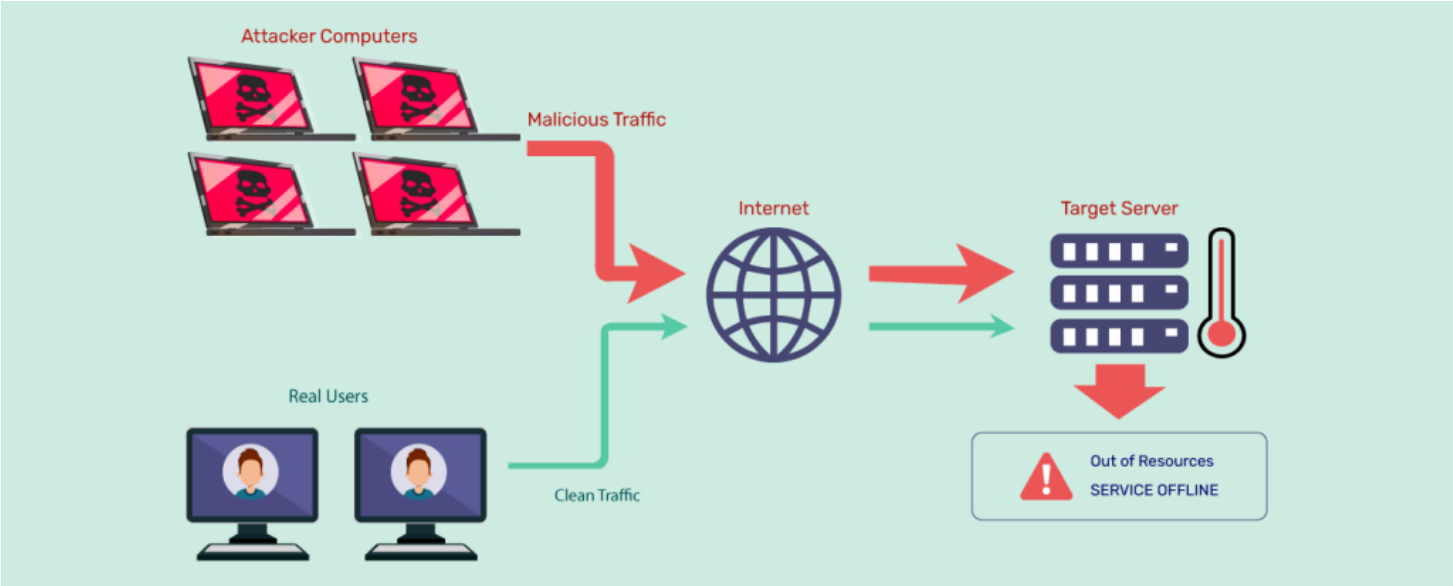
DDoS Attack Detection using Machine Learning in SDN



