NETWORK INTRUSION DETECTION SYSTEM

PHASE I REPORT

Submitted By

HARISH B – 2019202016

in partial fulfillment for the award of the degree of

MASTER OF COMPUTER APPLICATION

COLLEGE OF ENGINEERING, GUINDY

DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY

ANNA UNIVERSITY, CHENNAI - 600025

APRIL, 2022

ANNA UNIVERSITY, CHENNAI

BONAFIDE CERTIFICATE

This Report titled "**NETWORK INTRUSION DETECTION SYSTEM**" is the bonafide work of **HARISH B (2019202016)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

DR. S. SRIDHAR
Professor and Head of the Department
Department of IST
Anna University
Chennai – 600 025

MS. B. SIVA SANKARI
Project Guide
Department of IST
Anna University
Chennai – 600 025

ABSTRACT

A machine learning-based Network Intrusion Detection System (ML-NIDS) that detects anomalies through Machine Learning algorithms by analyzing behaviors of packets. However, the Machine Learning based Network Intrusion Detection System learns the characteristics of attack traffic based on training data, so it is inevitably vulnerable to attacks that have not been learned, just like pattern-matching machine learning.The proposed approach can provide more robust and more accurate classification with the same classification datasets compared to existing approaches, so we expect it will be used as one of feasible solutions to overcome weakness and limitation of existing Machine Learning based Network Intrusion Detection System.

ACKNOWLEDGEMENT

The satisfaction that accompanies the success would be incomplete without mentioning the names of people who made it possible.

I would like to express my earnest thanks to **Ms. B. Siva Shankari**, Teaching Fellow, Department of Information Science and Technology, CEG, Anna University for her valuable guidance, encouragement and attitude that has driven the project work in a steady pace to a successful completion.

I would like to thank **Dr. S. Sridhar**, Professor and Head, Department of Information Science and Technology, CEG, Anna University for his kind support.

I express my sincere thanks to the project committee, **Dr. Saswati Mukherjee,** Professor, Department of Information Science and Technology, **Dr. M. Vijayalakshmi**, Associate Professor, Department of Information Science and Technology, **Dr. E. Uma,** Assistant Professor, Department of Information Science and Technology, **Ms. P. S. Apirajitha**, Teaching Fellow, Department of Information Science and Technology, **Ms. C. M. Sowmya**, Teaching Fellow, Department of Information Science and Technology, for their valuable suggestions that have led to the betterment of the project.

Harish B

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**iii**
	ACKNOWLEDGEMENT	**iv**
	LIST OF FIGURES	**v**
1	**INTRODUCTION**	**1**
	1.1 DOMAIN	1
	1.2 PROBLEM STATEMENT	1
	1.3 MOTIVATION & OBJECTIVE	1
	1.4 SOFTWARE REQUIREMENT	1
2	**REVIEW OF LITERATURE**	**2**
3	**DESIGN**	**4**
	3.1 ARCHITECTURE DIAGRAM	4
	3.2 FLOW CHART	5
	3.3 MODULES DESCRIPTION	6
4	**IMPLEMENTATION AND ALGORITHMS**	**7**
5	**OUTPUT AND SCREENSHOTS**	**8**
6	**REFERENCES**	**9**

LIST OF FIGURES

3.1	Network Intrusion Detection System	4
3.2	Flowchart of Network Intrusion Detection System	5
4.1	Block Diagram of Sniffing	7
4.2	Block Diagram of Streaming	7

CHAPTER 1

INTRODUCTION

1.1 DOMAIN OF PROJECT

Network Security:

Network Security is an emerging field in the IT-sector. As more devices are connected to the internet, the attack surface for hackers is steadily increasing. Network-based Intrusion Detection Systems (NIDS) can be used to detect malicious traffic in networks and Machine Learning is an up and coming approach for improving the detection rate. In this thesis the NIDS Zeek is used to extract features based on time and data size from network traffic. The features are then analyzed with Machine Learning in Scikit-learn in order to detect malicious traffic.

1.2 PROBLEM STATEMENT

Network Intrusion Detection Systems (NIDSs) using pattern matching have a fatal weakness in that they cannot detect new attacks because they only learn existing patterns and use them to detect those attacks.

