Computer Networks and Communications
https://ojs.wiserpub.com/index.php/CNC

Research Article

Performance Evaluation of HTTP, DHCP and DNS Protocols of Data Packets for Vulnerabilities Using the Isolation Forest Algorithm

Tawo Godwin A[1], Ayansi Francis E[2], Faith Praise O[3], Osahon Okoro O[4], Vincent N Ogar[5*]

[1]Department of Electrical/Electronic, Faculty of Engineering, University of Cross River State, Calabar, Nigeria
[2]Department of Electrical/Electronic, Faculty of Engineering, Ambrose Alli University, