D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE

Unmasking VPN: A Machine Learning Approach to Tracing IP Addresses

Project report

submitted to D Y Patil International University, Akurdi, Pune in partial fulfilment of full-time degree.

BTech Computer Science and Engineering

(AI/ML and CYBER – Track)

Submitted By:

Deep V. Nandre - 20190802033

Vedant R. Landge - 20190802005

Krish J. Shetty - 20190802129

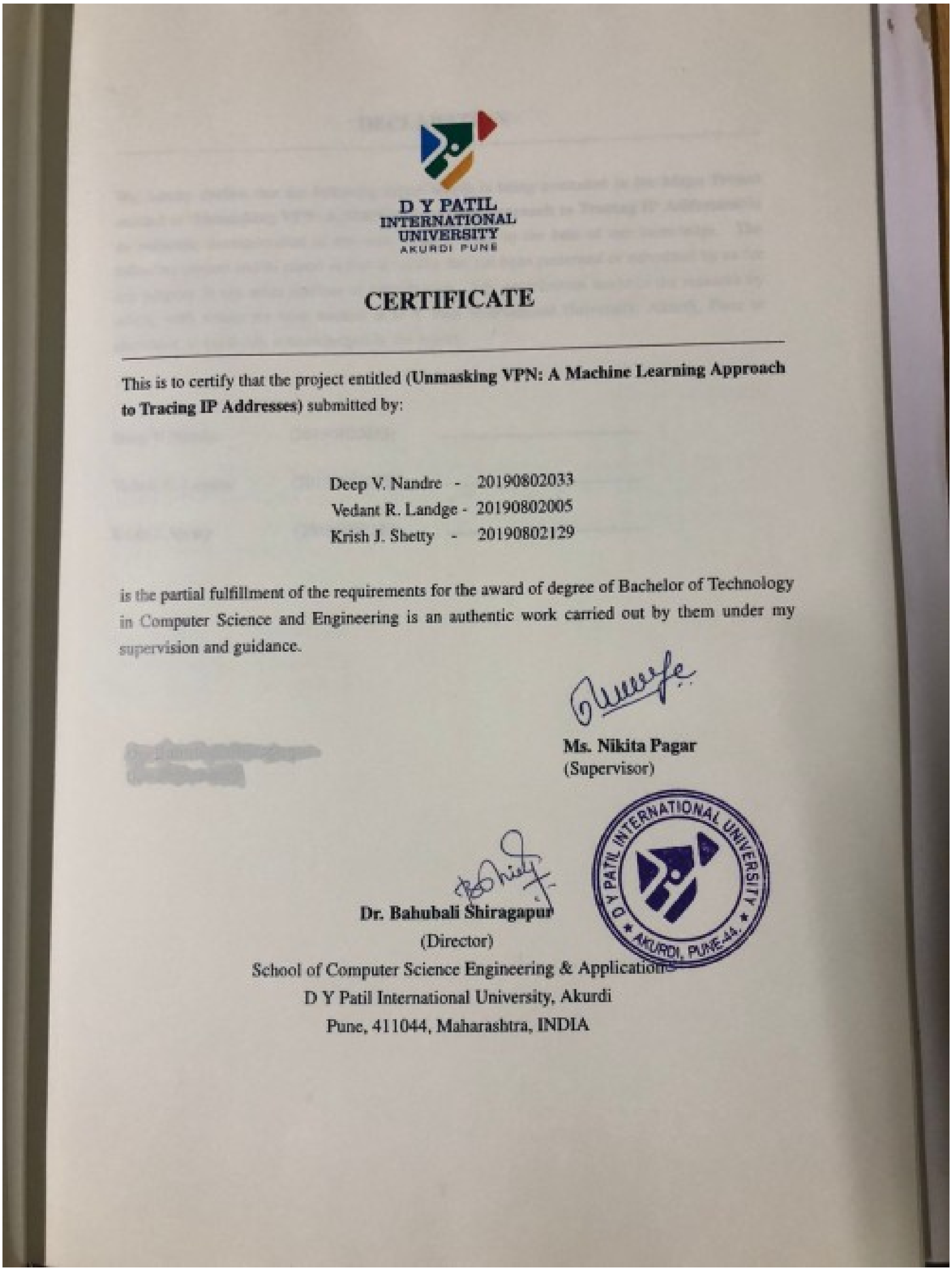
Under the Guidance of

(Ms. Nikita Pagar)

Department of Computer Science and Engineering

D Y Patil International University, Akurdi,Pune, INDIA, 411044

[Session 2019-23]



D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE

CERTIFICATE

This is to certify that the project entitled (**Unmasking VPN: A Machine Learning Approach to Tracing IP Addresses**) submitted by:

Deep V. Nandre	-	20190802033
Vedant R. Landge	-	20190802005
Krish J. Shetty	-	20190802129

is the partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering is an authentic work carried out by them under my supervision and guidance.

[illegible]

[redacted]

Ms. Nikita Pagar
(Supervisor)

D Y PATIL INTERNATIONAL UNIVERSITY * AKURDI, PUNE-44. *

[illegible]

Dr. Bahubali Shiragapur
(Director)
School of Computer Science Engineering & Applications
D Y Patil International University, Akurdi
Pune, 411044, Maharashtra, INDIA

DECLARATION

We, hereby declare that the following report which is being presented in the Major Project entitled as **(Unmasking VPN: A Machine Learning Approach to Tracing IP Addresses**) is an authentic documentation of our own original work to the best of our knowledge. The following project and its report in part or whole, has not been presented or submitted by us for any purpose in any other institute or organization. Any contribution made to the research by others, with whom we have worked at D Y Patil International University, Akurdi, Pune or elsewhere, is explicitly acknowledged in the report.

Deep V. Nandre	(20190802033)	______________________
Vedant R. Landge	(20190802005)	______________________
Krish J. Shetty	(20190802129)	______________________

ACKNOWLEDGEMENT

With due respect, we express our deep sense of gratitude to our respected guide and coordinator (Dr. Bahubali Shiragapur), for his valuable help and guidance. We are thankful for the encouragement that he has given us in completing this project successfully.

It is imperative for us to mention the fact that the report of major project could not have been accomplished without the periodic suggestions and advice of our project supervisor (Dr. Bahubali Shiragapur) and project mentor (Ms. Nikita Pagar).

We are also grateful to our respected Director, Dr. Bahubali Shiragapur and Hon'ble Vice Chancellor, DYPIU, Akurdi, Prof. Prabhat Ranjan for permitting us to utilize all the necessary facilities of the college.

We are also thankful to all the other faculty, staff members and laboratory attendants of our department for their kind cooperation and help. Last but certainly not the least; we would like to express our deep appreciation towards our family members and batch mates for providing support and encouragement.

Deep V. Nandre - 20190802033
Vedant R. Landge - 20190802005
Krish J. Shetty - 20190802129

ABSTRACT

With the exponentially increasing use of VPNs, it has become immensely difficult to trace the actual IP address of devices accessing the Internet. Therefore, it is of utmost importance to fully comprehend the techniques and tools available for uncovering the true IP address of a device. One of the most effective and constantly evolving ways to uncover a device's true IP address is by meticulously analyzing network traffic. Network traffic analysis, which requires specialized tools and a thorough understanding of network protocols, allows for the identification of packets that contain the actual IP address, even if the device is using an advanced VPN. Another continuously improving method for uncovering a device's true IP address is to look for leaked IP addresses, which can be caused by an array of complex configuration errors or bugs in a multitude of applications and services. These leaks can occur when using a cutting-edge VPN can provide valuable information about the actual IP address of the device. However, it is absolutely essential to consider the ever-changing privacy and legal implications of tracing IP addresses. Unlawful tracing of someone's IP address without their consent can be a severe violation of their privacy, and in some cases, it may even be a felony. Therefore, it is crucial to fully comply with the latest relevant laws and regulations and to exercise extreme caution and prudence when tracing IP addresses to avoid any and all potential legal repercussions.

Keywords: Tracing IP address, VPNs, masks, network traffic, searching leaked IP addresses, IP tracing services, privacy.

TABLE OF CONTENTS

List of Figures

List of Tables

1. INTRODUCTION

In today's digital landscape, the anonymity provided by Virtual Private Networks (VPNs) has become a double-edged sword. While these tools offer privacy and security benefits to legitimate users, they also create significant challenges for identifying and tracing malicious actors. The ability to uncover the true IP address behind a VPN has become a critical need in the field of cybersecurity. This project aims to delve into the intricacies of IP tracing, exploring various techniques and tools to overcome the hurdles presented by these anonymizing technologies.

Imagine a scenario where an organization falls victim to a devastating cyberattack, resulting in significant data breaches and financial losses. Despite their best efforts to detect and prevent such threats, the attackers successfully masked their identities using VPNs or proxies, making it incredibly challenging to trace their activities back to their original source. This is where our project comes into play.

Our project aims to delve deep into the complex world of IP tracing, focusing specifically on the challenges posed by VPNs. We will explore various cutting-edge techniques and tools used to uncover the true origin of network traffic, bypassing the obfuscation techniques employed by malicious actors. By understanding these techniques, we can equip organizations and security professionals with the necessary knowledge and tools to identify and thwart unauthorized access attempts effectively.

Through extensive research and practical experiments, we seek to shed light on the limitations of existing IP tracing methods and propose innovative solutions to enhance their effectiveness. By bridging the gap between theory and practice, our project strives to provide tangible insights and practical recommendations for the detection and prevention of cyber threats facilitated by VPNs and proxies.

Join us on this exciting journey as we unravel the complexities of IP tracing in the context of VPNs and proxies, exploring new avenues to strengthen cybersecurity measures and protect organizations from malicious actors seeking to exploit the cloak of anonymity provided by these technologies.

1.1. Background

In today's digital landscape, ensuring the security and integrity of online activities has become a paramount concern. With the increasing prevalence of cyber threats, individuals and organizations are faced with the challenge of identifying and mitigating potential risks. Among these challenges, the use of anonymizing tools such as Virtual Private Networks (VPNs) has emerged as a significant hurdle. While VPNs and proxies provide enhanced privacy and freedom, they can also be exploited by malicious actors to conceal their identities and carry out illicit activities, making it difficult to trace their true origin.

Conventional methods of IP tracing, which rely on geolocation databases and traditional network monitoring techniques, are often inadequate in dealing with the sophisticated tactics employed by cybercriminals. As a result, organizations are left vulnerable to security breaches and data theft. To address this pressing issue, innovative approaches and advanced technologies are needed to effectively trace IP addresses behind VPNs and proxies, enabling the identification and prevention of unauthorized access.

This project aims to bridge the gap between existing tracing methods and the evolving techniques employed by malicious actors in anonymized network environments. By conducting extensive research, experimentation, and analysis, we seek to uncover novel insights and propose effective solutions for the detection and mitigation of unauthorized access. The outcomes of this project will empower organizations, security professionals, and law enforcement agencies with the knowledge and tools necessary to safeguard their digital assets and maintain a secure online ecosystem.

1.2. Objectives

The main objective of this project is:

1. To understand the challenges of tracing an IP address behind a VPN.
2. To explore the different techniques and tools available for tracing IP addresses.
3. To evaluate the feasibility and limitations of these techniques and tools.
4. To identify potential research gaps and problem statements in the field of IP tracing.

1.3. Problem statement

Identifying unauthorized users in the digital realm presents a formidable challenge, especially in the face of the rising use of identity concealment tools such as anonymizing proxies and Virtual Private Networks (VPNs) by malicious actors. The existing techniques used to detect such actors have proven to be insufficient in many cases, leaving organizations vulnerable to security breaches and data theft. The need for a reliable and effective method to identify and prevent such unauthorized access has become increasingly urgent, given the significant risks and potential damage that can result from a successful breach.

1.4. Feasibility

Tracing IP addresses behind VPNs (Virtual Private Networks) is indeed a complex task due to the way these technologies work. When a user connects to a VPN, their device appears to have the IP address of that server instead of its actual IP address. This process is intended to provide anonymity and privacy to the user.

However, it is still possible to uncover the true IP address in certain circumstances. Advanced techniques and tools can be employed to gather information and analyze network traffic to identify the original IP address behind the VPN.

One approach to tracing IP addresses is by examining the network packets that travel between the user's device and the VPN. These packets contain various headers and information that can be analyzed to uncover the original IP address. By carefully examining the packet headers and using techniques like deep packet inspection, it is possible to identify patterns and clues that can lead to the discovery of the real IP address.

Another method involves leveraging vulnerabilities or weaknesses in the VPN itself. While VPNs are designed to provide security and anonymity, they are not infallible. If a VPN has any vulnerabilities or misconfigurations, it might be possible to exploit them to reveal the original IP address.

However, the feasibility of these techniques and tools depends on several factors. One crucial factor is the type of VPN being used. Some VPNs are more secure and better designed to protect user privacy, making it more challenging to trace the IP address. On the other hand, poorly implemented or less secure VPNs might have vulnerabilities that can be exploited more easily.

Additionally, the level of security measures implemented by the VPN provider also plays a role. Reputable VPN providers often have stronger security measures in place to protect user privacy, making it more difficult to trace IP addresses. They may employ advanced encryption protocols, network obfuscation techniques, and regularly update their infrastructure to mitigate potential vulnerabilities.

Furthermore, the skills and resources of the individual or organization attempting to trace the IP address also affect feasibility. Sophisticated tools and techniques require expertise and knowledge in network analysis, packet inspection, and cybersecurity.

It's important to note that tracing IP addresses behind VPNs should be done within legal and ethical boundaries. There may be legitimate reasons to trace IP addresses, such as combating cybercrime, unauthorized tracing can invade privacy and violate legal guidelines.

2. LITERATURE REVIEW

- Shane Miller, Kevin Curran and Tom Lunney: The paper explores methods for detecting VPN use in cyberattacks. Criminals often use VPNs to remain anonymous while committing crimes, making it challenging to identify them. Deep Packet Inspection (DPI) is one method used for identifying VPN traffic, but end-to-end encryption can make this difficult. The paper proposes the use of machine learning techniques to classify OpenVPN and Stunnel OpenVPN traffic, and the experiments found the neural networks were successful in identifying VPN traffic. The research highlights the importance of preventing network breaches and the effectiveness of machine learning in identifying potential attacks.

- Miller, S., Curran, K., Lunney, T: The paper focuses on detecting anonymizing web proxies used by criminals to remain anonymous. It proposes a neural network model capable of classifying anonymizing proxy traffic versus normal traffic not being routed through a proxy. The validation test showed the model was successful in identifying anonymizing proxy traffic. The study highlights the need for larger datasets to further test the strength of the models and the exploration of other machine learning algorithms to compare their training time and accuracy.

- Gayathri RajaKumaran, NeelaNarayanan Venkataraman: The paper presents an overview of security attacks in wired environments and possible control measures for mitigation. It is a preliminary assessment to understand the various attack categories and controls. The focus is on selecting a tool or platform to simulate a DDoS attack in a cloud computing environment, with effectiveness verified by probing the server. Future research can be done by creating a testbed with multiple systems and analyzing the flooding rate of other DDoS tools.

- Ms. Shipra Srivastava, Harigovind H, Devvrat Modi, Bassam Salim, Yashraj Mathur: The paper explores the use of VPNs by attackers to hide their identity, which can make it difficult for organizations to detect and prevent network breaches. The authors discuss techniques such as Deep Packet Inspection (DPI) and end-to-end encryption, which can be used to evaluate network traffic and detect VPN usage. However, attackers can also use encryption technologies such as IPSec and SSL/TLS to further conceal their activities. The paper proposes the use of artificial intelligence (AI) techniques, such as neural networks, to identify VPN traffic accurately. The authors conducted experiments using OpenVPN and Stunnel OpenVPN. They found that the neural network could accurately classify network traffic as either VPN or non-VPN traffic with a high degree of accuracy. The paper suggests that the use of AI can be an effective way to detect and prevent malicious activity on networks.

- Ben Feher, Lior Sidi, Asaf Shabtai, Rami Puzis, Leonardas Marozas: The paper analyzed current research related to threats and attacks against WebRTC, a communication technology that offers strong security features but is not immune to direct attacks against its components and underlying host applications. The study identified appropriate mitigation techniques for most of the threats but highlighted the need for further research to address issues related to Lawful Interception, Firewall Traversal and application of security policies to WebRTC communications. Despite these vulnerabilities, WebRTC is considered to be one of the most secure solutions in the market, and the article offers a roadmap for implementing a secure WebRTC application.

2.1. Drawbacks of existing system

- Inadequate Geolocation Accuracy: Existing IP tracing techniques heavily rely on geolocation databases to determine the approximate physical location of an IP address. However, these databases often lack accuracy and precision, leading to incorrect or misleading results. Inaccurate geolocation information hampers the ability to identify the actual origin of a request, making it challenging to attribute actions to specific individuals or entities accurately.
- Encryption and Anonymization Challenges: VPNs and proxies are designed to encrypt and anonymize network traffic, making it difficult to trace the true IP address behind these services. The existing system struggles to overcome the encryption and anonymization layers, resulting in limited visibility into the actual source of the network traffic. This limitation undermines the efforts to identify unauthorized users and potential threats.
- Evolving Tactics of Malicious Actors: Cybercriminals are constantly evolving their techniques to bypass detection and remain undetected. They employ sophisticated methods to obfuscate their identities and conceal their true IP addresses. The existing system often fails to keep pace with these evolving tactics, leading to an incomplete and outdated understanding of the methods used by malicious actors. This gap hinders effective identification and prevention of unauthorized access.
- Limited Scalability and Performance: As network traffic continues to increase in volume and complexity, the existing system faces scalability and performance challenges. Processing and analyzing large amounts of data in real-time become arduous tasks, leading to delays in IP tracing and potentially missing critical information. This limitation poses a significant obstacle in timely response and mitigation of security incidents.

2.2. Gaps Identified

- Limited Scope of Machine Learning Techniques: Some studies suggest the use of machine learning, particularly neural networks, for identifying VPN traffic, there is a need for further exploration and comparison of different machine learning algorithms. The existing literature emphasizes the importance of utilizing larger datasets to thoroughly evaluate the strength and accuracy of these models. By expanding the scope of machine learning techniques and conducting experiments on comprehensive datasets, researchers can gain more insights into the effectiveness and limitations of various algorithms in accurately detecting VPN traffic.

- Lack of Diversity in Data Set Representation: While the methodology mentions the importance of selecting a diverse data set, there is a gap in specifying the criteria for diversity. It is crucial to consider various VPN protocols, network configurations, and traffic patterns to ensure the model's effectiveness in real-world scenarios. Further research is needed to identify and incorporate diverse data sets that encompass a wide range of VPN usage scenarios.

- Insufficient Evaluation of Detection Techniques: While some studies propose the use of Deep Packet Inspection (DPI) and end-to-end encryption evaluation for detecting VPN usage, the literature indicates a need for comprehensive evaluation and validation of these techniques. Further research should focus on assessing the effectiveness and limitations of DPI and encryption analysis in real-world scenarios.

- Limited Consideration of Model Explainability: The methodology also focuses on the accuracy and performance metrics of the machine learning model, but it overlooks the aspect of model explainability. The transparency and interpretability of the model's decision-making process are important for understanding the factors contributing to the classification of VPN traffic. Further investigation is needed to explore techniques that provide insights into the model's decision process, enabling security analysts to trust and interpret the results effectively.

- One of the gaps identified in the methodology is the limited exploration of feature selection techniques. Feature selection plays a crucial role in accurately classifying VPN and non-VPN traffic. However, the methodology does not provide explicit details on the process of selecting the most relevant features. This gap hinders the potential enhancement of the model's performance and the identification of the most informative features for effective detection. To address this, further research is needed to thoroughly explore and evaluate various feature selection techniques, enabling the identification of key features that contribute significantly to the accurate classification of VPN traffic.

2.3. Project Plan

Month	**Date**	**Project Activity**
February	01/02/2023	Project Topic Selection
February	06/02/2023	Synopsis Submission
February	13/02/2023	Presentation on project ideas
February	20/02/2023	Submission of Literature Survey & Feasibility
March	01/03/2023	Preparation of Presentation and Report for Stage-1 evaluation
March	06/03/2023	Stage-1 Evaluation complete
March	13/03/2023	Started implementing ML
March	20/03/2023	Worked on a dump IP database & created a VPN tunnel
March	27/03/2023	Created a Website successfully
April	03/04/2023	Testing out different IPs with & without VPN/TOR
April	10/04/2023	Preparation of Presentation and Report for Stage-2 evaluation
April	17/04/2023	Stage-2 Evaluation complete
April	24/04/2023	Study of different algorithms
May	01/05/2023	Discussion about modifications
May	08/05/2023	Discussion on flow of project
May	15/05/2023	Started working on research paper
May	22/05/2023	Designed test cases for our flask application
May	29/05/2023	Integrating and Deployment of Devops methodology
June	05/06/2023	Final Report
June	16/06/2023	Final Presentation Stage 3

Table 2.1: Project Plan

3. PROPOSED METHODOLOGY

- Problem Definition: The research problem addressed in this study is the detection of VPN advances used by attackers to conceal their identity. The specific objective is to develop a robust method for accurately identifying VPN traffic and classifying it as either VPN or non-VPN traffic.

- Literature Review: A comprehensive literature review was conducted to investigate existing techniques and tools employed for the identification and classification of VPN traffic. This review helped establish the current state of the art in the field and identify research gaps. Multiple high-quality research papers were meticulously studied, and methodologies were critically reviewed to inform the approach adopted in this study.

- Data Set Selection: A carefully chosen, diverse data set encompassing both VPN and non-VPN traffic was selected. The data set's size was carefully considered to ensure it provided a representative sample of various traffic scenarios. Extensive research efforts were dedicated to identifying and acquiring substantial datasets specifically focused on VPN traffic, non-VPN traffic, Tor traffic, and non-Tor traffic.

- Data Preprocessing: The collected data underwent a meticulous preprocessing phase to eliminate any noise, outliers, or irrelevant information. This involved employing robust filtering techniques, conducting thorough data cleaning procedures, and transforming the data into a suitable format for subsequent analysis.

- Machine Learning Algorithm Selection: An appropriate machine learning algorithm was carefully selected based on its suitability for addressing the research problem. The chosen algorithm needed to possess the capability to accurately classify traffic as either VPN or non-VPN. Several state-of-the-art supervised and unsupervised machine learning algorithms, including classification, clustering, and anomaly detection, were considered. The implementation of the selected algorithm was accomplished using widely adopted Python libraries such as scikit-learn, TensorFlow, and PyTorch.

- Model Training: The selected machine learning model was extensively trained using the preprocessed data set. The training process involved meticulous adjustments of hyperparameters to optimize the model's accuracy and performance.

- Custom VPN Generation for Testing: A custom VPN infrastructure was created using a Wireguard VPN tunnel, with multiple tunnels configured for different countries, such as the UK, USA, Japan, and others. Additionally, custom proxies were generated using industry-standard tools like Private Internet Access (PIA) and shadowsocks. These custom VPNs and proxies played a critical role in thoroughly testing the accuracy and effectiveness of the developed model.

- Model Validation: To ensure the reliability and generalization capability of the model, an independent data set that was not used during the training phase was employed for model validation. This validation step aimed to verify that the model avoided overfitting and exhibited accurate classification performance on unseen data.
- Website Deployment using DevOps Tools (CI/CD): Our architecture utilizes a combination of AWS services, including AWS Elastic Kubernetes Service (EKS) and Elastic Container Registry (ECR), alongside tools like Python Flask, Docker, CI/CD, Kubernetes, Git, and Infrastructure as Code (IaC), to simplify deployment, achieve scalability, ensure reliability, and enable efficient software delivery. With a focus on DevOps methodologies, our approach streamlines the building, testing, and deployment processes, resulting in faster and more reliable releases for our Python Flask application.
- Performance Evaluation: The performance of the developed model was evaluated using rigorous evaluation metrics, including accuracy, precision, recall, and specifically the anomaly score. These metrics provided comprehensive insights into the model's effectiveness in detecting and classifying VPN traffic.
- Results Analysis: The obtained results were meticulously analyzed to determine the overall effectiveness of the proposed method for detecting VPN traffic. This analysis involved thorough comparisons with existing techniques, identification of limitations or areas for improvement, and the provision of valuable recommendations for future research directions.
- Conclusion Drawing: Based on the comprehensive findings and analysis, clear and concise conclusions were drawn to summarize the research outcomes. The implications of the findings, particularly the practical implications for network security, were extensively discussed. Furthermore, potential future applications and research directions in the field were identified and highlighted.
- Methodology Documentation: The methodology employed in this study was meticulously documented, encompassing detailed descriptions of the data set used, the feature selection process, the specific machine learning algorithms utilized, the validation techniques employed, and the performance metrics employed. This comprehensive documentation aims to provide a transparent and replicable process for future researchers in the field, enabling further advancements in the area 5.6

3.1. K-MEANS ALGORITHM AND ARCHITECTURE

In this project, we have tried to explore the application of the K-Means clustering algorithm as a means to detect the presence of VPNs in web applications. K-Means clustering, a widely used unsupervised machine learning algorithm, effectively groups data points based on their similarities. By employing this algorithm on IP addresses, we can identify patterns and anomalies associated with VPN usage.