1.3 MOTIVATION AND OBJECTIVE

The Objective of this project is to perform classification to improve detection of malicious traffic. Research which traffic features and machine learning algorithms that are suitable for detecting different kinds of malicious traffic. Write scripts to extract those features. Train machine learning models from a labeled dataset with malicious traffic. Evaluate the performance of the different models and scripts.

1.4 SOFTWARE REQUIREMENT

Development Platform: Linux

Language: Java, Python

Dataset: KDD-CUP-99

CHAPTER 2

LITERATURE STUDY

J. Alikhanov, R. Jang, M. Abuhamad, D. Mohaisen, D. Nyang and Y. Noh[1], Machine Learning (ML) based Network Intrusion Systems (NIDSs) operate on flow features which are obtained from flow exporting protocols. Recent success of ML and Deep Learning (DL) based NIDS solutions assume such flow information (*e.g.,* avg. packet size) is obtained from all packets of the flow. However, often in practice flow exporter is deployed on commodity devices where packet sampling is inevitable. As a result, applicability of such ML based NIDS solutions in the presence of sampling is an open question. In this study, we explore the impact of packet sampling on the performance and efficiency of ML-based NIDSs. Unlike previous work, our proposed evaluation procedure is immune to different settings of flow export stage. Hence, it can provide a robust evaluation of NIDS even in the presence of sampling. Through sampling experiments we established that malicious flows with shorter size are likely to go unnoticed even with mild sampling rates such as 1/10 and 1/100. Next, using the proposed evaluation procedure we investigated the impact of various sampling techniques on NIDS detection rate and false alarm rate. Detection rate and false alarm rate is computed for three sampling rates, for four different sampling techniques and for three classifiers.

T. Kim and W. Pak [2], proposed that the types of ML-NIDS are packet-based methods that use packet data directly as features, and session-based methods that use statistical data for a logical group called a session instead of packets as features.

The packet-based method can be classified in two ways: one detection method uses a single packet to detect a pattern for malicious data in every packet received, and the other detection method uses multiple packets, storing and combining packets belonging to the same session into one dataset that is used for detection. Both the single-packet detection method and the multi-packet detection method search for malicious code or patterns in the packet payloads. However, attacks exploiting normal packets, like a Distributed denial-of-service (DDoS), are hard to detect with the packet-based method, and the pattern-matching algorithm can easily be bypassed by adding random data to the payload.

When using session features, it is impossible to bypass the NIDS just by adding some dummy data. In addition, regardless of the packet size or the length of the session, the size of the entire feature is always the same, so the session-based method is more advantageous than the packet-based method for handling large volumes of traffic. The NIDS using session features mostly uses machine learning algorithms to classify the received traffic. So far, various ML-NIDSs have been developed and are expected to overcome the weaknesses of the PM-NIDS. Inevitably, malicious users are developing various methods to bypass the ML-NIDS (largely divided into white box, gray box, and black box methods), depending on what information can be used.

D. Han et al.[3] Machine learning (ML), especially deep learning (DL) techniques have been increasingly used in anomaly-based network intrusion detection systems (NIDS). Many adversarial attacks have been proposed to evaluate the robustness of ML-based NIDSs. Unfortunately, existing attacks mostly focused on feature-space and/or white-box attacks, which make impractical assumptions in real-world scenarios, leaving the study on practical gray/black-box attacks largely unexplored. To bridge this gap, we conduct the first systematic study of the gray/black-box traffic-space adversarial attacks to evaluate the robustness of ML-based NIDSs. Our work outperforms previous ones in the following aspects: (i) practical -the proposed attack can automatically mutate original traffic with extremely limited knowledge and affordable overhead while preserving its functionality; (ii)generic -the proposed attack is effective for evaluating the robustness of various NIDSs using diverse ML/DL models and non-payload-based features; (iii)explainable -we propose an explanation method for the fragile robustness of ML-based NIDSs. We extensively evaluate the robustness of various NIDSs using diverse feature sets and ML/DL models. Experimental results show our attack is effective with affordable execution cost and the proposed defense method can effectively mitigate such attacks.