Contents

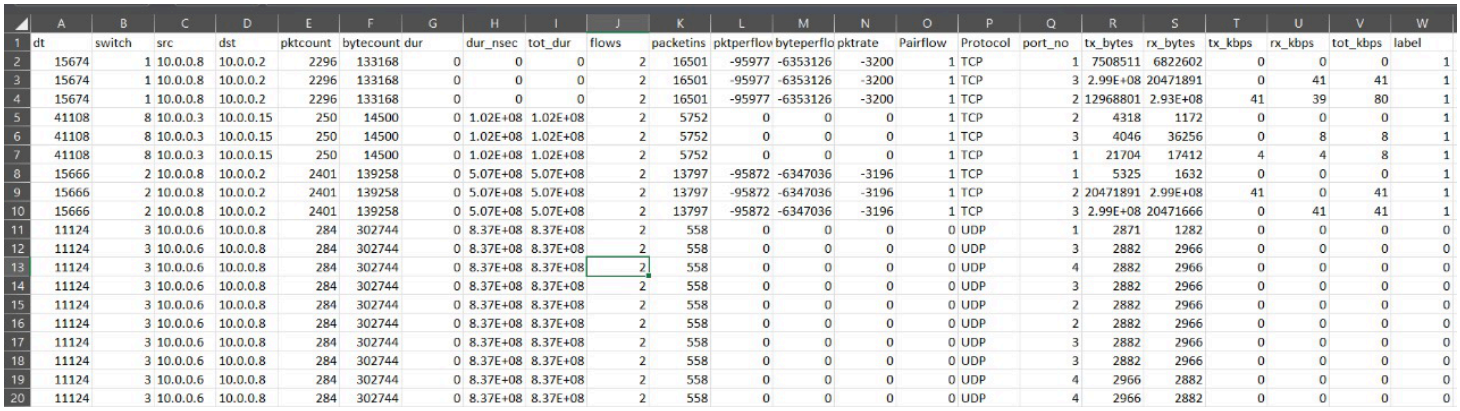
📋 Introduction

This document outlines the process of DDoS (Distributed Denial of Service) attack detection using Machine Learning (ML) techniques in Software-Defined Networking (SDN) environments.

The goal is to develop models capable of distinguishing between benign and malicious network traffic to enhance network security.

🌟 Data Analysis

Dataset Overview



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	dt	switch	src	dst	pktcount	bytecount	dur	dur_nsec	tot_dur	flows	packetins	pktperflow	byteperflo	pktrate	Pairflow	Protocol	port_no	tx_bytes	rx_bytes	tx_kbps	rx_kbps	tot_kbps	label
2	15674	1	10.0.0.8	10.0.0.2	2296	133168	0	0	0	2	16501	-95977	-6353126	-3200	1	TCP	1	7508511	6822602	0	0	0	1
3	15674	1	10.0.0.8	10.0.0.2	2296	133168	0	0	0	2	16501	-95977	-6353126	-3200	1	TCP	3	2.99E+08	20471891	0	41	41	1
4	15674	1	10.0.0.8	10.0.0.2	2296	133168	0	0	0	2	16501	-95977	-6353126	-3200	1	TCP	2	12968801	2.93E+08	41	39	80	1
5	41108	8	10.0.0.3	10.0.0.15	250	14500	0	1.02E+08	1.02E+08	2	5752	0	0	0	1	TCP	2	4318	1172	0	0	0	1
6	41108	8	10.0.0.3	10.0.0.15	250	14500	0	1.02E+08	1.02E+08	2	5752	0	0	0	1	TCP	3	4046	36256	0	8	8	1
7	41108	8	10.0.0.3	10.0.0.15	250	14500	0	1.02E+08	1.02E+08	2	5752	0	0	0	1	TCP	1	21704	17412	4	4	8	1
8	15666	2	10.0.0.8	10.0.0.2	2401	139258	0	5.07E+08	5.07E+08	2	13797	-95872	-6347036	-3196	1	TCP	1	5325	1632	0	0	0	1
9	15666	2	10.0.0.8	10.0.0.2	2401	139258	0	5.07E+08	5.07E+08	2	13797	-95872	-6347036	-3196	1	TCP	2	20471891	2.99E+08	41	0	41	1
10	15666	2	10.0.0.8	10.0.0.2	2401	139258	0	5.07E+08	5.07E+08	2	13797	-95872	-6347036	-3196	1	TCP	3	2.99E+08	20471666	0	41	41	1
11	11124	3	10.0.0.6	10.0.0.8	284	302744	0	8.37E+08	8.37E+08	2	558	0	0	0	0	UDP	1	2871	1282	0	0	0	0
12	11124	3	10.0.0.6	10.0.0.8	284	302744	0	8.37E+08	8.37E+08	2	558	0	0	0	0	UDP	3	2882	2966	0	0	0	0
13	11124	3	10.0.0.6	10.0.0.8	284	302744	0	8.37E+08	8.37E+08	2	558	0	0	0	0	UDP	4	2882	2966	0	0	0	0
14	11124	3	10.0.0.6	10.0.0.8	284	302744	0	8.37E+08	8.37E+08	2	558	0	0	0	0	UDP	3	2882	2966	0	0	0	0
15	11124	3	10.0.0.6	10.0.0.8	284	302744	0	8.37E+08	8.37E+08	2	558	0	0	0	0	UDP	2	2882	2966	0	0	0	0
16	11124	3	10.0.0.6	10.0.0.8	284	302744	0	8.37E+08	8.37E+08	2	558	0	0	0	0	UDP	2	2882	2966	0	0	0	0
17	11124	3	10.0.0.6	10.0.0.8	284	302744	0	8.37E+08	8.37E+08	2	558	0	0	0	0	UDP	3	2882	2966	0	0	0	0
18	11124	3	10.0.0.6	10.0.0.8	284	302744	0	8.37E+08	8.37E+08	2	558	0	0	0	0	UDP	3	2882	2966	0	0	0	0
19	11124	3	10.0.0.6	10.0.0.8	284	302744	0	8.37E+08	8.37E+08	2	558	0	0	0	0	UDP	4	2966	2882	0	0	0	0
20	11124	3	10.0.0.6	10.0.0.8	284	302744	0	8.37E+08	8.37E+08	2	558	0	0	0	0	UDP	4	2966	2882	0	0	0	0

