International Journal of Core Engineering & Management
Volume-8, Issue-01, 2025 ISSN No: 2348-9510

ANALYZING WIRESHARK DATA FOR MALICIOUS TRAFFIC USING AI

Britton Almy
University of Tennessee, Knoxville

Binoy Kurikaparambil Revi
University of Tennessee, Knoxville

Abstract

The objective of this project is to assess real time network traffic patterns to detect unusual patterns or activities that can be classified as threats. The project shall be split into two sections, one that deals with capturing the packets and the other that deals with learning the packets using machine learning algorithms and detecting the anomalies. We used simulation to simulate the malicious traffic for testing and verification purpose. Major on the shelf tools we used are well vetted tools like Wireshark and Flightsim for data capture and package simulation respectively. For the machine learning section of this project the program was developed in Python with the popular machine learning libraries like Keras and TensorFlow.

I. INTRODUCTION

In today's world, cyber-attacks are becoming as strong as defense systems build to protect users from these attacks. The fact is that in theory no algorithms or defense system in 100% secure. The moment industry considers an algorithm, or a system as secure, the other side starts spending effort to break it. Another very important aspect is that with the advancement of computing power like enhancement of powerful GPUs and in-memory databases, the computation has become quicker and improving as time progress. This gives the users and adversaries equal advantage to use these resources. Many algorithms used in the modern cryptographic world assumes that it takes more than a lifetime to break so called "secure to use" algorithm. However, adversaries can use the faster memory to enhance databases like rainbow tables to strategize the attack differently. So, considering all this from the user standpoint, network security seems to be first form of defense against any attacks. In the past most of the network security systems uses some or other form of static knowledge to design the software or hardware logic to enable defense system. However, as this knowledge is available to the adversaries, these strategies of network defense system start becoming week. As AI becomes prevelent it has also become the focus of security and attacks. [4]
In an enterprise security system in which the IT security professional who monitor the security if uses old system depends on basic log management tool to collect and search through logs, then the organization's network security can be considered at risk considering the modern cybersecurity and network attacks. These systems work by a set of rules and doesn't deviate unless the hardware or software is updated or upgraded. Well, then the question is, how about updating the hardware and software whenever there is a need? If we think logically, it is possible. However, practically it is not quite feasible. This calls for network outages, hardware, and software unknow issues during the update effectively making it a bad choice. So, what is the missing piece in the system? At this

point of time, it can be considered as “learning”. AI and Machine learning to the rescue!

We can think about Artificial Intelligence and Machine Learning techniques as techniques to learn network traffic and identifying threat or malicious activity on the network. As the models get mature, pre-trained model can be a good entity to design a new system.
The objective of this project is to assess real time network traffic patterns to detect unusual patterns or activities that can be classified as threats. The project shall be split into two sections, one that deals with capturing the packets and the other that deals with learning the packets using machine learning algorithms and detecting the anomalies.

For the capturing component of this project, we will use the network traffic analyzer software called Wireshark. For the analysis component we will write code to ingest the logs from Wireshark and look for malicious traffic. Because of time constraints we will pre-determine what traffic is being analyzed. To ensure that we can generate some non-threatening malicious sample traffic we will use an open source software available on Github called FlightSim.

We chose this project based on these criteria:

- This has a high potential to be a real-world application. [5]
- It is an application that can be used in various systems and scaled to use in various conditions.
- This will allow us to try different machine learning models available on this specific application.
- The application can be enhanced to have software li- braries to capture the package, UI integration and more.

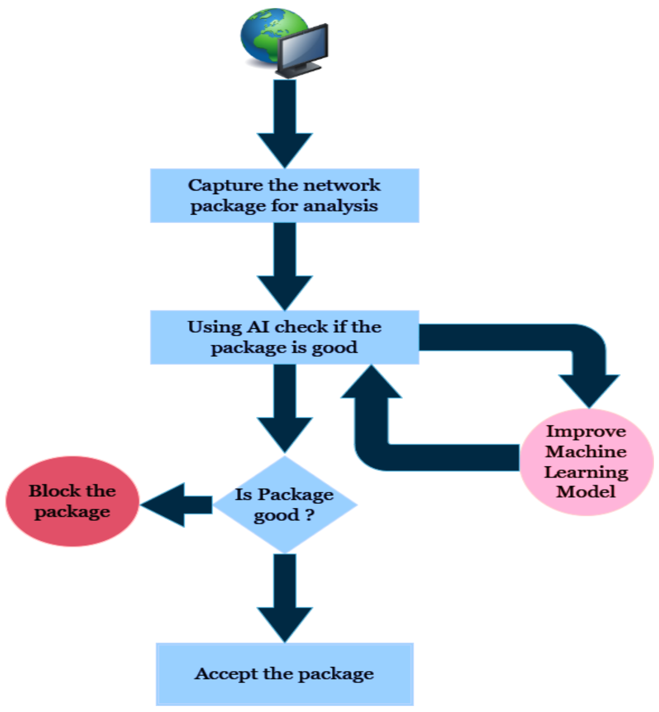

Capture the network package for analysis
Using AI check if the package is good
Improve Machine Learning Model
Block the package
Is Package good ?
Accept the package

Fig. 1. High Level Diagram

II. DATA PROCESSING TOOLS

1. **Wireshark:** Wireshark [1] is a powerful network proto- col analyzer widely used by network administrators, security professionals, and developers to examine the traffic flowing over a computer network. It allows users to capture, analyze, and interpret data packets in real-time or from saved files. Wireshark provides detailed insights into network communications, including the ability to identify and troubleshoot network issues, detect security threats, and monitor network performance. With its intuitive graphical interface and extensive filtering capabilities, Wireshark enables users to inspect packet headers, decode protocols, and extract valuable information from network traffic. Whether diagnosing network problems, investigating security incidents, or optimizing network performance, Wireshark serves as an indispensable tool for under- standing and managing computer networks effectively.