CHAPTER 3

SYSTEM DESIGN AND ARCHITECTURE

3.1 ARCHITECTURE DIAGRAM

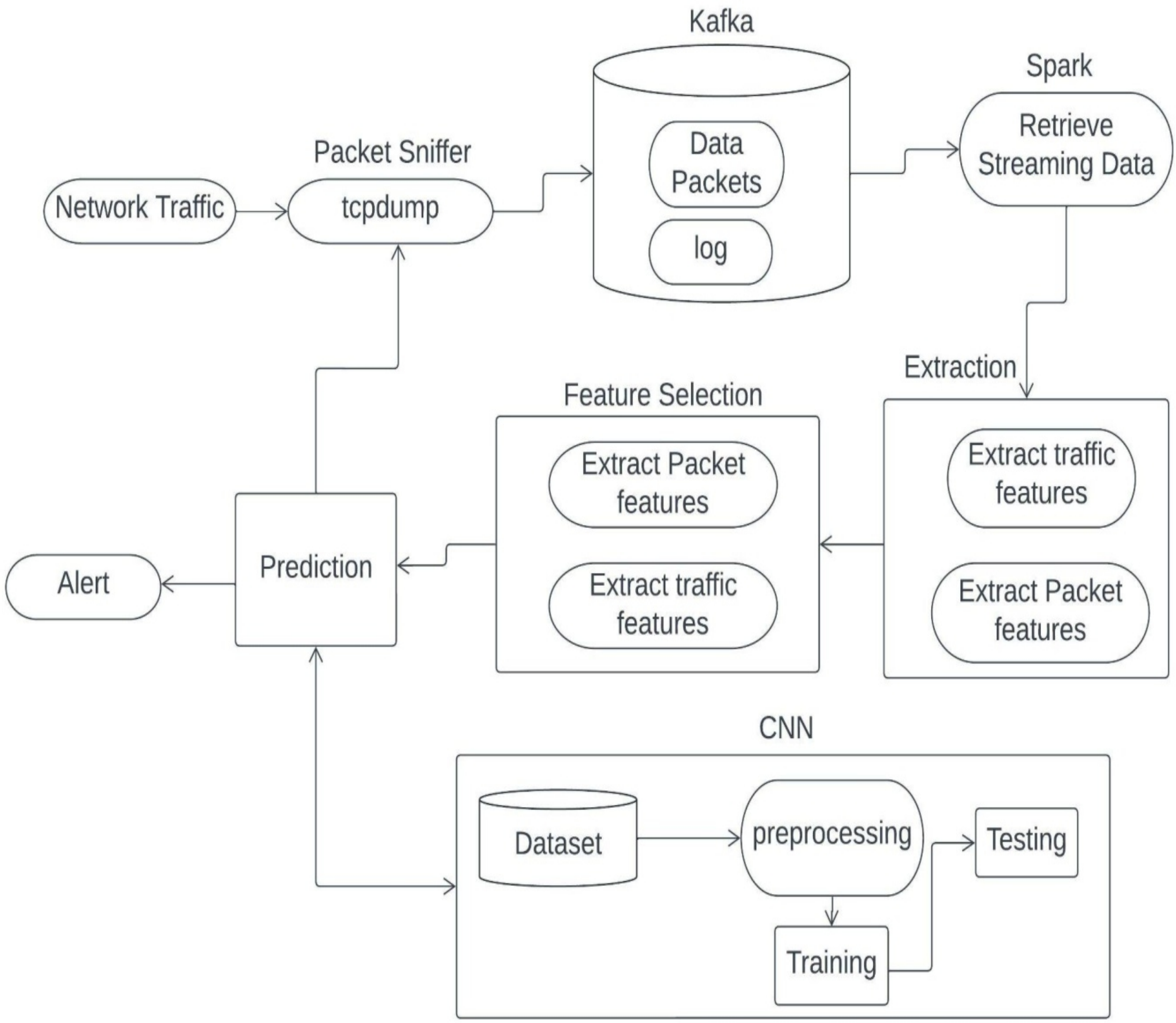


Fig 3.1 Network Intrusion Detection System

3.2 FLOW CHART

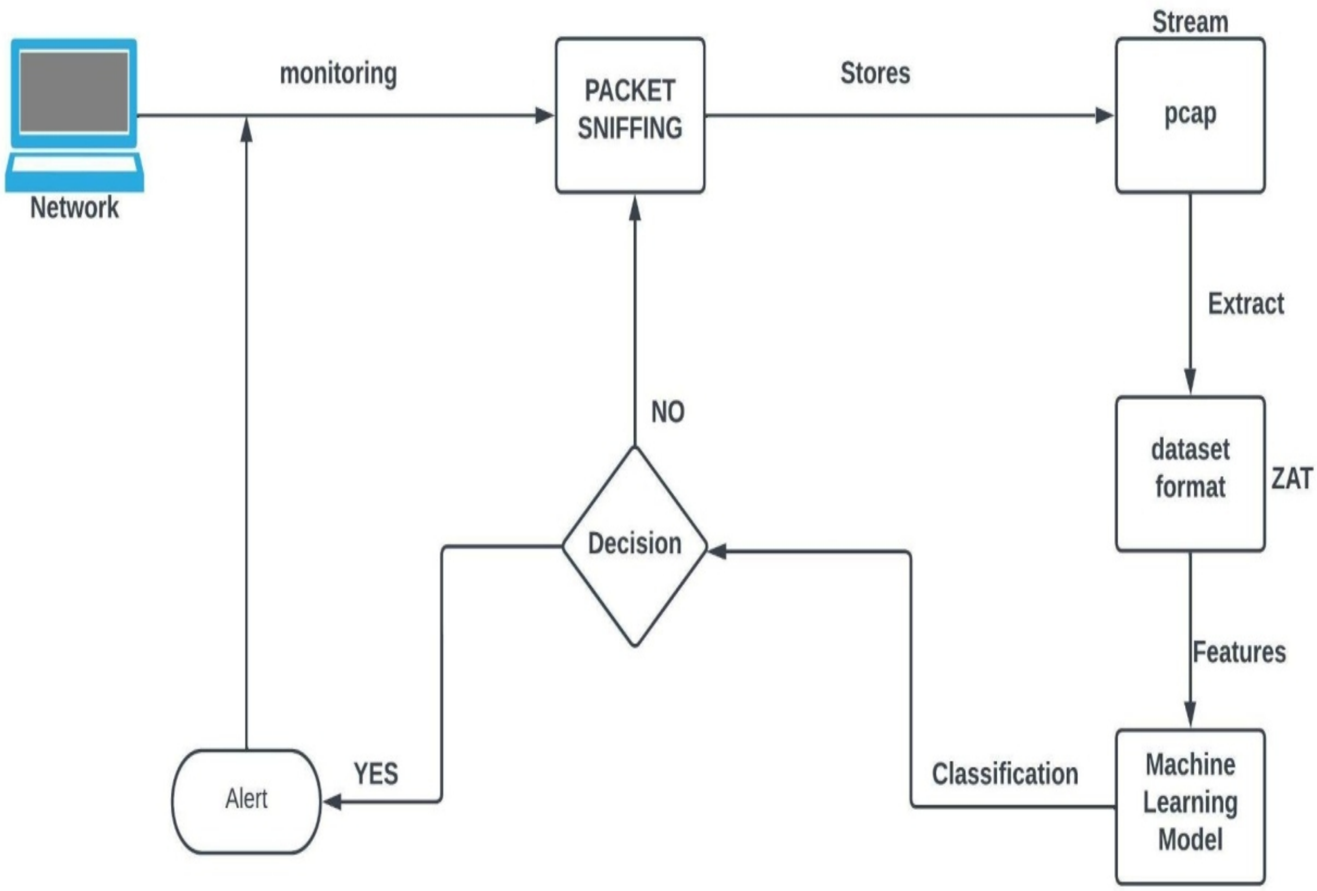


Fig 3.2 Flowchart of Network Intrusion Detection System

3.3 MODULES DESCRIPTION

There are 5 components in the system. They are

[1] Streaming
[2] Retrieve Data
[3] Feature Extraction and Selection
[4] Classification

3.3.1 Streaming

Kafka is used to create a producer which extracts packets from pcap and stream the packets one by one to the consumer. It is used for real time packet streaming without losing the packet.

3.3.2 Retrieve Data

Spark is used to retrieve data from a producer which is created by Kafka. Retrieve streaming data will be used to collect packet details for analysis. Packet details are stored in location for features analysis

3.3.3 Features Extraction and Selection

Features are extracted from packet details and stored as Dataset format with attributes. From the extracted features some features were selected for analysis purposes.

3.3.4 Classification

Here we use the concept of CNN for classification methods. The last module includes the classification in which Machine Learning algorithms will be used. Tensor Flow is a matlab-friendly open source library for numerical computation that makes machine learning faster and easier.

CHAPTER 4

IMPLEMENTATION AND ALGORITHMS

4.1 SNIFFING

It allows the user to display TCP/IP and other packets being transmitted or received over a network to which the computer is attached. This will store a pcap file which contains packets in it.

INPUT:

sudo tcpdump -i wlp2s0 -s0 -w sniff.pcap

OUTPUT:

Pcap file will be generated.

Figure 4.1 Block Diagram of Sniffing

4.2 STREAM

Data received in real time is referred to as streaming data, because it flows in as it is created. Zookeeper is initialized to start kafka and the topic is created, then the producer will stream the pcap file.

INPUT:

- bin/zookeeper-server-start.sh config/zookeeper.properties
- bin/kafka-server-start.sh config/server.properties
- bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic sniffer
- bin/kafka-console-producer.sh --bootstrap-server localhost:9092 --topic sniffer < ../sniff.pcap

OUTPUT:

Packets will streamed by producer

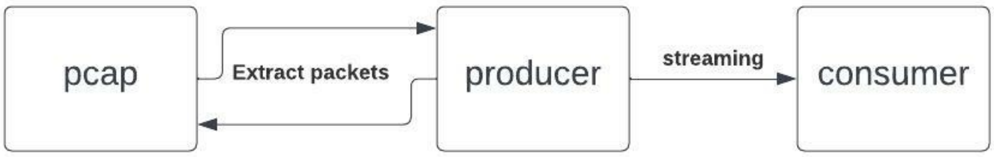
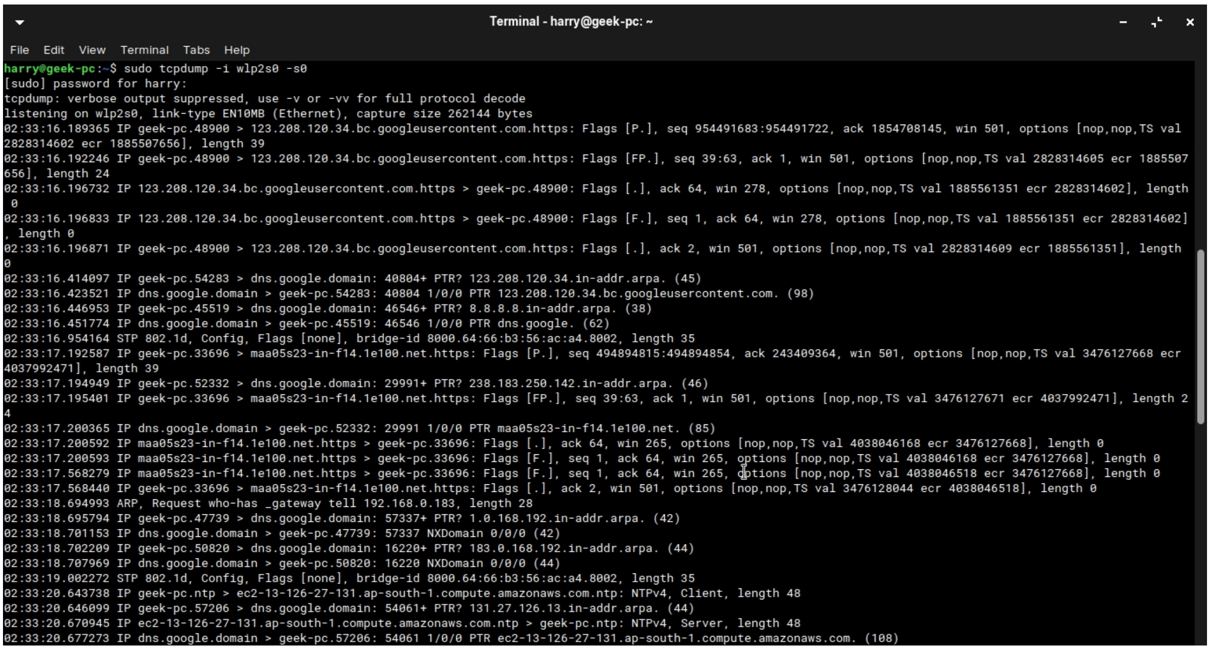


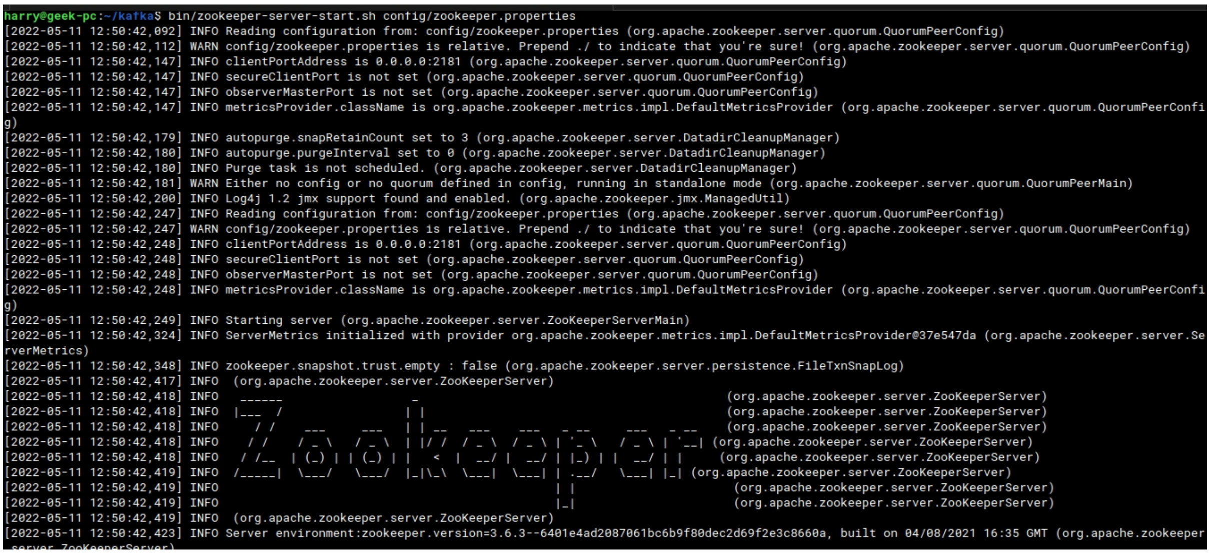
Figure 4.2 Block Diagram of Streaming

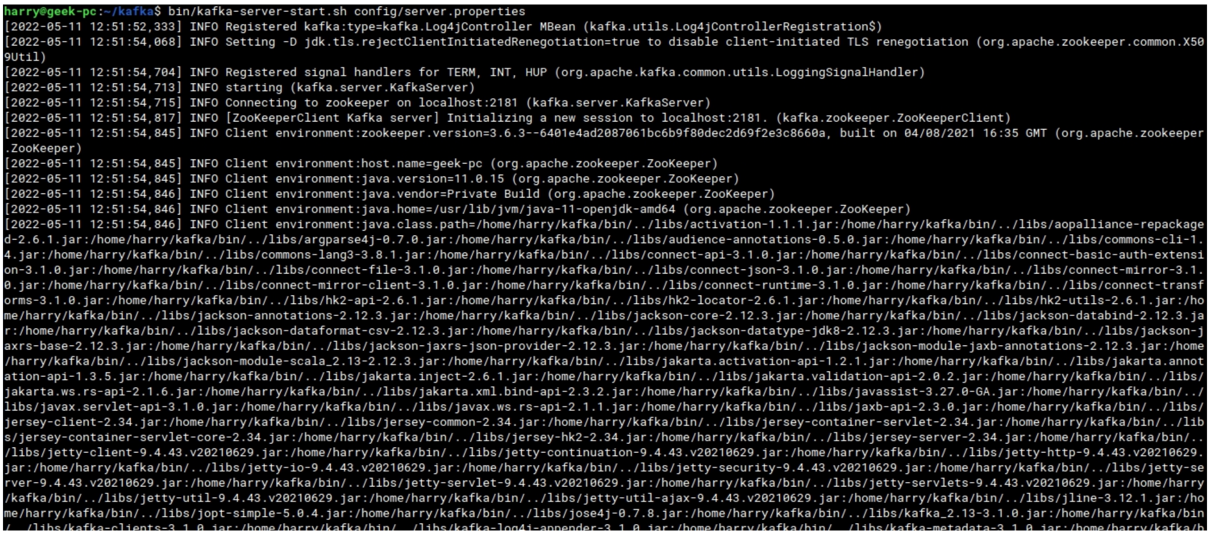
CHAPTER 5
OUTPUT AND SCREENSHOTS

5.1 Sniffing

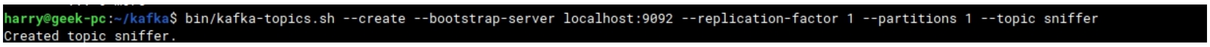


5.2 Streaming





```
harry@geek-pc:~/kafka$ bin/kafka-server-start.sh config/server.properties
[2022-05-11 12:51:52,333] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2022-05-11 12:51:54,068] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X50
9Util)
[2022-05-11 12:51:54,704] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.common.utils.LoggingSignalHandler)
[2022-05-11 12:51:54,713] INFO starting (kafka.server.KafkaServer)
[2022-05-11 12:51:54,715] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2022-05-11 12:51:54,817] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2022-05-11 12:51:54,845] INFO Client environment:zookeeper.version=3.6.3--6401e4ad2087061bc6b9f80dec2d69f2e3c8660a, built on 04/08/2021 16:35 GMT (org.apache.zookeeper
.ZooKeeper)
[2022-05-11 12:51:54,845] INFO Client environment:host.name=geek-pc (org.apache.zookeeper.ZooKeeper)
[2022-05-11 12:51:54,845] INFO Client environment:java.version=11.0.15 (org.apache.zookeeper.ZooKeeper)
[2022-05-11 12:51:54,846] INFO Client environment:java.vendor=Private Build (org.apache.zookeeper.ZooKeeper)
[2022-05-11 12:51:54,846] INFO Client environment:java.home=/usr/lib/jvm/java-11-openjdk-amd64 (org.apache.zookeeper.ZooKeeper)
[2022-05-11 12:51:54,846] INFO Client environment:java.class.path=/home/harry/kafka/bin/../libs/activation-1.1.1.jar:/home/harry/kafka/bin/../libs/aopalliance-repackage
d-2.6.1.jar:/home/harry/kafka/bin/../libs/argparse4j-0.7.0.jar:/home/harry/kafka/bin/../libs/audience-annotations-0.5.0.jar:/home/harry/kafka/bin/../libs/commons-cli-1.
4.jar:/home/harry/kafka/bin/../libs/commons-lang3-3.8.1.jar:/home/harry/kafka/bin/../libs/connect-api-3.1.0.jar:/home/harry/kafka/bin/../libs/connect-basic-auth-extensi