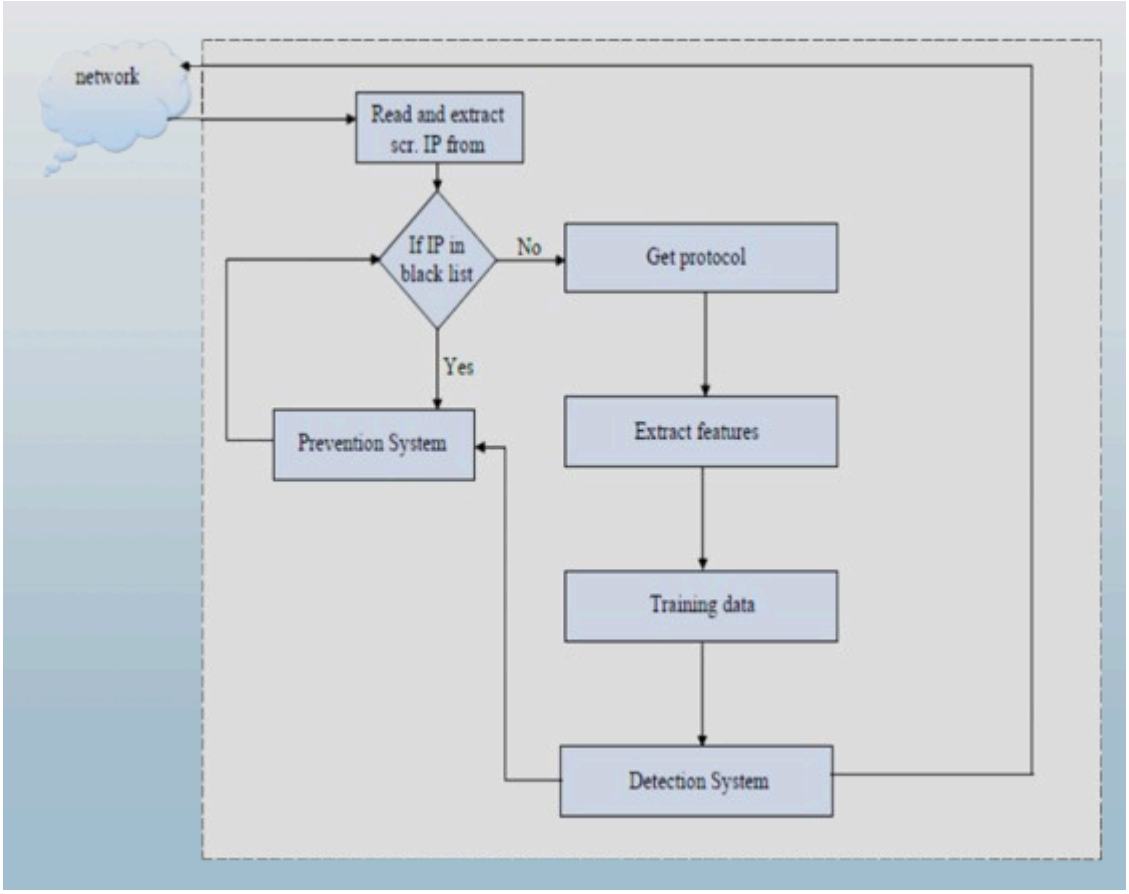
- The dataset contains information about network traffic, including features like packet count, byte count, protocol, duration, etc.
- The labels indicate whether the traffic is benign or malicious (0 for benign, 1 for malicious).