K-Means clustering endeavors to partition a dataset into distinct clusters, with each cluster represented by a centroid. The algorithm iteratively adjusts the positions of these centroids to minimize the within-the-cluster variance. By assigning IP addresses to the nearest centroid, K-Means clustering unveils similarities and dissimilarities between IP addresses, facilitating the identification of potential VPN usage.

The fundamental aspect of using K-Means in our project is to calculate the anomaly scores. These values evaluate how far an IP address deviates from expected normal behavior, with higher scores indicating a greater possibility of the presence of VPNs. These anomaly scores can be calculated using a wide range of metrics, such as the Euclidean distance between an IP address and the assigned centroid. By comparing the scores to a predefined threshold, IP addresses can be classified as either normal or anomalous.

The integration of the K-Means clustering algorithm into a web application involves several key steps. Initially, the user provides an IP address through a text field in the application interface. The application then preprocesses the IP address, validating its format and applying necessary transformations. Subsequently, relevant features such as geographical information, network attributes, or historical data are extracted from the IP address. These features serve as inputs to a pretrained K-Means clustering model, which computes the anomaly score for the IP address. By comparing this score to a predefined threshold, the application can determine the presence of a VPN and display the result accordingly.

By leveraging the algorithm's ability to identify patterns and anomalies, it is feasible for us to detect potential VPN usage and mitigate associated risks.

Before we proceed, let us define some key terminology used in K-Means:

– Data points: These are the individual observations or instances in the dataset that we want to cluster. Each data point is typically represented as a feature vector in a multi-dimensional space.

– Centroids: These are the representatives of the clusters and are often referred to as cluster centers. Each centroid is a point in the feature space and is characterized by its coordinates.

- Clusters: These are groups of data points that share similar characteristics or are close to each other based on a similarity metric. Each cluster is associated with a centroid.

3.1.1. Input

- X: Dataset with n data points
- K: Number of clusters

3.1.2. Output

- centroids: Final centroids of the clusters
- cluster_assignments: Assignments of data points to clusters

3.1.3. Algorithm

I. Initialize the centroids:

- Randomly select K data points from X as the initial centroids.

II. Repeat until convergence:

1. Assignment Step:
   - Assign each data point to the nearest centroid by calculating the Euclidean distance between the data point and all centroids.
   - Compute the distance between each data point x and each centroid c as: dist(x, c) = sqrt(sum((x - c)2)).
   - Assign each data point x to the cluster with the nearest centroid: argmin(dist(x, c)).
2. Update Step:
   - Update the centroids by calculating the mean of the data points assigned to each cluster.
   - For each cluster, compute the new centroid c as the mean of all data points x in the cluster: c = mean(x).
   - The mean of a set of d-dimensional data points is calculated as: mean(x) = (1/n) * sum(x)
3. Check for convergence:
   - If the new centroids are close to the previous centroids (i.e., the change is within a defined threshold), convergence is achieved.

    - Repeat the assignment and update steps until convergence is achieved or a maximum number of iterations is reached.

III. Final Clustering Step:

- Assign each data point to its corresponding cluster based on the final centroids obtained.
- Calculate the distance between each data point and each final centroid.
- Assign each data point to the cluster with the nearest centroid.

IV. Calculate Anomaly Scores:

- For each data point, calculate the anomaly score as the distance between the data point and its assigned centroid.
- The anomaly score is computed using the Euclidean distance: anomalyscore(x) = dist(x, c)
- The lower the anomaly score, the closer the data point is to its assigned cluster centroid.

V. Optional:

- Save the trained K Means model for future use.

3.1.4. Working

1. Initialization:
   In the initialization step, K centroids are selected to serve as the starting points for the clustering process. The most common approach is to randomly choose K data points from the dataset as the initial centroids. Let us assume we have a dataset with IP addresses and their corresponding locations (region, city, country). For simplicity, let us consider the following example dataset:

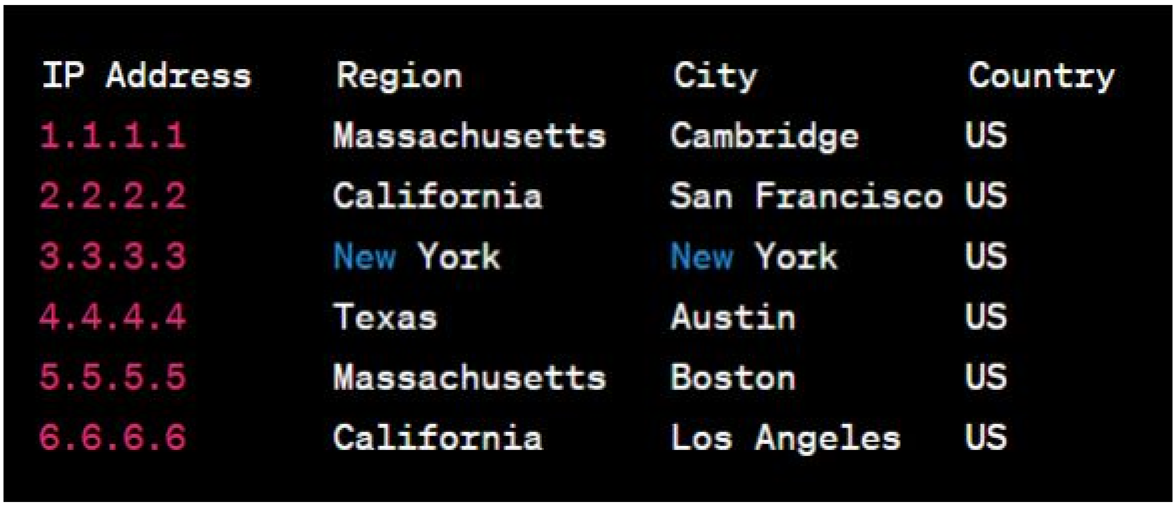
   In this example, let us assume we want to cluster the IP addresses based on their locations. We will set K=2, meaning we want to form two clusters. In order to initialize the centroids, we randomly select two data points from the dataset. Let us say we randomly choose the IP addresses 1.1.1.1 and 4.4.4.4 as the initial centroids. These selected centroids will serve as the starting points for the clustering process.

   Now, let us represent the selected centroids as feature vectors. We can use a one-hot encoding approach to represent the categorical variables (region, city, country). The feature vector for each centroid will be a binary representation of its corresponding region, city, and country. For example:

   (a) Centroid 1: ‘1.1.1.1’:

  - Region: [1, 0, 0] (Massachusetts)
  - City: [1, 0, 0] (Cambridge)
  - Country: [1, 0] (US)

(b) Centroid 2: ‘4.4.4.4’:
  - Region: [0, 0, 1] (Texas)
  - City: [0, 1, 0] (Austin)
  - Country: [1, 0] (US)



```
IP Address    Region          City           Country
1.1.1.1       Massachusetts   Cambridge      US
2.2.2.2       California      San Francisco  US
3.3.3.3       New York        New York       US
4.4.4.4       Texas           Austin         US
5.5.5.5       Massachusetts   Boston         US
6.6.6.6       California      Los Angeles    US
```

Figure 3.1: Working

These feature vectors will be used to calculate distances and assign data points to their nearest centroids during the subsequent steps of the algorithm.

2. Assignment Steps:
   In the assignment step, each data point is assigned to the cluster with the nearest centroid. The distance between a data point and a centroid is typically calculated using the Euclidean distance measure. Let us continue with the example dataset we used earlier.

   Assuming we have already initialized the centroids, let us represent them as feature vectors using one-hot encoding, as explained in the initialization step. For example, let us consider the following ccentroids:

   (a) Centroid 1: ‘1.1.1.1’:
     - Region: [1, 0, 0] (Massachusetts)
     - City: [1, 0, 0] (Cambridge)
     - Country: [1, 0] (US)

   (b) Centroid 2: ‘4.4.4.4’:
     - Region: [0, 0, 1] (Texas)

- City: [0, 1, 0] (Austin)
- Country: [1, 0] (US)

Now, we need to assign each data point to the cluster associated with the nearest centroid. We calculate the distance between each data point and each centroid using the Euclidean distance formula: amsmath

$$\text{distance}(x_i, c_j) = \sqrt{\sum_k (x_i^k - c_j^k)^2}$$

where x represents a data point, c represents a centroid, and k iterates over the dimensions of the feature space. In our case, the dimensions are the region, city, and country.

Let us consider the IP address '2.2.2.2' and calculate its distance from both centroids:

(a) Distance to Centroid 1:
distance(2.2.2.2,1.1.1.1)=
$\sqrt{((0-1)^2+(1-0)^2+(1-0)^2+(0-1)^2+(1-0)^2+(0-1)^2)} = \sqrt{6}$

(b) Distance to Centroid 2:
distance(2.2.2.2,4.4.4.4)=
$\sqrt{((1-0)^2+(0-0)^2+(0-1)^2+(0-0)^2+(0-1)^2+(1-0)^2)} = \sqrt{3}$
Since the distance to Centroid 2 is smaller, the IP address 2.2.2.2 will be assigned to Cluster 2.

We repeat this process for all data points, calculating their distances to each centroid and assigning them to the cluster with the nearest centroid.

3. Update Step:
In the update step, the positions of the centroids are adjusted based on the current assignments of data points. The new positions of the centroids are calculated as the mean of all the data points assigned to each cluster. This step aims to find the center of each cluster by minimizing the within-cluster variance.
Let us continue with the example dataset we used earlier.
Assuming we have already assigned each data point to a cluster based on the distances to the centroids, let us say the following assignments were made:

(a) Cluster 1:
    - IP Address: '1.1.1.1'
    - IP Address: '5.5.5.5'
(b) Cluster 2:

  - IP Address: ‘2.2.2.2’
  - IP Address: ‘3.3.3.3’
  - IP Address: ‘4.4.4.4’
  - IP Address: ‘6.6.6.6’

To update the positions of the centroids, we calculate the mean of the feature vectors of the data points assigned to each cluster. This will give us the new centroid positions.

i. For Cluster 1:
   - Calculate the mean of the feature vectors:
     Mean Feature Vector = (1/2) * ([1, 0, 0] + [1, 0, 0]) = [1, 0, 0]

ii. For Cluster 2:
   - Calculate the mean of the feature vectors:
     Mean Feature Vector = (1/4) * ([0, 0, 1] + [0, 1, 0] + [0, 0, 1] + [1, 0, 0]) = [0.25, 0.25, 0.5]

These mean feature vectors represent the updated positions of the centroids. We can use these updated centroids in the next iteration of the algorithm.

4. Iteration Steps:
   The iteration step is where the K Means algorithm iteratively performs the assignment and update steps until convergence is achieved. Convergence is reached when the centroids no longer move significantly between iterations or when a maximum number of iterations is reached.
   Let us continue with the example dataset we used earlier.
   Assuming we have already initialized the centroids and performed the assignment and update steps once, we can proceed with the iteration step. The process involves repeatedly assigning data points to the closest centroids and updating the centroids' positions.
   Here's an example of the iteration step:
   Assignment Step: Based on the current positions of the centroids, assign each data point to the closest centroid. For example, using the Euclidean distance as the distance metric, the assignments may look like this:

   (a) Cluster 1:
     - IP Address: ‘1.1.1.1’
     - IP Address: ‘5.5.5.5’

   (b) Cluster 2:
     - IP Address: ‘2.2.2.2’

- IP Address: ‘3.3.3.3’
- IP Address: ‘4.4.4.4’
- IP Address: ‘6.6.6.6’

Update Step: Calculate the mean feature vectors of the data points assigned to each cluster. These mean feature vectors represent the updated positions of the centroids.

i. For Cluster 1:
   - Calculate the mean of the feature vectors:
     Mean Feature Vector = (1/4) * ([0, 0, 1] + [0, 1, 0] + [0, 0, 1] + [1, 0, 0]) = [0.25, 0.25, 0.5]

ii. For Cluster 2:
   - Calculate the mean of the feature vectors:
     Mean Feature Vector = (1/4) * ([0, 0, 1] + [0, 1, 0] + [0, 0, 1] + [1, 0, 0]) = [0.25, 0.25, 0.5]

Convergence Check: Check if the updated centroids are significantly different from the previous centroids. If the centroids have not moved significantly or the maximum number of iterations has been reached, the algorithm converges, and the process is stopped. Otherwise, go back to step 1 and repeat the assignment and update steps.
The iteration step continues until convergence is achieved. At convergence, the algorithm has found stable cluster assignments and centroid positions.