2. **Flightsim:** FlightSim [2], an open-source project de- veloped by AlphaSOC, it is a powerful tool designed to simulate real-world malicious network traffic for the purpose of training and testing network security solutions. FlightSim mimics various cyber threats, including malware infections, command-and-control communication, and data exfiltration. This software allows individuals to evaluate the effectiveness of intrusion detection systems (IDS), security information and event management (SIEM) platforms, and network traffic anal- ysis (NTA) tools in detecting and mitigating these simulated threats. It is easily deployed and customized and should allow us to generate specific traffic for identification.

III. AI COMPONENTS

An AI model designed to process traffic generated from Wireshark for the analysis of malicious traffic will incorporate various techniques from machine learning and cybersecurity. Below is a general outline of the model Please add the flow chart with all the AI components for better visualizations and reading.

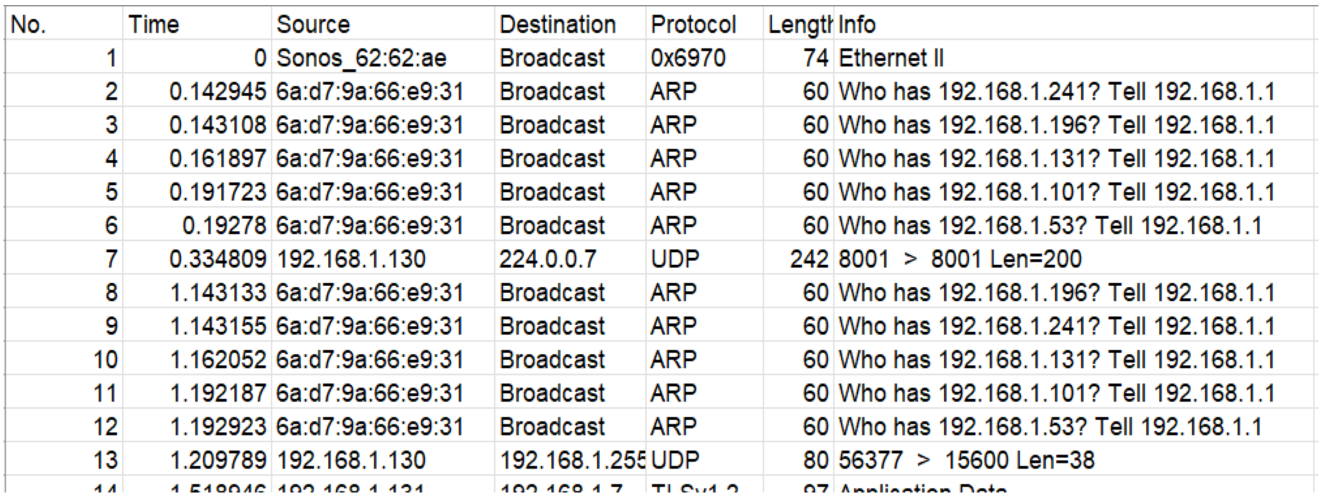
1. **Data Preprocessing:** The traffic data captured by Wire- shark will be preprocessed to extract relevant features for analysis. This preprocessing will involve converting raw packet data into a structured format suitable for input into machine learning algorithms. For this project, we have exported the data from the Wireshark to a csv file. The parameters captured to the csv file are:
   - Time
   - Source
   - Destination
   - Protocol
   - Length
   - Info

   The scan file is then used by the machine learning algorithm to extract the features and train the model.

2. **Feature Extraction:** Considering the profile of the ap- plication and studying the data, we considered supervised learning as a best path to go toward machine learning. This is because

the parameters are well defined, and system need to understand the label to classify the outcome to good package and bad package. We selected the Source and Destination parameters as features that can be used to generate a label field (True/False) that tells us if the package is malicious. The output of the feature extraction is source, destination, and labels. Let's call this as feature dataset. Features such as packet size, protocol type, source and destination IP addresses, ports, and packet payloads can also be extracted from the preprocessed data as features. Additional features may include temporal patterns, frequency of communication, and anomalies in traffic behavior.

3. **Training Data**: Using the feature dataset from the ex- traction, training dataset is created. The complete dataset is split into training and validation dataset. Idea here is to use the training data to train the model and use the validation dataset to evaluate the model.



No.	Time	Source	Destination	Protocol	Lengtł	Info
1	0	Sonos_62:62:ae	Broadcast	0x6970	74	Ethernet II
2	0.142945	6a:d7:9a:66:e9:31	Broadcast	ARP	60	Who has 192.168.1.241? Tell 192.168.1.1
3	0.143108	6a:d7:9a:66:e9:31	Broadcast	ARP	60	Who has 192.168.1.196? Tell 192.168.1.1
4	0.161897	6a:d7:9a:66:e9:31	Broadcast	ARP	60	Who has 192.168.1.131? Tell 192.168.1.1
5	0.191723	6a:d7:9a:66:e9:31	Broadcast	ARP	60	Who has 192.168.1.101? Tell 192.168.1.1
6	0.19278	6a:d7:9a:66:e9:31	Broadcast	ARP	60	Who has 192.168.1.53? Tell 192.168.1.1
7	0.334809	192.168.1.130	224.0.0.7	UDP	242	8001 > 8001 Len=200
8	1.143133	6a:d7:9a:66:e9:31	Broadcast	ARP	60	Who has 192.168.1.196? Tell 192.168.1.1
9	1.143155	6a:d7:9a:66:e9:31	Broadcast	ARP	60	Who has 192.168.1.241? Tell 192.168.1.1
10	1.162052	6a:d7:9a:66:e9:31	Broadcast	ARP	60	Who has 192.168.1.131? Tell 192.168.1.1
11	1.192187	6a:d7:9a:66:e9:31	Broadcast	ARP	60	Who has 192.168.1.101? Tell 192.168.1.1
12	1.192923	6a:d7:9a:66:e9:31	Broadcast	ARP	60	Who has 192.168.1.53? Tell 192.168.1.1
13	1.209789	192.168.1.130	192.168.1.255	UDP	80	56377 > 15600 Len=38
14	1.518946	192.168.1.131	192.168.1.7	TLSv1.2	97	Application Data

Fig. 2. Sample Scan Data

4. **Machine Learning Algorithm:** Various machine learning algorithms will be employed for the detection of the malicious traffic patterns. Possible techniques such as Support Vector Machines (SVM), Random Forests, or Deep Learning models like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs). With the research done so far, we picked Support Vector Machine (SVM) for this project. A support Vector Machine is supervised machine learning model used to solve complex classification problems by performing data transformations that determine boundaries between data points based on predefined classes. Technically SVM identifies a clear hyperplane that distinguishably separate data points of different class. SVMs are designed for binary classification problem. But by combining several binary classifiers SVM can be used to handle complex multiclass problems.

5. **Anomaly Detection:** In addition to pattern recognition, the AI model may incorporate anomaly detection techniques to identify deviations from normal traffic behavior. This could involve building a profile of typical network behavior and flagging any deviations from this baseline as potentially malicious.

6. **Real-Time Analysis:** We understand that to be effective in a practical setting, the AI model

would need to be capable of real-time analysis, quickly identifying and responding to potential threats as they occur due to time limitations for the project we may not move to this phase.

IV. METHOD

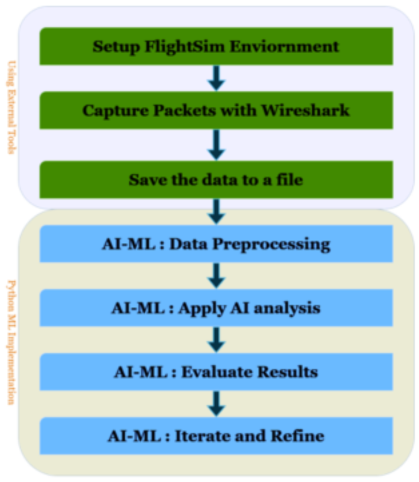

Using External Tools
Setup FlightSim Enviornment
Capture Packets with Wireshark
Save the data to a file
Python ML Implementation
AI-ML : Data Preprocessing
AI-ML : Apply AI analysis
AI-ML : Evaluate Results
AI-ML : Iterate and Refine

Fig. 3. Meathod Flowchart

1) **Setup and run the FlightSim Environment** - We begin by setting up the FlightSim environment to generate simulated malicious traffic. We will select two to three types of traffic for simulation.
2) **Capture Packets with Wireshark** - We will use Wire- shark to capture the network packets generated by Flight- Sim.
3) **Save Wireshark Logs** - The captured packets from Wireshark will be saved in a log file format such as PCAP (Packet Capture) for further analysis. The log file will contain comprehensive information about the captured traffic, including packet headers, payloads, and any other relevant metadata.
4) **Preprocess Wireshark Logs** - We will preprocess the Wireshark logs to extract features relevant for analysis. This may involve parsing the captured packets to extract information such as source and destination IP addresses, port numbers, packet sizes, protocol types, and times- tamps.
5) **Apply AI Analysis** - We will then utilize an AI model trained to analyze network traffic for malicious patterns. The preprocessed Wireshark logs will be fed into the AI model for analysis. The AI model should be capable of identifying anomalous patterns, malicious payloads, or known signatures of cyber threats.
6) **Detect Malicious Traffic** - Based on the output of the AI analysis, we hope to identify and classify the traffic captured by Wireshark as either benign or malicious. The AI model should provide insights into the nature of the detected threats, enabling further investigation or mitigation actions.
7) **Evaluate Results** - We eEvaluate the performance of the AI model in detecting malicious traffic based on the Wireshark logs by assessing the accuracy, precision, re- call, and other relevant metrics to gauge the effectiveness of the AI-driven analysis approach.
8) **Iterate and Refine** - Finally, we will iterate on the method by refining the AI model, adjusting

International Journal of Core Engineering & Management
Volume-8, Issue-01, 2025 ISSN No: 2348-9510

feature extraction techniques, or experimenting with different scenarios in FlightSim to improve the detection capabilities and overall accuracy of the analysis. [?]