📋 Dataset Parameters

- `dt` : Timestamp of the event.
- `switch` : Switch ID.
- `src` : Source IP address.
- `dst` : Destination IP address.
- `pktcount` : Count of packets in the flow.
- `bytecount` : Count of bytes in the flow.
- `dur` : Duration of the flow in seconds.
- `dur nsec` : Duration of the flow in nanoseconds.
- `tot dur` : Total duration of the flow.
- `flows` : Number of flows.
- `packetins` : Count of packet insertions.
- `pktperflow` : Packets per flow.
- `byteperflow` : Bytes per flow.
- `pktrate` : Packet rate per second.
- `Pairflow` : Pair flow.
- `Protocol` : Protocol used in the flow (e.g., TCP, UDP).
- `port_no` : Port number.
- `tx_bytes` : Transmitted bytes.
- `rx_bytes` : Received bytes.
- `tx_kbps` : Transmitted kilobits per second.

- `rx_kbps` : Received kilobits per second.
- `tot_kbps` : Total kilobits per second.
- `label` : Label indicating the classification or outcome of the flow.

🌟 Proposed Algorithm



- `Capture Source IP` : Extract the source IP address from network traffic
- `Check Blacklist` :
  - If IP not in blacklist, proceed to identify the communication protocol.
  - If IP is blacklisted, take preventive actions (e.g., block it).
- `Feature Extraction` : Extract relevant features (e.g., packet size, ports) from network data
- `Train ML Model` : Use extracted features to train the machine learning model
- `Detection System` : Analyze incoming traffic using the trained model
- `Classification` : Classify traffic as normal or malicious

📋 Results

Model Implementation

- Implemented classical ML models including Logistic Regression, Support Vector Machine (SVM), Decision Tree, Random Forest, and k-Nearest Neighbors (KNN).
- Utilized feature scaling and preprocessing techniques for model training.
- Conducted hyperparameter tuning using GridSearchCV to optimize model performance.

Observed Results

- `All Features`

Algorithm	Accuracy	Precision	Recall	F1-Score
Logistic Regression	76.4%	0.73	0.80	0.78
Support Vector Machines (SVM)	97.0%	0.97	0.98	0.96
Random Forest	99.99%	1.00	1.00	1.00
K-Nearest Neighbors	98%	0.99	0.99	0.99

- `Selected Features`

Algorithm	Accuracy	Precision	Recall	F1-Score
Logistic Regression	75.21%	0.85	0.77	0.81
Support Vector Machines (SVM)	91.77%	0.90	0.93	0.94
Random Forest	99.42%	0.99	1.00	1.00
K-Nearest Neighbors	94.19%	0.91	1.00	0.93

- Logistic Regression, SVM, Decision Tree, Random Forest, and KNN models were trained and evaluated.
- Decision Tree and Random Forest exhibited promising performance in terms of accuracy and classification metrics.

✨ Conclusion

- ML models show promise in detecting DDoS attacks in SDN environments.
- Feature selection and preprocessing techniques play a crucial role in enhancing model performance.
- Decision Tree and Random Forest models demonstrate effectiveness in distinguishing between benign and malicious network traffic.
- Continued research and development in ML-based DDoS detection can contribute to strengthening network security in SDN infrastructures.

Group 18 Members:

- **Arihant Garg (21CS01033)**
- **Abeed Shaik (21CS01072)**
- **Priyam Saha (21CS01076)**

Link to GitHub Repository with codes : GitHub