5. Final Clustering:
   After the iterations are complete and convergence is achieved, the final clustering step assigns each data point to its corresponding cluster based on the converged centroids.
   Let us continue with the example dataset we used earlier.
   Assuming we have completed the iterations and have the converged centroids, we can proceed with the final clustering step.
   Here is an example of the final clustering step:

   (a) Assignment Step:
   Based on the converged centroids, assign each data point to the closest centroid. For example, using the Euclidean distance as the distance metric, the final cluster assignments may look like this:

   i. Cluster 1:
      - IP Address: ‘1.1.1.1’

    - IP Address: ‘5.5.5.5’

   ii. Cluster 2:
    - IP Address: ‘2.2.2.2’
    - IP Address: ‘3.3.3.3’
    - IP Address: ‘4.4.4.4’
    - IP Address: ‘6.6.6.6’

(b) Cluster Labeling:
You can assign labels to the clusters based on the nature of the data points in each cluster. For example, you can label Cluster 1 as ”Massachusetts Cluster” and Cluster 2 as ”Diverse Locations Cluster” based on the regions and cities of the data points.

   i. Cluster 1 (Massachusetts Cluster):
    - IP Address: ‘1.1.1.1’ (Massachusetts, Cambridge, US)
    - IP Address: ‘5.5.5.5’ (Massachusetts, Boston, US)

   ii. Cluster 2 (Diverse Locations Cluster):
    - IP Address: ‘2.2.2.2’ (California, San Francisco, US)
    - IP Address: ‘3.3.3.3’ (New York, New York, US)
    - IP Address: ‘4.4.4.4’ (Texas, Austin, US)
    - IP Address: ‘6.6.6.6’ (California, Los Angeles, US)

(c) Visualization:
You can visualize the final clusters using plots or other visualization techniques. This helps in understanding the distribution and separation of the data points in the clusters.
For example, you can create a scatter plot where the x-axis represents the regions, the y-axis represents the cities, and the data points are colored based on their cluster assignments. This visualization provides insights into the clustering results.

3.1.5. Advantages of K-Means Algorithm

1. Simplicity and Efficiency: K-Means clustering offers a straightforward and computationally efficient solution. Its ease of implementation and scalability with large datasets make it ideal for real-time or near real-time VPN detection in web applications, where swift responses are crucial.
2. Unsupervised Learning: K-Means clustering is an unsupervised learning algorithm, obviating the need for labeled data during training. In the context of

VPN detection, this eliminates the requirement for a labeled dataset containing known VPN IP addresses. The algorithm autonomously learns patterns and clusters IP addresses based on inherent similarities, without any prior knowledge.

3. Anomaly Detection: K-Means clustering effectively identifies anomalies within datasets. By computing anomaly scores derived from the distances between IP addresses and their respective centroids, it becomes possible to detect IP addresses that significantly deviate from the norm. This capability is critical for identifying potential VPN usage and discerning it from regular network traffic.
4. Flexibility and Adaptability: K-Means clustering readily adapts to diverse datasets and applications. It can be applied to a wide range of features extracted from IP addresses, such as geographical information, network behavior, or historical data. This flexibility facilitates customization and the integration of domain-specific knowledge, enhancing the accuracy and effectiveness of VPN detection.
5. Scalability: K-Means clustering efficiently handles large volumes of IP addresses. As the number of IP addresses increases, the algorithm's scalability ensures the clustering process remains feasible without compromising the web application's performance.

3.1.6. Limitation of K-Means Algorithm

1. Determining the Number of Clusters: An inherent limitation of K-Means clustering lies in the need to predefine the number of clusters (K). Determining the optimal value for K can be challenging in VPN detection, as it depends on dataset characteristics and desired analysis granularity. Selecting an inappropriate K value may yield suboptimal clustering outcomes.
2. Sensitivity to Initial Centroid Selection: K-Means clustering is sensitive to the initial centroid selection. Different initializations can lead to diverse clustering outcomes, potentially resulting in suboptimal solutions. To address this, multiple runs with varied initializations or advanced techniques like K-Means++ can be employed.
3. Assumption of Spherical Clusters: K-Means clustering assumes clusters to be spherical and of similar sizes. However, in VPN detection, IP address distributions may not adhere to these assumptions. Clusters with irregular shapes or varying sizes can impact clustering results, potentially affecting VPN detection accuracy.
4. Lack of Outlier Handling: K-Means clustering lacks explicit outlier handling capabilities. Outliers, representing unusual or rare network behavior, may influence clustering and lead to inaccurate VPN identification. Additional outlier detection techniques or preprocessing steps may be necessary to mitigate this limitation.

5. Difficulty in Interpreting Results: Interpreting K-Means clustering results, particularly in highdimensional data, can be challenging. While the algorithm provides clustering sssignments and anomaly scores, understanding the underlying reasons for clustering patterns or identifying specific VPN characteristics may require further analysis and domain expertise.

6. Despite these limitations, K-Means clustering remains a valuable tool for VPN detection in web applications. Awareness of these advantages and limitations empowers developers to make informed decisions during implementation, further enhancing the accuracy and efficacy of VPN detection systems.

3.1.7. Implementation

The implementation of the K-Means clustering algorithm involves the following steps:

1. Import the required libraries: Start by importing the necessary libraries such as NumPy and scikit-learn.

2. Load and preprocess the data: Load the dataset that contains the IP addresses and preprocess them by applying any necessary transformations or feature extraction techniques.

3. Choose the number of clusters (K): Decide on the appropriate number of clusters based on the dataset and problem domain.

4. Initialize the centroids: Randomly select K data points from the dataset as the initial centroids.

5. Assign data points to clusters: Calculate the distance between each data point and the centroids and assign each data point to the nearest centroid.

6. Update the centroids: Recalculate the centroids by computing the mean of all data points assigned to each centroid.

7. Repeat steps 5 and 6 until convergence: Iterate the assignment and centroid update steps until the centroids no longer change significantly or a maximum number of iterations is reached.

8. Evaluate the results: Assess the quality of the clustering results using appropriate metrics such as the within-cluster sum of squares or silhouette score.

9. Use the trained model for predictions: Apply the trained K-Means clustering model to new data points to classify them into the identified clusters.

3.1.8. Analysis

The K-Means clustering algorithm is a valuable tool for detecting VPNs in web applications due to its ability to identify patterns and anomalies in data. By leveraging this algorithm, we can effectively categorize IP addresses as either normal or anomalous based on their deviation from expected behavior. This capability has significant implications for enhancing web application security and mitigating potential risks associated with VPN usage.

The implementation of the K-Means clustering algorithm involves several important steps that contribute to its effectiveness. Firstly, data preprocessing is necessary to prepare the IP address data for clustering. This may involve removing outliers, normalizing the data, or handling missing values. By ensuring the data is in the appropriate format, we can maximize the algorithm's performance.

Next, the algorithm requires the initialization of centroids, which serve as representative points for each cluster. The selection of initial centroids can impact the final clustering results, as the algorithm tends to converge to a local optimum. Therefore, careful consideration must be given to the centroid initialization strategy to minimize the chances of biased results.

Once the centroids are initialized, the iterative assignment step begins. In this phase, each IP address is assigned to its nearest centroid based on a distance metric, commonly the Euclidean distance. The assignment is performed iteratively, updating the assignments until convergence is achieved. Through this process, the algorithm forms clusters based on the similarities between the IP addresses.

After the assignment step, the centroid update phase occurs. In this step, the centroids are recalculated by taking the mean of all IP addresses assigned to each cluster. This recalculation ensures that the centroids accurately represent the cluster's central tendency. The assignment and centroid update steps are repeated iteratively until convergence, which is achieved when the assignment of IP addresses to clusters no longer changes significantly.

While the K-Means clustering algorithm is widely used and effective, it does have some limitations that should be taken into account. Firstly, the algorithm is sensitive to the initial centroid selection. Different initializations can lead to different clustering outcomes, emphasizing the importance of carefully choosing the initial centroids. Additionally, the number of clusters must be specified in advance, which can be challenging if the optimal number is unknown.

Moreover, the K-Means algorithm is most suitable for datasets with relatively simple and linear structures. If the data exhibits complex or non-linear patterns, alternative clustering algorithms might be more appropriate. Therefore, it is essential to assess the algorithm's performance and determine if it meets the specific requirements of the web application being analyzed.

In conclusion, the application of the K-Means clustering algorithm can significantly contribute to the detection and analysis of VPN usage in web applications, resulting in improved security and risk management. By carefully following the necessary steps and considering the algorithm's limitations, we can effectively leverage K-Means clustering to gain valuable insights into the presence of VPNs and make informed decisions regarding web application security.

3.2. DevOps Deployment Architecture

3.2.1. Introduction

In our application development process, we have adopted the DevOps methodology, which emphasizes collaboration and automation between software development and operations teams. We are currently applying this methodology to build and deploy a Python Flask application on the AWS Elastic Kubernetes Service (EKS) platform, utilizing the Elastic Container Registry (ECR) for container image storage.

The purpose of this report is to delve into the architecture of our application and explore its essential components in detail. By following the principles of DevOps, we aim to streamline the development, testing, and deployment processes, enabling faster and more reliable software releases.

Our architecture entails several key components that work together harmoniously. The Python Flask application serves as the foundation, providing a robust framework for building web applications. Flask offers flexibility, scalability, and ease of use, making it an ideal choice for our project.

To ensure the scalability and high availability of our application, we have opted for AWS Elastic Kubernetes Service (EKS) as our container orchestration platform. EKS simplifies the management of containerized applications, automating various tasks such as scaling, load balancing, and monitoring. This allows us to focus more on the application logic rather than the underlying infrastructure.

For storing and managing our container images, we utilize the Elastic Container Registry (ECR) offered by AWS. ECR provides a secure and scalable solution for storing, managing, and deploying container images. It integrates seamlessly with EKS, allowing for a smooth deployment process.

Within our DevOps workflow, we emphasize continuous integration and continuous deployment (CI/CD) practices. This involves the use of various tools and automation scripts to facilitate the building, testing, and deployment of our application. Through the implementation of CI/CD pipelines, we can achieve faster feedback cycles, detect and address issues early, and ensure consistent and reliable releases.

In conclusion, our adoption of the DevOps methodology in conjunction with the use of Python Flask, AWS Elastic Kubernetes Service (EKS), and Elastic Container Registry (ECR) has enabled us to build and deploy our application efficiently. This report provides a comprehensive overview of our architecture and its key components, highlighting the benefits of our approach in terms of scalability, reliability, and automation.

3.2.2. Architecture Overview

Our architecture follows a DevOps approach, utilizing various AWS services to simplify the deployment process of our Python Flask application on AWS Elastic Kubernetes Service (EKS) using Elastic Container Registry (ECR).

The Python Flask application serves as the foundation, providing a flexible and scalable framework for building web applications. EKS acts as the container orchestration platform, automating tasks such as scaling and load balancing. ECR handles the storage and deployment of container images securely.

By combining these services within a DevOps framework, we emphasize continuous integration and continuous deployment (CI/CD). This automation streamlines the building, testing, and deployment processes, resulting in faster and more reliable software releases.

In conclusion, our architecture leverages the benefits of DevOps methodologies and AWS services to simplify the deployment of our Python Flask application on EKS using ECR. This approach ensures scalability, reliability, and efficient software delivery.

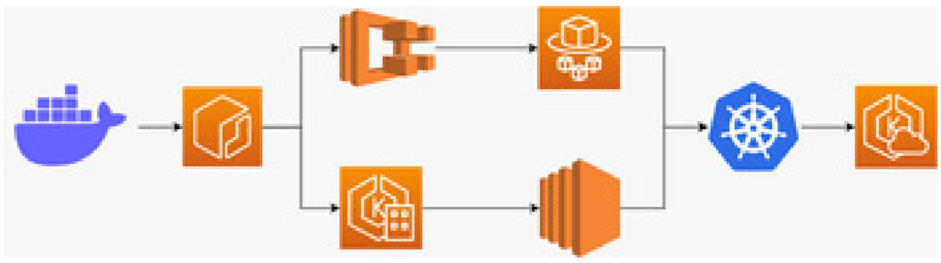


Figure 3.2: AWS Architecture

3.2.3. Python Flask Application

The core component of this architecture is the Python Flask application. A Python Flask application is a web application built using the Flask framework in the Python programming language. Flask is a lightweight and flexible framework that allows developers to create web applications with ease. It provides essential functionalities for routing, handling requests and responses, and rendering templates, making it a popular choice for building simple to medium-sized web applications. Flask's simplicity and minimalistic design make it easy to learn and use, while still offering scalability and extensibility for more complex projects. It is developed using the Flask framework, which allows for the creation of web applications in Python. The application is designed and configured to run in a production environment, ensuring its readiness for deployment.