V. EXECUTION

A. FlightSim Simulation

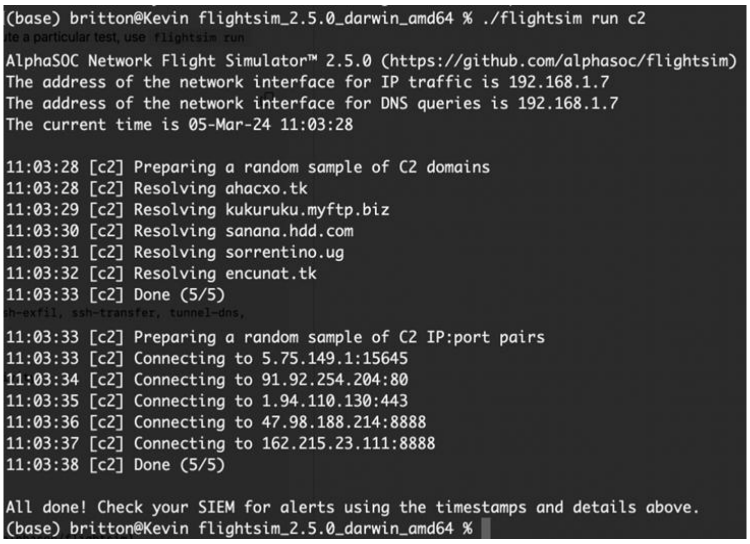
FlightSim was used to generate C2 sample traffic while Wireshark captured all network traffic. This traffic was saved to a log and the addresses that were generated for the C2 activity by FlightSim were saved to another file as malicious IP addresses. FlightSim generated traffic to five random websites,

1) ahacxo.tk
2) kukuruku.myftp.biz
3) sanana.hdd.com
4) sorrentino.ug
5) encunat.tk

It then created a random port sampling for the resolved IP addresses of these sites,

1) 5.75.149.1:15645
2) 91.92.254.204:80
3) 1.94.110.130:443
4) 47.98.188.214:8888
5) 162.215.23.111:8888

FlightSim then connected to the IP and port combination to simulate malicious traffic. While FlightSim was working Wireshark was monitoring all network traffic to capture the packets and save them to a log.



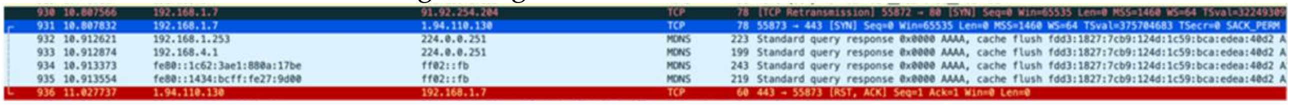
```
(base) britton@Kevin flightsim_2.5.0_darwin_amd64 % ./flightsim run c2
te a particular test, use flightsim run

AlphaSOC Network Flight Simulator™ 2.5.0 (https://github.com/alphasoc/flightsim)
The address of the network interface for IP traffic is 192.168.1.7
The address of the network interface for DNS queries is 192.168.1.7
The current time is 05-Mar-24 11:03:28

11:03:28 [c2] Preparing a random sample of C2 domains
11:03:28 [c2] Resolving ahacxo.tk
11:03:29 [c2] Resolving kukuruku.myftp.biz
11:03:30 [c2] Resolving sanana.hdd.com
11:03:31 [c2] Resolving sorrentino.ug
11:03:32 [c2] Resolving encunat.tk
11:03:33 [c2] Done (5/5)
h-exfil, ssh-transfer, tunnel-dns,
11:03:33 [c2] Preparing a random sample of C2 IP:port pairs
11:03:33 [c2] Connecting to 5.75.149.1:15645
11:03:34 [c2] Connecting to 91.92.254.204:80
11:03:35 [c2] Connecting to 1.94.110.130:443
11:03:36 [c2] Connecting to 47.98.188.214:8888
11:03:37 [c2] Connecting to 162.215.23.111:8888
11:03:38 [c2] Done (5/5)

All done! Check your SIEM for alerts using the timestamps and details above.
(base) britton@Kevin flightsim_2.5.0_darwin_amd64 %
```

Fig. 4. FlightSim C2 Traffic Screen Shot



```
930 10.807566  192.168.1.7                  91.92.254.204  TCP    78 [TCP Retransmission] 55872 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=32249309
931 10.807832  192.168.1.7                  1.94.110.130   TCP    78 55873 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=375704683 TSecr=0 SACK_PERM
932 10.912621  192.168.1.253                224.0.0.251    MDNS  223 Standard query response 0x0000 AAAA, cache flush fdd3:1827:7cb9:124d:1c59:bca:edea:40d2 A
933 10.912874  192.168.4.1                  224.0.0.251    MDNS  199 Standard query response 0x0000 AAAA, cache flush fdd3:1827:7cb9:124d:1c59:bca:edea:40d2 A
934 10.913373  fe80::1c62:3ae1:880a:17be    ff02::fb       MDNS  243 Standard query response 0x0000 AAAA, cache flush fdd3:1827:7cb9:124d:1c59:bca:edea:40d2 A
935 10.913554  fe80::1434:bcff:fe27:9d00    ff02::fb       MDNS  219 Standard query response 0x0000 AAAA, cache flush fdd3:1827:7cb9:124d:1c59:bca:edea:40d2 A
936 11.027737  1.94.110.130                 192.168.1.7    TCP    60 443 → 55873 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
```

Fig. 5. Wireshark Screen Shot

VI. IMPLEMENTATION DETAILS

The code is written in python using Jupyter notebook. To run the code, two input files are required. First file is the scan csv file from flightsim and the second is the text file that has the malicious IP information for labeling. Following packages are used from the python library.

- pandas
- requests
- tensorflow
- sklearn
- numpy

A. Section 1: Import data from the CSV File

The following section of the code import the csv file that is generated by the flightsim software.

B. Section 2: Data Pre-Processing

During the data pre-processing part, we read the labels from the maliciousips.txt. This is nothing but a lookup table that has all the malicious Ip information that can be used to label the data extracted. Also in this section, a new column is added to dataset to represent the value of the label. If the label is false, that mark that row as malicious. If it is true, it is considered as good. For this project we also made an assumption about the port numbers we are looking in to.