on-3.1.0.jar:/home/harry/kafka/bin/../libs/connect-file-3.1.0.jar:/home/harry/kafka/bin/../libs/connect-json-3.1.0.jar:/home/harry/kafka/bin/../libs/connect-mirror-3.1.
0.jar:/home/harry/kafka/bin/../libs/connect-mirror-client-3.1.0.jar:/home/harry/kafka/bin/../libs/connect-runtime-3.1.0.jar:/home/harry/kafka/bin/../libs/connect-transf
orms-3.1.0.jar:/home/harry/kafka/bin/../libs/hk2-api-2.6.1.jar:/home/harry/kafka/bin/../libs/hk2-locator-2.6.1.jar:/home/harry/kafka/bin/../libs/hk2-utils-2.6.1.jar:/ho
me/harry/kafka/bin/../libs/jackson-annotations-2.12.3.jar:/home/harry/kafka/bin/../libs/jackson-core-2.12.3.jar:/home/harry/kafka/bin/../libs/jackson-databind-2.12.3.ja
r:/home/harry/kafka/bin/../libs/jackson-dataformat-csv-2.12.3.jar:/home/harry/kafka/bin/../libs/jackson-datatype-jdk8-2.12.3.jar:/home/harry/kafka/bin/../libs/jackson-j
axrs-base-2.12.3.jar:/home/harry/kafka/bin/../libs/jackson-jaxrs-json-provider-2.12.3.jar:/home/harry/kafka/bin/../libs/jackson-module-jaxb-annotations-2.12.3.jar:/home
/harry/kafka/bin/../libs/jackson-module-scala_2.13-2.12.3.jar:/home/harry/kafka/bin/../libs/jakarta.activation-api-1.2.1.jar:/home/harry/kafka/bin/../libs/jakarta.annot
ation-api-1.3.5.jar:/home/harry/kafka/bin/../libs/jakarta.inject-2.6.1.jar:/home/harry/kafka/bin/../libs/jakarta.validation-api-2.0.2.jar:/home/harry/kafka/bin/../libs/
jakarta.ws.rs-api-2.1.6.jar:/home/harry/kafka/bin/../libs/jakarta.xml.bind-api-2.3.2.jar:/home/harry/kafka/bin/../libs/javassist-3.27.0-GA.jar:/home/harry/kafka/bin/../
libs/javax.servlet-api-3.1.0.jar:/home/harry/kafka/bin/../libs/javax.ws.rs-api-2.1.1.jar:/home/harry/kafka/bin/../libs/jaxb-api-2.3.0.jar:/home/harry/kafka/bin/../libs/