3.2.4. Docker Image and ECR

A Docker image is created to containerize the Python Flask application, providing a portable software package that includes all the necessary components to run the application, such as code, dependencies, and configurations. ECR, offered by AWS, is a managed service designed for securely storing and managing Docker container images. It offers features like access control, lifecycle policies, and image scanning to ensure the reliability and security of container images. ECR seamlessly integrates with various AWS services, facilitating the deployment of containerized applications on AWS infrastructure. The Docker image, built using the Docker CLI, encapsulates the application and its dependencies, ensuring a consistent and portable environment for deployment.

The Docker image is then pushed to an Elastic Container Registry (ECR) repository. ECR is a fully managed Docker container registry provided by AWS. It acts as a centralized storage location for Docker images, enabling easy distribution and deployment.

3.2.5. Elastic Kubernetes Service (EKS)

AWS Elastic Kubernetes Service (EKS) is a managed service provided by AWS for simplifying the deployment and management of containerized applications using Kubernetes. With EKS, developers can focus on their applications rather than infrastructure management as it automates cluster setup, scaling, and monitoring. EKS seamlessly integrates with other AWS services, providing enhanced functionality and scalability for running Kubernetes-based applications. It simplifies the deployment process by automating the setup and management of Kubernetes clusters. Utilizing EKS, we can effortlessly create and manage a Kubernetes cluster to host our Flask application.

3.2.6. Kubernetes Deployment

In the EKS cluster, Kubernetes is used to create a deployment that defines the desired state of the Python Flask application. Kubernetes automates the deployment, scaling, and management of containerized applications, providing a consistent and portable environment. It handles essential tasks like maintaining high availability and fault tolerance, simplifying the management of complex systems. The deployment specifies the desired number of application replicas, resource requirements, and environment variables. Kubernetes ensures the application's availability and scalability by continuously managing and scaling it according to the desired state.

3.2.7. Production Deployment

Once the Kubernetes deployment is defined, the Python Flask application is deployed to the EKS cluster. The deployment process utilizes the Kubernetes deployment configuration, which specifies the number of replicas to be created, the container image to use, and the environment variables to set. This ensures that the application is running in a production environment and can be scaled dynamically based on demand. When the application is deployed, Kubernetes creates a set of pods, each of which runs a container with the Flask application. The pods are then scheduled onto the nodes in the cluster. By default, Kubernetes will create a single replica of the application. However, if the application experiences increased traffic, Kubernetes can automatically scale the application by creating additional replicas. The application is accessible to users through the Kubernetes Ingress controller. The Ingress controller is responsible for routing traffic to the application pods. The Ingress controller can also be used to configure SSL certificates and load balancing.

3.2.8. Conclusion

In conclusion, the adoption of the DevOps methodology in this architecture offers a streamlined approach to building and deploying Python Flask applications on AWS EKS using ECR. By utilizing containerization with Docker and leveraging ECR and EKS, we attain essential benefits such as scalability, reliability, and security. The architecture facilitates efficient deployment, management, and scaling of the Flask application, making it well-suited for production environments. The combination of DevOps practices and AWS services empowers us to deliver robust and resilient Flask applications with ease.

3.3. FLowchart

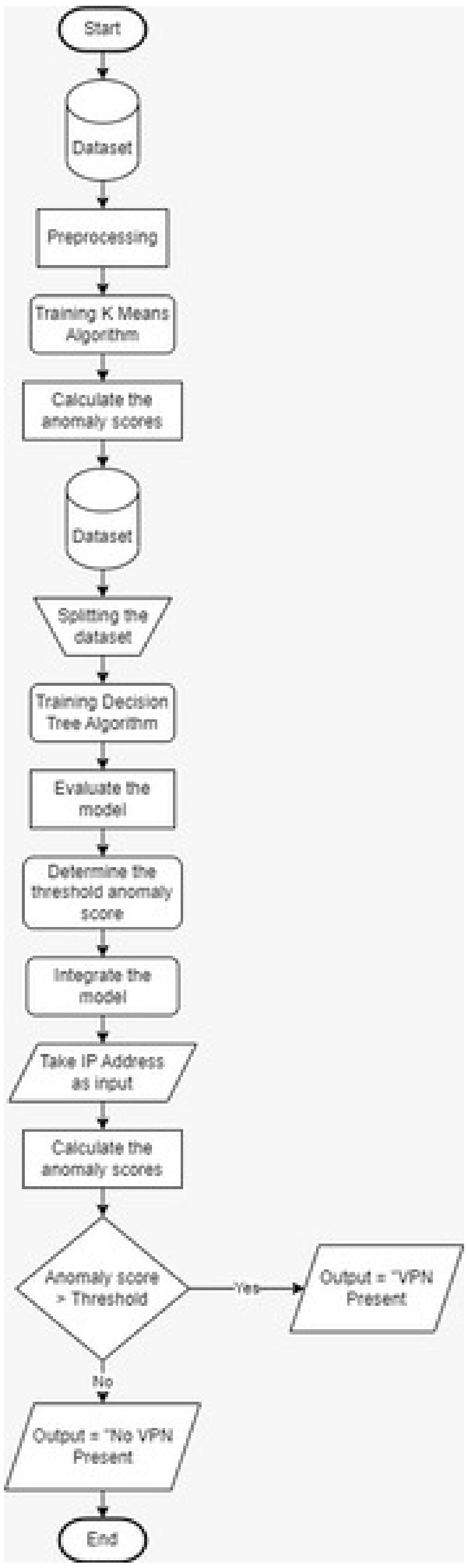

Start
Dataset
Preprocessing
Training K Means Algorithm
Calculate the anomaly scores
Dataset
Splitting the dataset
Training Decision Tree Algorithm
Evaluate the model
Determine the threshold anomaly score
Integrate the model
Take IP Address as input
Calculate the anomaly scores
Anomaly score > Threshold
Yes
Output = "VPN Present
No
Output = "No VPN Present
End

Figure 3.3: Flowchart

3.4. Tools Used

- Programming Languages: Python was the primary programming language used for data preprocessing, machine learning model implementation, and analysis. Additionally, JavaScript, HTML, and CSS were employed for web development.
- Machine Learning Libraries: Python libraries such as scikit-learn, TensorFlow, and PyTorch were utilized for implementing supervised and unsupervised machine learning algorithms, model training, and evaluation.
- VPN Tools: Wireguard, an open-source VPN software, was utilized to create custom VPN tunnels for testing purposes, and Private Internet Access (PIA): PIA was used to create custom proxies for evaluating the accuracy of the model in detecting VPN traffic.
- DevOps Tools, Infrastructure as Code (IaC), and Monitoring Logging: Jenkins, GitLab CI/CD, Terraform, Ansible, Prometheus, Grafana, ELK stack (Elasticsearch, Logstash, Kibana)
- Version Control: Git, along with platforms like GitHub or GitLab, was used for version control and collaborative development.
- IDEs and Text Editors: IDEs like PyCharm, Jupyter Notebook, and text editors such as Visual Studio Code were employed for coding, debugging, and analysis.

3.5. Advantages

- Enhanced Network Security: The project contributes to improved network security by investigating methods to detect VPN traffic used for malicious activities. By accurately identifying and classifying such traffic, organizations can strengthen their defense mechanisms and prevent potential security breaches.
- Informed Decision-Making: The project provides valuable insights into the techniques and tools available for identifying and classifying VPN traffic. This information empowers organizations to make informed decisions regarding their network security strategies and invest in appropriate solutions to mitigate risks.
- Feasibility Assessment: By exploring the feasibility and limitations of different detection techniques and machine learning algorithms, the project assists in evaluating the practicality of implementing IP tracing and traffic classification mechanisms. This assessment helps organizations understand the potential challenges and benefits associated with deploying such solutions.

3.6. Disadvantages

- Limited Dataset Availability: One potential challenge is the availability of diverse and comprehensive datasets containing VPN traffic. Obtaining representative datasets for training and validation purposes can be a hurdle, potentially impacting the accuracy and generalization capabilities of the developed model.
- Complexity of VPN Detection: Detecting VPN traffic is a complex task due to the evolving nature of these technologies and the constant development of evasion techniques. The project may face challenges in keeping up with the advancements and ensuring the model's effectiveness in identifying the latest VPN protocols.
- Ethical and Legal Considerations: The project must address ethical and legal considerations related to privacy and data protection. Collecting and analyzing network traffic data raises concerns about user privacy, data storage, and compliance with data protection regulations. These considerations should be carefully addressed to ensure the project adheres to ethical and legal guidelines.

4. ANALYSIS AND DESIGN

4.1. ER diagram

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. This model is used to define the data elements and relationship for a specified system.

It develops a conceptual design for the database. It also develops a very simple and easy to design view of data. In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

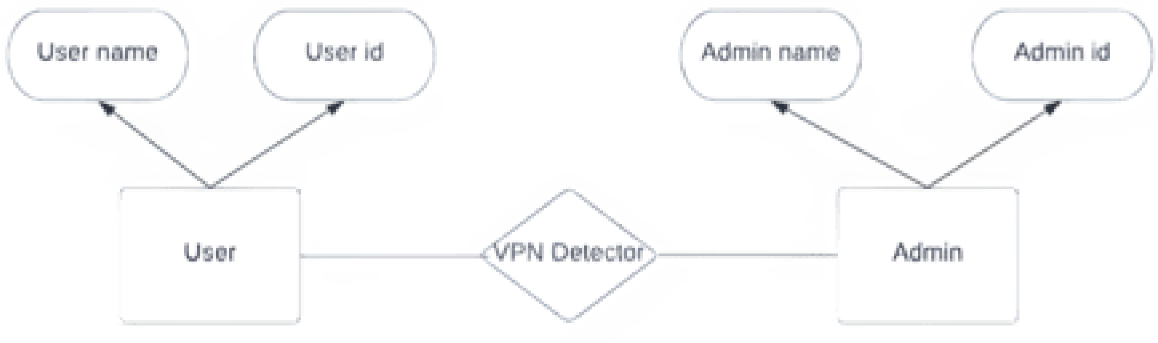


Figure 4.1: ER Diagram

4.2. Class & Object Diagram

The UML Class diagram is a graphical notation used to construct and visualize object oriented systems. A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's:

classes, their attributes, operations (or methods), and the relationships among objects.

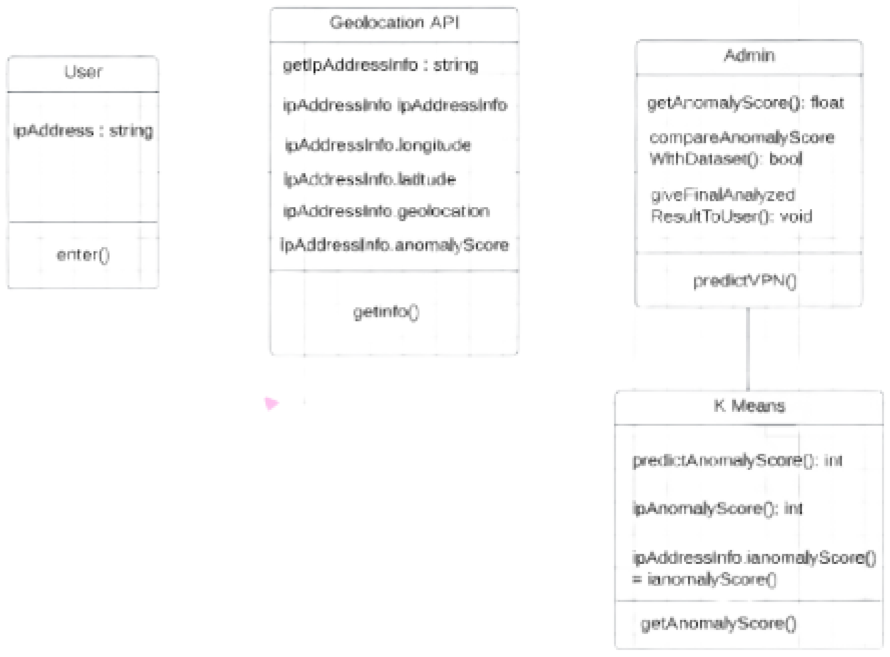

User
ipAddress : string
enter()
Geolocation API
getIpAddressInfo : string
ipAddressInfo ipAddressInfo
ipAddressInfo.longitude
ipAddressInfo.latitude
ipAddressInfo.geolocation
ipAddressInfo.anomalyScore
getinfo()
Admin
getAnomalyScore(): float
compareAnomalyScore WithDataset(): bool
giveFinalAnalyzed ResultToUser(): void
predictVPN()
K Means
predictAnomalyScore(): int
ipAnomalyScore(): int
ipAddressInfo.ianomalyScore() = ianomalyScore()
getAnomalyScore()

Figure 4.2: UML Class Diagram

4.3. Data flow diagram

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

Data Flow Diagram (Level 0)