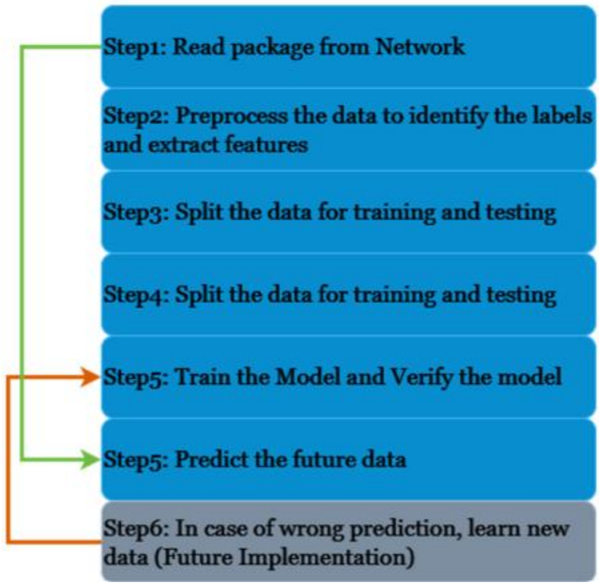

Step1: Read package from Network
Step2: Preprocess the data to identify the labels and extract features
Step3: Split the data for training and testing
Step4: Split the data for training and testing
Step5: Train the Model and Verify the model
Step5: Predict the future data
Step6: In case of wrong prediction, learn new data (Future Implementation)

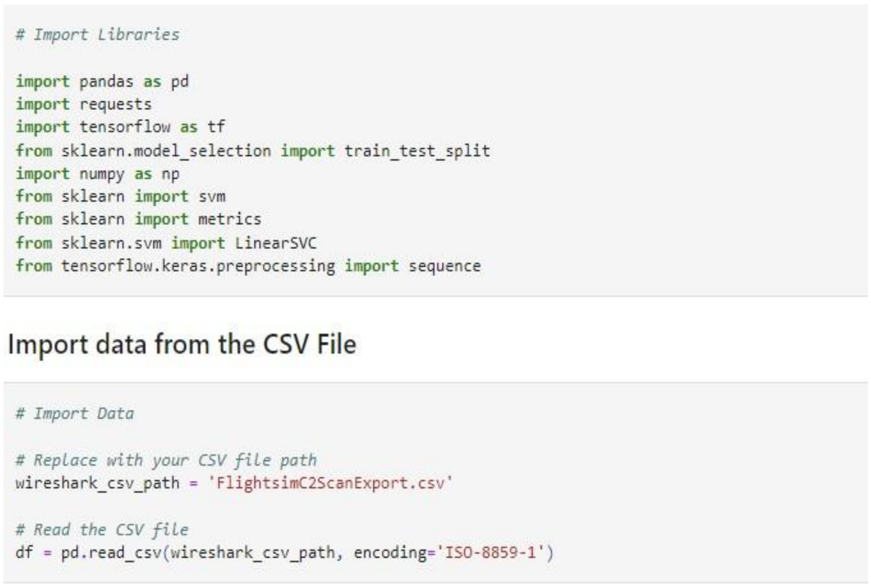
Fig. 6. Algorithm

C. Section 3: Create test and Train Datasets

Next step in the data processing is to create a training and test dataset. For this purpose we split the entire dataset into 2. First set contains randomly selected 70There was some extra work we did to these test and training datasets to get it ready. This is basically transforming training features which are in human readable strings format to token. We use the python library Tokenizer to convert the strings to tokens. The tokens are basically a list of numbers. However these are not uniform across all tokens. So we have to add zeros at the start to make sure all the lists are of equal

International Journal of Core Engineering & Management
Volume-8, Issue-01, 2025 ISSN No: 2348-9510

length. Now datasets are ready for training and verification. The code to create test and train datasets is given below.



```
# Import Libraries

import pandas as pd
import requests
import tensorflow as tf
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn import svm
from sklearn import metrics
from sklearn.svm import LinearSVC
from tensorflow.keras.preprocessing import sequence
```

Import data from the CSV File

```
# Import Data

# Replace with your CSV file path
wireshark_csv_path = 'FlightsimC2ScanExport.csv'

# Read the CSV file
df = pd.read_csv(wireshark_csv_path, encoding='ISO-8859-1')
```

