasticsearch.

Action

- Verified Snort logs directly
- Tested Kafka connectivity
- Checked Logstash service status
- Restarted broken ingestion pipeline

Result

Alert flow restored end-to-end with real-time visualization.

Other challenges included:

- Snort self-traffic misunderstanding
- Python package restrictions in Kali
- Permissions on Snort log files
- Kafka topic persistence confusion

8. GAP ANALYSIS: REMAINING RISKS AND MITIGATION

Gap	Risk	Mitigation
No TLS	Data exposure	Enable Kafka TLS
No Auth	Unauthorized access	Add SASL/Auth
Limited Rules	Missed attacks	Expand rule sets
Manual Forwarder	Maintenance	Use Filebeat

9. FURTHER IMPROVEMENTS AND ROADMAP

- Add Suricata comparison sensor
- Enable encrypted Kafka transport
- MITRE ATT&CK tagging automation
- Integrate Filebeat instead of Python
- Add anomaly-based detection models
- Long-term alert retention policies

10. MITRE ATT&CK MAPPING TABLE

Technique ID	Name	Observed
T1046	Network Service Scanning	Yes
T1018	Network Discovery	Yes
T1040	Network Sniffing	Simulated
T1595	Active Scanning	Yes

CONCLUSION

This project successfully demonstrates a **distributed intrusion detection architecture** using industry-relevant tools. The system validates real-time detection, scalable ingestion, and centralized security analytics across multiple network segments.

The implementation reflects **SOC-style operational workflows** and provides a strong foundation for further enterprise-grade security monitoring enhancements.