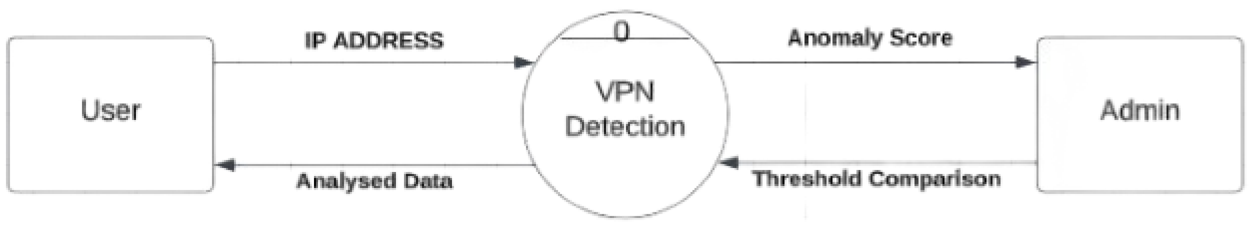


Figure 4.3: Data Flow Diagram Level 0

Data Flow Diagram (Level 1)

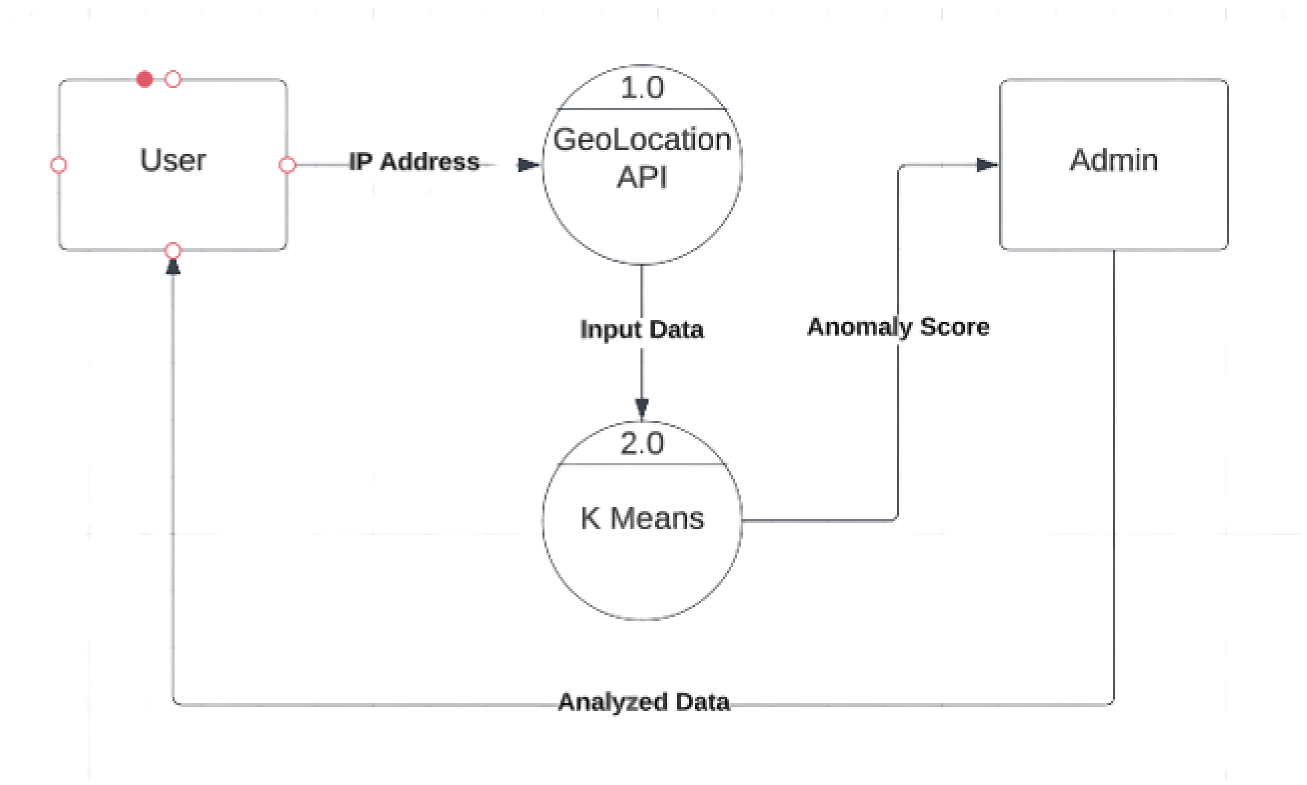


Figure 4.4: Data Flow Diagram Level 1

Data Flow Diagram (Level 2)

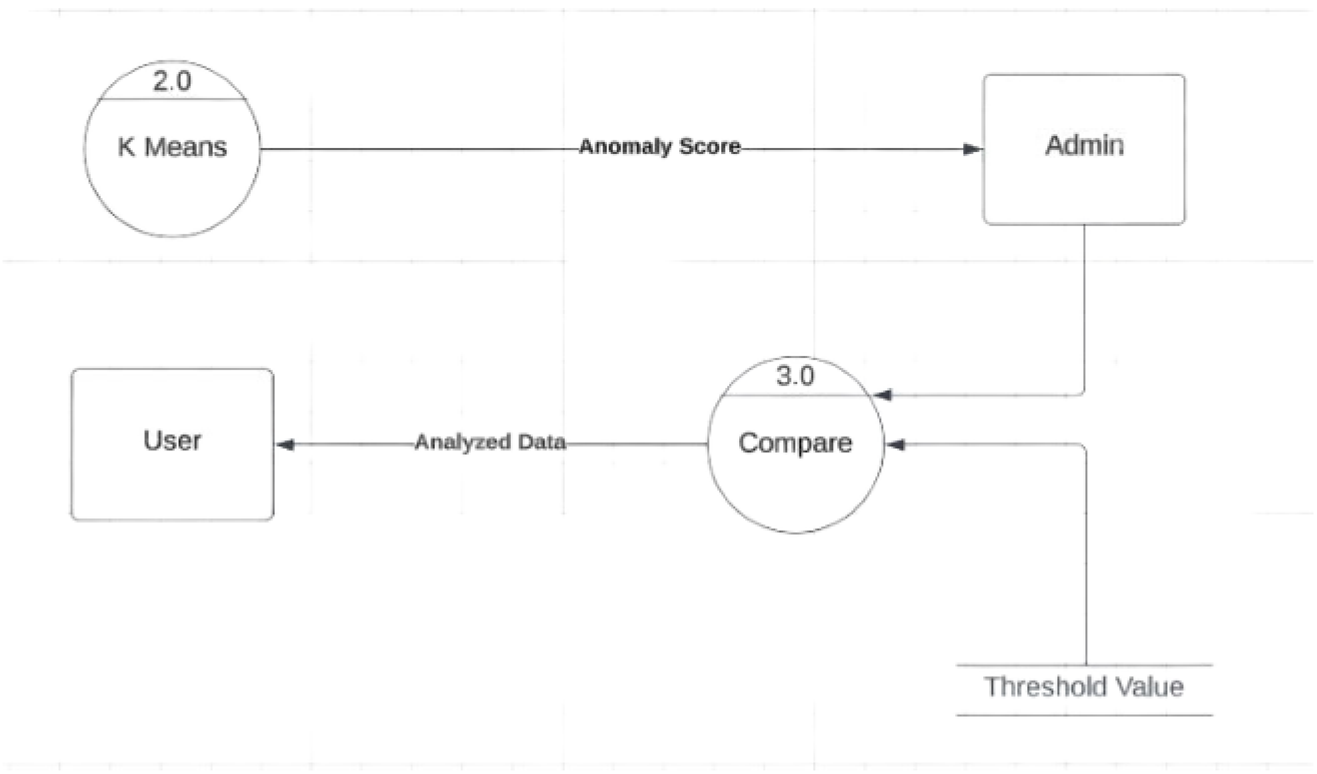


Figure 4.5: Data Flow Diagram Level 2

4.4. UML Use Case Diagram

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

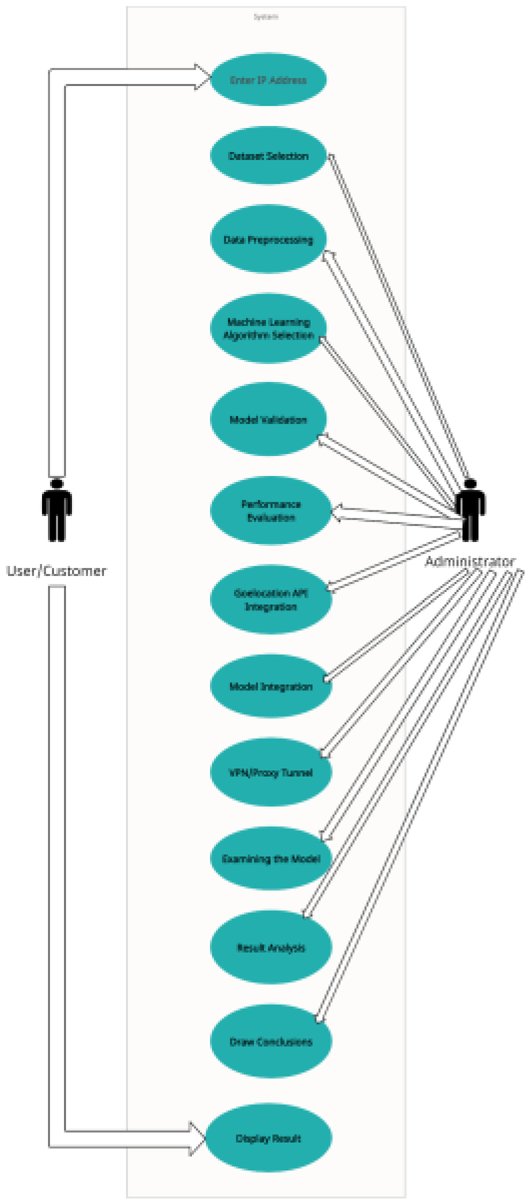


Figure 4.6: Use Case Diagram

4.5. Component Diagram

Component diagrams are used to model the physical aspects of a system. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

Component diagrams are used to visualize the physical components in a system. Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

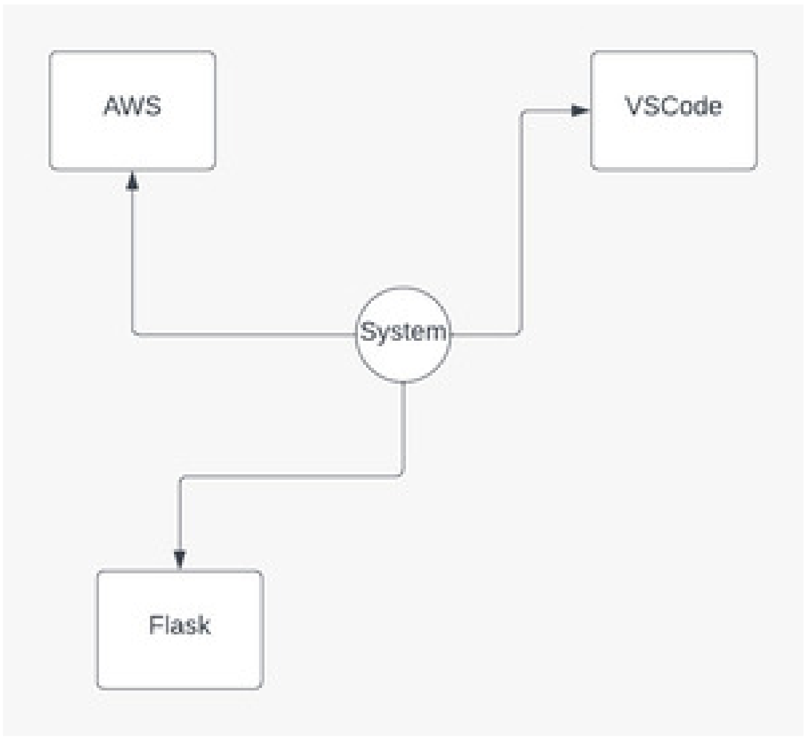


Figure 4.7: Component Diagram

4.6. Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent.

It captures the dynamic behavior of the system. Activity diagram is used to show message flow from one activity to another. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques.