Fig. 7. Data Preprocessing

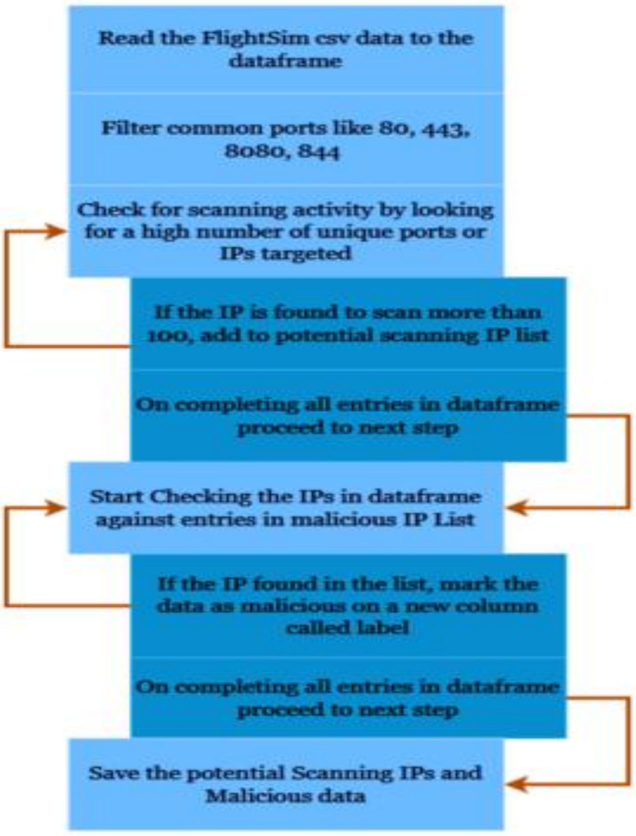
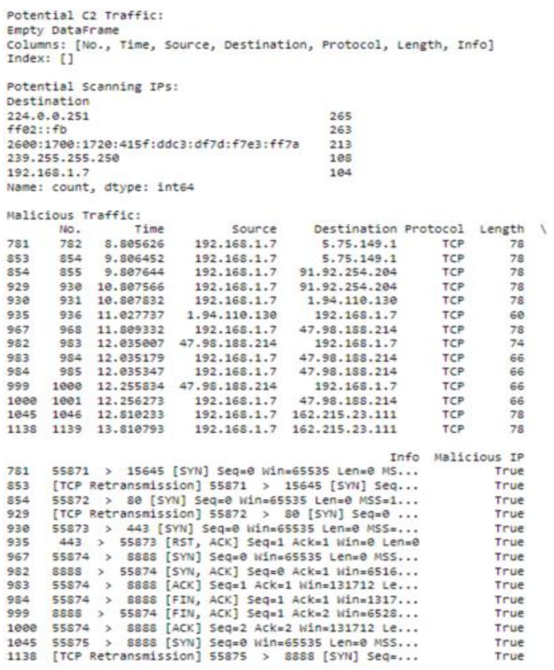

Read the FlightSim csv data to the dataframe
Filter common ports like 80, 443, 8080, 844
Check for scanning activity by looking for a high number of unique ports or IPs targeted
If the IP is found to scan more than 100, add to potential scanning IP list
On completing all entries in dataframe proceed to next step
Start Checking the IPs in dataframe against entries in malicious IP List
If the IP found in the list, mark the data as malicious on a new column called label
On completing all entries in dataframe proceed to next step
Save the potential Scanning IPs and Malicious data

Fig. 8. Data Preprocessing Algorithm



```
Potential C2 Traffic:
Empty DataFrame
Columns: [No., Time, Source, Destination, Protocol, Length, Info]
Index: []

Potential Scanning IPs:
Destination
224.0.0.251                                 265
ff02::fb                                    263
2600:1700:1720:415f:ddc3:df7d:f7e3:ff7a     213
239.255.255.250                             108
192.168.1.7                                 104
Name: count, dtype: int64

Malicious Traffic:
       No.       Time          Source     Destination Protocol  Length  \
781    782   8.805626     192.168.1.7      5.75.149.1      TCP      78
853    854   9.806452     192.168.1.7      5.75.149.1      TCP      78
854    855   9.807644     192.168.1.7   91.92.254.204      TCP      78
929    930  10.807566     192.168.1.7   91.92.254.204      TCP      78
930    931  10.807832     192.168.1.7    1.94.110.130      TCP      78
935    936  11.027737    1.94.110.130     192.168.1.7      TCP      60
967    968  11.809332     192.168.1.7   47.98.188.214      TCP      78
982    983  12.035007   47.98.188.214     192.168.1.7      TCP      74
983    984  12.035179     192.168.1.7   47.98.188.214      TCP      66
984    985  12.035347     192.168.1.7   47.98.188.214      TCP      66
999   1000  12.255834   47.98.188.214     192.168.1.7      TCP      66
1000  1001  12.256273     192.168.1.7   47.98.188.214      TCP      66
1045  1046  12.810233     192.168.1.7  162.215.23.111      TCP      78
1138  1139  13.810793     192.168.1.7  162.215.23.111      TCP      78

                                                   Info  Malicious IP
781   55871  >  15645 [SYN] Seq=0 Win=65535 Len=0 MS...          True
853   [TCP Retransmission] 55871  >  15645 [SYN] Seq...          True
854   55872  >  80 [SYN] Seq=0 Win=65535 Len=0 MSS=1...          True
929   [TCP Retransmission] 55872  >  80 [SYN] Seq=0 ...          True
930   55873  >  443 [SYN] Seq=0 Win=65535 Len=0 MSS=...          True
935    443  >  55873 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0          True
967   55874  >  8888 [SYN] Seq=0 Win=65535 Len=0 MSS...          True
982   8888  >  55874 [SYN, ACK] Seq=0 Ack=1 Win=6516...          True
983   55874  >  8888 [ACK] Seq=1 Ack=1 Win=131712 Le...          True
984   55874  >  8888 [FIN, ACK] Seq=1 Ack=1 Win=1317...          True
999   8888  >  55874 [FIN, ACK] Seq=1 Ack=2 Win=6528...          True
1000  55874  >  8888 [ACK] Seq=2 Ack=2 Win=131712 Le...          True
1045  55875  >  8888 [SYN] Seq=0 Win=65535 Len=0 MSS...          True
1138  [TCP Retransmission] 55875  >  8888 [SYN] Seq=...          True
```