jersey-client-2.34.jar:/home/harry/kafka/bin/../libs/jersey-common-2.34.jar:/home/harry/kafka/bin/../libs/jersey-container-servlet-2.34.jar:/home/harry/kafka/bin/../lib
s/jersey-container-servlet-core-2.34.jar:/home/harry/kafka/bin/../libs/jersey-hk2-2.34.jar:/home/harry/kafka/bin/../libs/jersey-server-2.34.jar:/home/harry/kafka/bin/..
/libs/jetty-client-9.4.43.v20210629.jar:/home/harry/kafka/bin/../libs/jetty-continuation-9.4.43.v20210629.jar:/home/harry/kafka/bin/../libs/jetty-http-9.4.43.v20210629.
jar:/home/harry/kafka/bin/../libs/jetty-io-9.4.43.v20210629.jar:/home/harry/kafka/bin/../libs/jetty-security-9.4.43.v20210629.jar:/home/harry/kafka/bin/../libs/jetty-se
rver-9.4.43.v20210629.jar:/home/harry/kafka/bin/../libs/jetty-servlet-9.4.43.v20210629.jar:/home/harry/kafka/bin/../libs/jetty-servlets-9.4.43.v20210629.jar:/home/harry
/kafka/bin/../libs/jetty-util-9.4.43.v20210629.jar:/home/harry/kafka/bin/../libs/jetty-util-ajax-9.4.43.v20210629.jar:/home/harry/kafka/bin/../libs/jline-3.12.1.jar:/ho
me/harry/kafka/bin/../libs/jopt-simple-5.0.4.jar:/home/harry/kafka/bin/../libs/jose4j-0.7.8.jar:/home/harry/kafka/bin/../libs/kafka_2.13-3.1.0.jar:/home/harry/kafka/bin
```



```
harry@geek-pc:~/kafka$ bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic sniffer
Created topic sniffer.
```

REFERENCES

1. J. Alikhanov, R. Jang, M. Abuhamad, D. Mohaisen, D. Nyang and Y. Noh, "Investigating the Effect of Traffic Sampling on Machine Learning-Based Network Intrusion Detection Approaches," in *IEEE Access*, vol. 10, pp. 5801-5823, 2022

2. T. Kim and W. Pak, "Robust Network Intrusion Detection System Based on Machine-Learning With Early Classification," in IEEE Access, vol. 10, pp. 10754-10767, 2022.

3. D. Han et al., "Evaluating and Improving Adversarial Robustness of Machine Learning-Based Network Intrusion Detectors," in IEEE Journal on Selected Areas in Communications, vol. 39, no. 8, pp. 2632-2647, Aug. 2021.