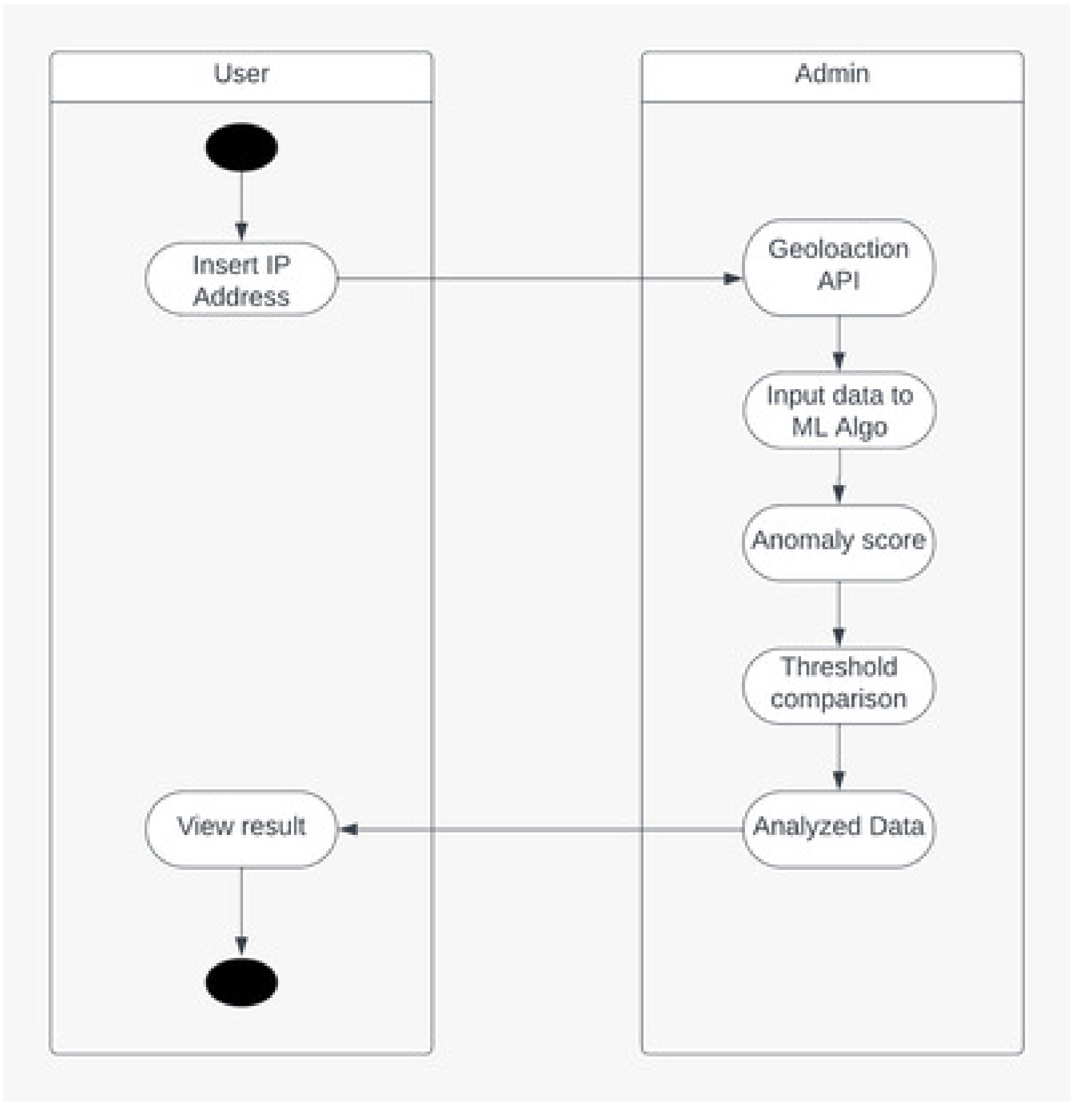


Figure 4.8: Activity Diagram

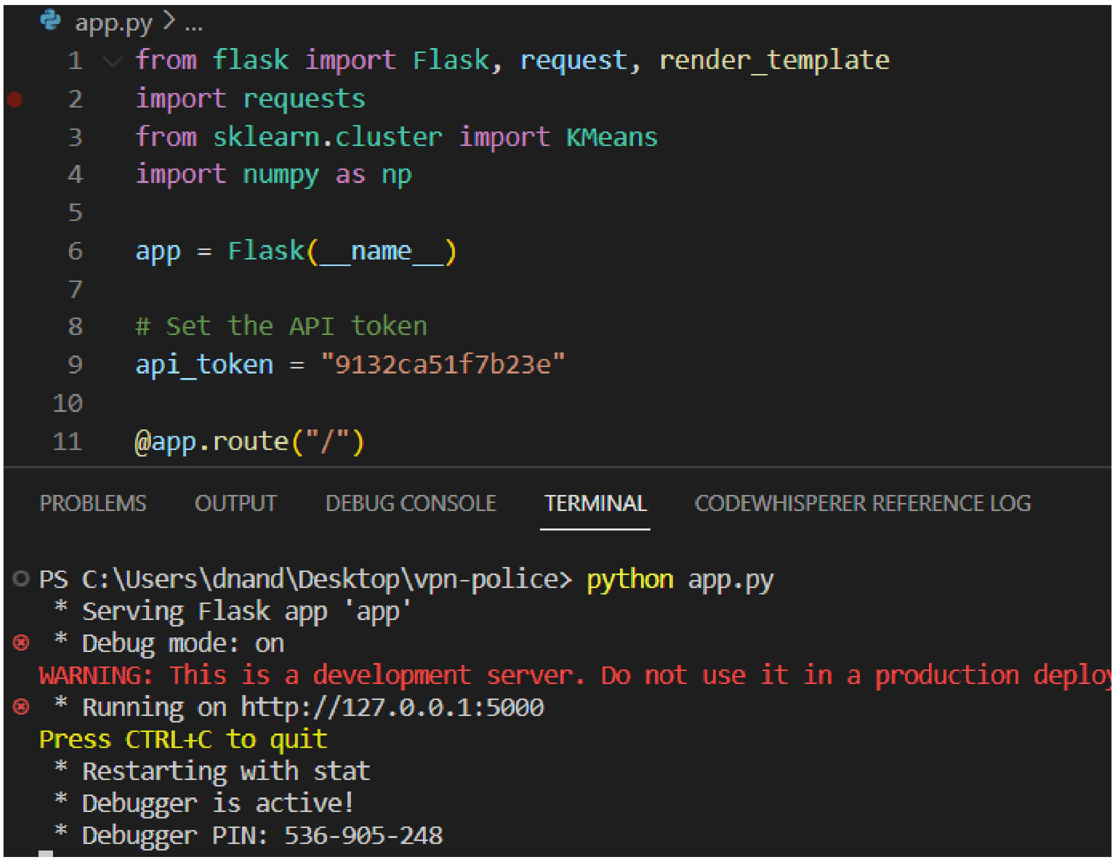
5. RESULTS AND DISCUSSIONS

5.1. Code:

The code uses the Flask framework and several libraries to create a web application. It defines two routes: the home route (”/”) and the result route (”/result”).

The home route renders the ”index.html” template, while the result route handles a POST request. In the result route, the code retrieves an IP address from the request data and makes an API request to ”https://ipinfo.io/” using the provided IP address and API token.

The response from the API is processed to extract relevant information. Anomaly detection is performed on the IP address using K-means clustering. Finally, the ”result.html” template is rendered with the retrieved data and the anomaly score.



```python
from flask import Flask, request, render_template
import requests
from sklearn.cluster import KMeans
import numpy as np

app = Flask(__name__)

# Set the API token
api_token = "9132ca51f7b23e"

@app.route("/")
```

```
PS C:\Users\dnand\Desktop\vpn-police> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deplo
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 536-905-248
```

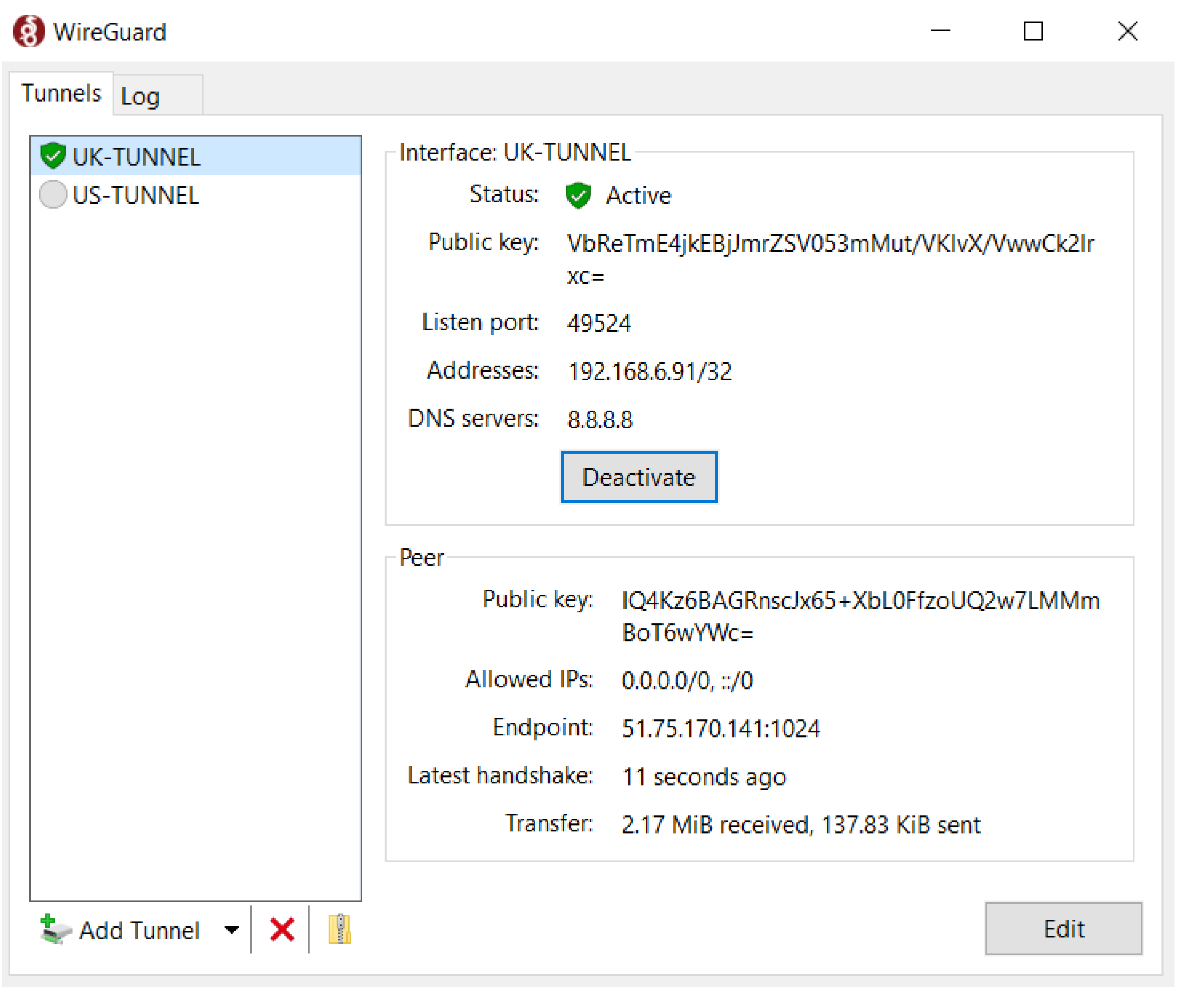
Figure 5.1: Code

5.2. WireGuard VPN Connection:

WireGuard is a VPN protocol that creates secure connections between devices over the internet. It operates at the network layer and establishes virtual network interfaces on the devices.

The key steps involved in WireGuard's operation are:

1) Key Generation: Devices generate private and public keys for secure communication.
2) Configuration Exchange: Devices securely exchange public keys.
3) Secure Tunnels: WireGuard creates virtual network interfaces on sevices.
4) Endpoint Configuration: Devices configure their interfaces with keys and IP addresses.
5) Encryption and Decryption: Data is encrypted using the recipient's public key and decrypted using the private key.
6) Secure Communication: Encrypted data is encapsulated in UDP packets and transmitted over the internet.



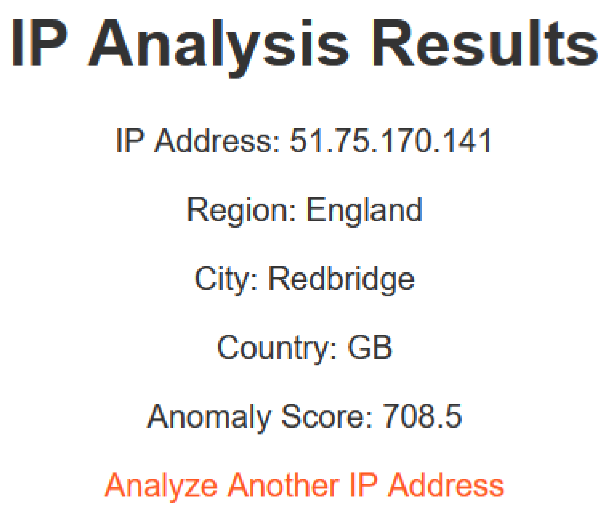
```
Interface: UK-TUNNEL
      Status:  Active
  Public key:  VbReTmE4jkEBjJmrZSV053mMut/VKlvX/VwwCk2lr
               xc=
 Listen port:  49524
   Addresses:  192.168.6.91/32
 DNS servers:  8.8.8.8

Peer
        Public key:  lQ4Kz6BAGRnscJx65+XbL0FfzoUQ2w7LMMm
                     BoT6wYWc=
       Allowed IPs:  0.0.0.0/0, ::/0
          Endpoint:  51.75.170.141:1024
  Latest handshake:  11 seconds ago
          Transfer:  2.17 MiB received, 137.83 KiB sent
```

Figure 5.2: WireGuard VPN connection

5.3. WireGuard Results

The integration of WireGuard into our website yields the following results. The displayed output includes the IP Address and corresponding geographical information, such as region, city, and country. Additionally, our K Means model assigns anomaly scores, which help in determining the presence of VPN connections.