Fig. 9. Output of data prepossessing

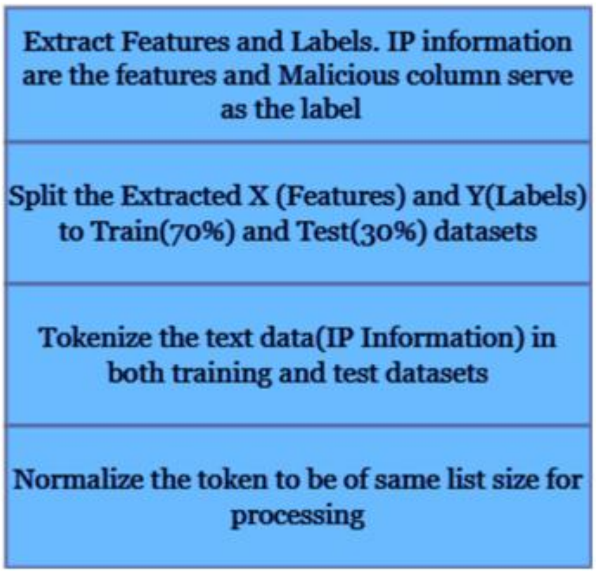
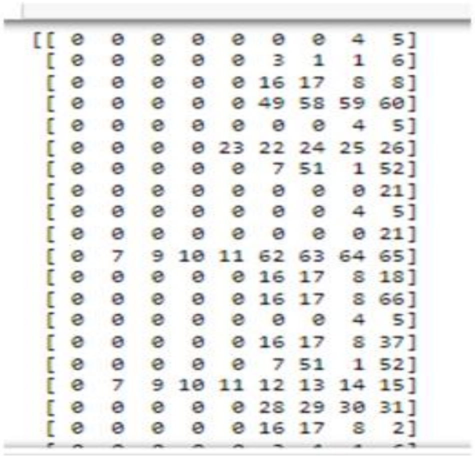

Extract Features and Labels. IP information are the features and Malicious column serve as the label
Split the Extracted X (Features) and Y(Labels) to Train(70%) and Test(30%) datasets
Tokenize the text data(IP Information) in both training and test datasets
Normalize the token to be of same list size for processing

Fig. 10. Output of data prepossessing

International Journal of Core Engineering & Management
Volume-8, Issue-01, 2025 ISSN No: 2348-9510

The output below shows the sample of final tokenized data.



```
[[ 0  0  0  0  0  0  0  4  5]
 [ 0  0  0  0  0  3  1  1  6]
 [ 0  0  0  0  0 16 17  8  8]
 [ 0  0  0  0  0 49 58 59 60]
 [ 0  0  0  0  0  0  0  4  5]
 [ 0  0  0  0 23 22 24 25 26]
 [ 0  0  0  0  0  7 51  1 52]
 [ 0  0  0  0  0  0  0  0 21]
 [ 0  0  0  0  0  0  0  4  5]
 [ 0  0  0  0  0  0  0  0 21]
 [ 0  7  9 10 11 62 63 64 65]
 [ 0  0  0  0  0 16 17  8 18]
 [ 0  0  0  0  0 16 17  8 66]
 [ 0  0  0  0  0  0  0  4  5]
 [ 0  0  0  0  0 16 17  8 37]
 [ 0  0  0  0  0  7 51  1 52]
 [ 0  7  9 10 11 12 13 14 15]
 [ 0  0  0  0  0 28 29 30 31]
 [ 0  0  0  0  0 16 17  8  2]
```

Fig. 11. Tokenized data

D. Section 4: Create Model Using Vector Machines

We used Support Vector Machine(SVM) in our project to perform the machine learning functions. The SVM Library we used is for sklearn which is widely used and proven to perform well. Support Vector Machine is a powerful supervised ma- chine learning algorithm that tries to find a hyperplane that best separates two classes. SVM can be used for both regression and classification tasks, but widely used in classification tasks. There are different types of SVM. We used Linear SVM as the data we are trying to learn are perfectly linearly separable. In other words, data can be perfectly classified into 2 by using a single linear straight line. Code for implementing SVM is given below.

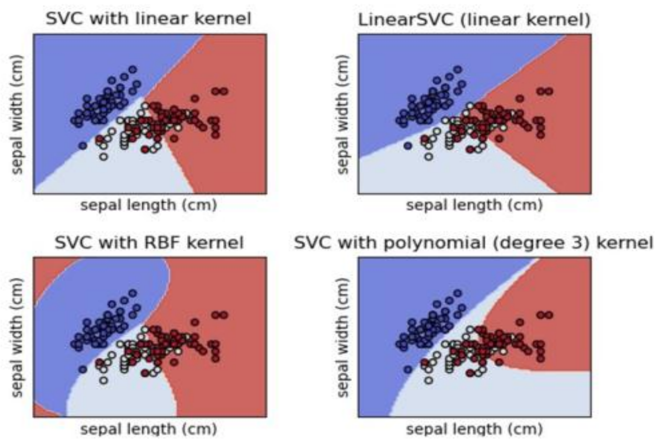


Fig. 12. From SKLEARN Documentation [6]

E. Section 5: Performance Analysis

The following matrices are calculated to validate the model. Accuracy: Percentage of total correct prediction Precision:

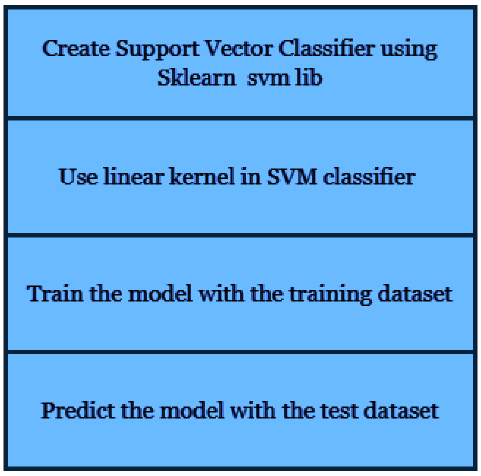
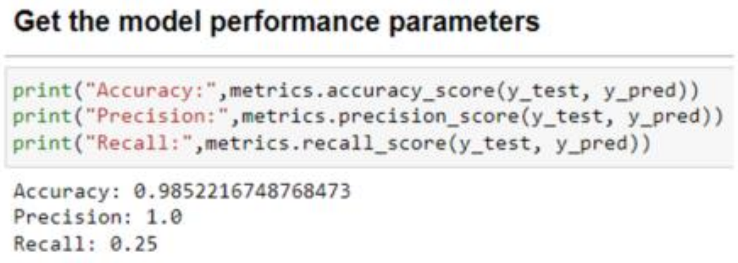

Create Support Vector Classifier using Sklearn svm lib
Use linear kernel in SVM classifier
Train the model with the training dataset
Predict the model with the test dataset

Fig. 13. SVM Code

Proportion of positive cases that are identified correctly. Re- call: Recall is the proportion of actual positive cases which are correctly identified.



Get the model performance parameters

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.9852216748768473
Precision: 1.0
Recall: 0.25
```

Fig. 14. SVM Code

VII. CONCLUSION

The primary objective of this endeavour was to showcase the feasibility of generating, logging, and subsequently analyzing sample malicious traffic through automated means. In achieving this aim, the project effectively demonstrated the programmatic capability to identify such malicious traffic in- stances. Nonetheless, it's worth noting that the demonstration was conducted under tightly controlled conditions, relying on historical data from logs and predefined IP addresses for recognition purposes. While this exercise underscored the potential for traffic recognition given appropriate data, it represents just a preliminary stage in the evolution of this capability.

Moving forward, the natural progression would entail re- fining the process to enable real-time analysis of live data streams. This advancement would empower the program to promptly detect malicious traffic instances as they occur and provide timely alerts regarding their presence and severity levels. However, it's important to acknowledge that implementing such real-time functionality extends beyond the current scope of the project. Nonetheless, laying the groundwork for this future development remains a crucial consideration for maximizing the efficacy of traffic monitoring and threat detection systems.

VIII. CURRENT STATE

The current state of AI in network security is quite advanced and actively evolving. AI and Machine Learning (ML) are increasingly being integrated into cybersecurity systems to enhance threat detection, intrusion response, and malware identification. According to a recent survey, 45 percent of organizations have already implemented AI and ML in their cybersecurity systems, and an additional 35 percent plan to do so. However, 20 percent of organizations are hesitant, believing it's not yet the right time to adopt these technologies. AI in cybersecurity represents a transformative era, with potential to significantly improve security measures. It's particularly promising for improving threat detection and response capabilities. Despite the optimism, the integration of AI into security workflows faces challenges such as dual-use concerns, bridg- ing skill gaps, and ensuring appropriate reliance on automated systems(2). For 2024, over half of the organizations surveyed are planning to implement generative AI solutions, indicating a strong trend towards more sophisticated AI applications in the security sector(2).

IX. FUTURE WORK

In the future with time and resources, we can envision this project to go under enhancement to evolve as a new product that serves as Smart Network Analysis and Management. This requires enhancements like replacing wireshark with libraries that are integrated with the python code that act as server which expect the input from these libraries and pass to machine learning code for prediction. A web application can be implemented to visualize the data, analysis and prediction results.

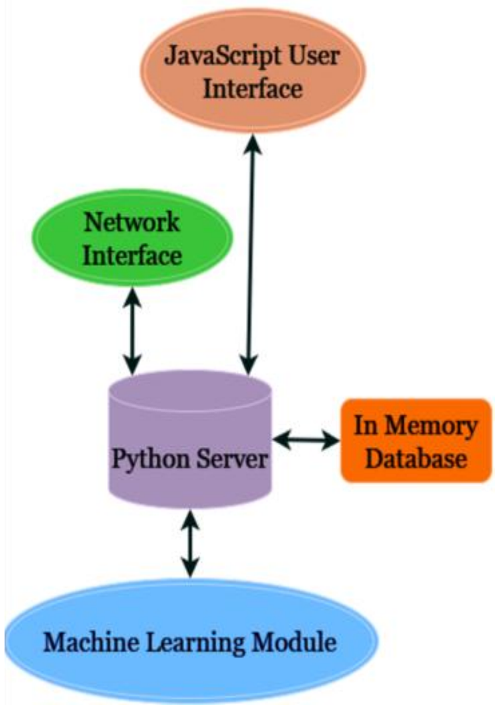


Fig. 15. Future Implementation Idea

REFERENCES

1. Wireshark Network Protocol Analyzer Available at: https://www.wireshark.org.
2. FlightSim Reference. Available at: https://github.com/alphasoc/flightsim.
3. FlightSim Blog. Available at: https://medium.com/alphasoc/uncover-your- detection-blindspots-with-network-flight-simulator-cda358b8749e.
4. Elisa Bertino, Murat Kantarcioglu, Cuneyt Gurcan Akcora, Sagar Sam- tani, Sudip Mittal, and Maanak Gupta. AI for Security and Security for AI. In Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy (CODASPY '21), pages 333–334, Asso- ciation for Computing Machinery, New York, NY, USA, 2021. Available at: https://doi.org/10.1145/3422337.3450357.
5. Yupeng Hu, Wenxin Kuang, Zheng Qin, Kenli Li, Jiliang Zhang, Yansong Gao, Wenjia Li, and Keqin Li. Artificial Intelligence Security: Threats and Countermeasures. ACM Comput. Surv., 55(1): Article 20, January 2023, 36 pages. Available at: https://doi.org/10.1145/3487890.
6. Sklearn Documentation Available at: https://scikit-learn.org/stable/autoexamples/svm/plotirissvc.html.