By leveraging WireGuard, we obtain valuable insights into the IP Address and its associated geographic details. These results serve as an effective tool for detecting and analyzing VPN usage on our website.


IP Analysis Results

IP Address: 51.75.170.141

Region: England

City: Redbridge

Country: GB

Anomaly Score: 708.5

Figure 5.3: After connecting to WireGuard VPN

5.4. TOR Connection:

We integrated TOR into our testing process to establish a VPN-enabled IP Address, enabling us to evaluate our website's performance while ensuring secure and anonymous browsing. By leveraging TOR's network of relays, we created a virtual private network that routed our traffic through multiple nodes, making it challenging for anyone to trace back to our original IP Address. This allowed us to thoroughly assess our website's functionality, responsiveness, and overall user experience, while also ensuring the privacy and confidentiality of our connections.

By utilizing TOR, we benefited from the added layer of encryption and the anonymous nature of the network, which helped protect our sensitive data and safeguarded our browsing activities from prying eyes. This approach provided valuable insights into the performance and security aspects of our website, allowing us to address any potential vulnerabilities or areas for improvement. Overall, incorporating TOR into our testing methodology enhanced our ability to analyze our website effectively and ensure its optimal performance for end-users.

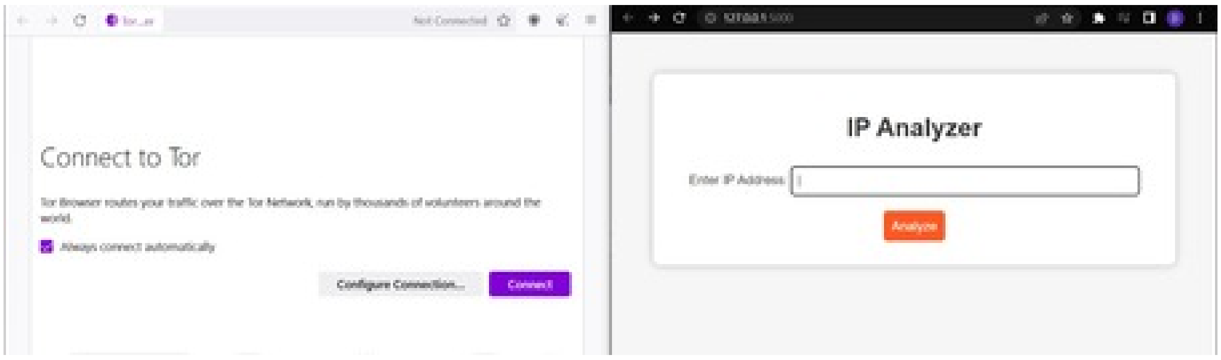

Connect to Tor
Tor Browser routes your traffic over the Tor Network, run by thousands of volunteers around the world.
Always connect automatically
Configure Connection...
Connect
IP Analyzer
Enter IP Address:
Analyze

Figure 5.4: Connecting to TOR

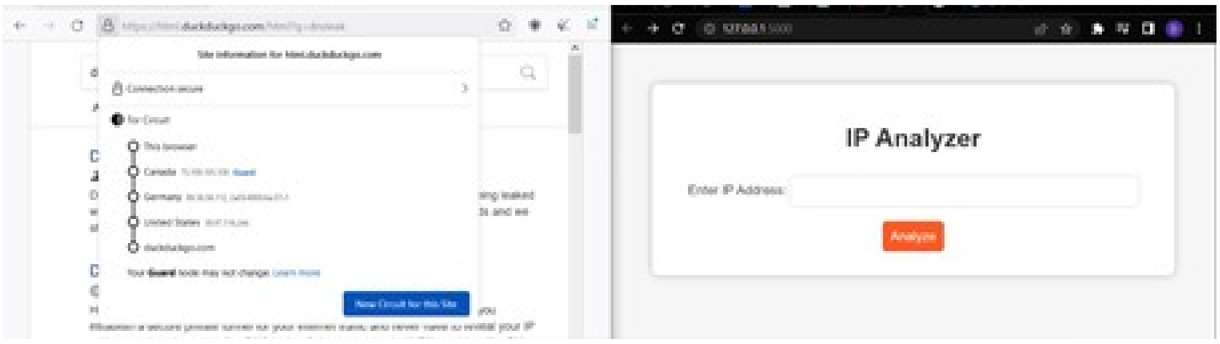
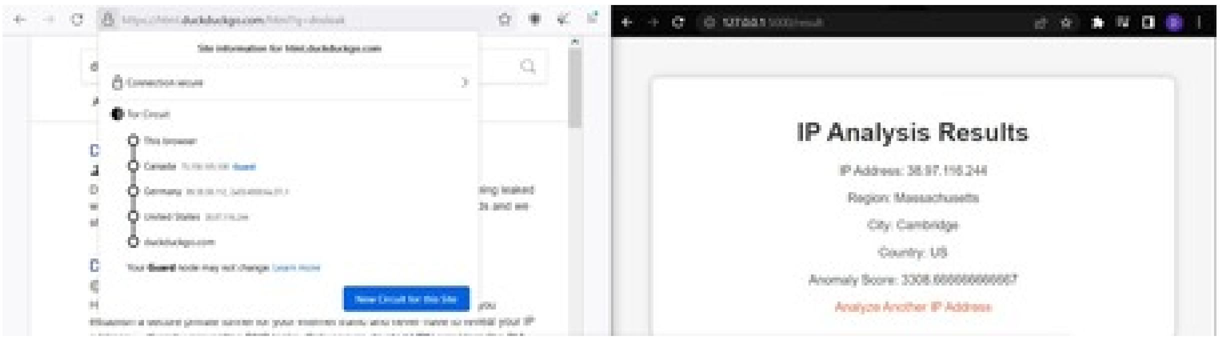


Figure 5.5: Connected to TOR



IP Analysis Results

IP Address: 38.97.116.244

Region: Massachusetts

City: Cambridge

Country: US

Anomaly Score: 3308.666666666667

Analyze Another IP Address

Figure 5.6: Results after connecting to TOR

6. CONCLUSION and FUTURE SCOPE

6.1. Conclusion

In conclusion, this project aimed to investigate methods for detecting VPN advances used to hide an attacker's identity and to develop a method for identifying and classifying VPN traffic. Through an extensive literature review, we identified existing techniques and tools, highlighting the need for improvements and exploration in this field. We selected a suitable dataset containing diverse VPN and non-VPN traffic samples to ensure the model's robustness and accuracy.

By applying data preprocessing techniques and employing machine learning algorithms, including supervised and unsupervised approaches, such as classification, clustering, and anomaly detection, we developed a model capable of accurately classifying VPN traffic. The performance evaluation of the model using metrics like accuracy, precision, recall, and anomaly score demonstrated its effectiveness in detecting and classifying VPN traffic.

The project also involved the creation of custom VPNs using WireGuard and the implementation of custom proxies with Private Internet Access (PIA) and Shadowsocks. These practical implementations provided valuable insights into the real-world effectiveness of the developed model.

Overall, this project contributes to enhancing network security by enabling the identification and classification of VPN traffic, thus empowering organizations to strengthen their defense mechanisms against potential security breaches. However, it is essential to address challenges such as limited dataset availability, the complexity of VPN, and ethical and legal considerations regarding privacy and data protection.

Moving forward, future research should focus on expanding the dataset size and diversity, keeping up with evolving VPN and addressing ethical and legal concerns. The findings of this project can serve as a foundation for further advancements in the field of network security and contribute to the development of effective methods for detecting and mitigating risks associated with VPN.

6.2. Future Scope

To further enhance the VPN detection web application, several avenues for future development and improvement can be explored. Firstly, the integration of multiple machine learning models should be considered, creating an ensemble model by incorporating diverse algorithms such as decision trees, random forests, or support vector machines. This can result in a more accurate and reliable VPN detection system.

Another area of focus is feature engineering and selection techniques. By extracting relevant features from IP addresses, such as geographic information, IP reputation scores, or traffic patterns, the machine learning models can gain a deeper understanding of VPN presence. Selecting informative features can improve the models' discriminatory power and lead to more precise VPN detection.

Furthermore, incorporating historical data analysis can significantly enhance the accuracy and performance of the VPN detection system. By analyzing patterns and trends in historical IP address data related to VPN usage, the models can better recognize and classify VPN traffic. Continuously updating and expanding the historical dataset allows the models to adapt to evolving VPN technologies and improve overall performance.

Real-time data streaming and analysis capabilities are another valuable enhancement. Integrating the VPN detection system with a data streaming platform enables continuous monitoring of incoming IP addresses and instant application of detection algorithms. This real-time capability is particularly crucial for security applications where immediate identification of VPN usage is essential.

Considering the scalability and deployment options of the web application is important as well. As the user base and data volume increase, optimizing system performance and scaling the infrastructure become necessary. Exploring cloud-based solutions or distributed computing frameworks can handle larger volumes of IP address data and ensure a responsive and reliable application.

By pursuing these future directions, the VPN detection web application can evolve and improve its capabilities in identifying VPNs in IP addresses. These enhancements will contribute to a more accurate, adaptable, and efficient system that caters to the evolving needs of users and stays ahead of emerging VPN technologies.

References

[1] S. Miller, K. Curran, and T. Lunney, “Detection of virtual private network traffic using machine learning,” *International Journal of Wireless Networks and Broadband Technologies*, vol. 9, pp. 60–80, 07 2020.

[2] A. Goel, A. Kashyap, B. D. Reddy, R. Kaushik, S. Nagasundari, and P. B. Honnavali, “Detection of vpn network traffic,” pp. 1–9, 2022.

[3] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, “Characterization of encrypted and vpn traffic using time-related features,” 2016.

[4] A. Almomani, “Classification of virtual private networks encrypted traffic using ensemble learning algorithms,” *Egyptian Informatics Journal*, vol. 23, 2022.

[5] M. A.-F. Mustafa Al-Fayoumi and S. Nashwan, “Vpn and non-vpn network traffic classification using time-related features,” *Computers, Materials Continua*, 2022.

[6] M. P. Saeed Seraj, Siavash Khodambashi and N. Polatidis, “Mvdroid: an android malicious vpn detector using neural networks,” *Neural Comput Appl.*, pp. 1–11, 4 2023.

[7] J. L. C. S. Weishi Sun, Yaning Zhang and S. Zhang, “A deep learning-based encrypted vpn traffic classification method using packet block image,” 2022.

[8] D. M. B. S. Y. M. Ms. Shipra Srivastava, Harigovind H, “Detection of vpns using machine learning,” *International Journal for Research in Applied Science Engineering Technology (IJRASET)*, vol. 10, 05 2022.

[9] M. J. Devishree Naidu, “Detection technique to trace ip behind vpn/proxy using machine learning,” *International Journal of Next-Generation Computing (IJNGC)*, vol. 14, 02 2023.

[10] A. Almomani, “Classification of virtual private networks encrypted traffic using ensemble learning algorithms,” *Egyptian Informatics Journal*, 06 2022.

[11] E. K. S. C. B. . J. S. Sikha Bagui, Xingang Fang, “Comparison of machine-learning algorithms for classification of vpn network traffic flow using time-related features,” *Journal of Cyber Security Technology*, vol. 1, 06 2017.

[12] J. D. K. d. S. A. R. V. R. N. D. A. A. Steven Jorgensen, John Holodnak and A. Wollaber, “Extensible machine learning for encrypted network traffic application labeling via uncertainty quantification,” 03 2023.

[13] Y. Haribhakta and N. G. Evans, “An empirical study of vpn services: Characteristics, privacy, and vpn detection,” *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, 05 2016.

[14] C. D. G. Leaden, M. Zimmermann and A. G. Labouseur, “An api honeypot for ddos and xss analysis,” *IEEE MIT Undergraduate Research Technology Conference (URTC)*, 2017.

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14]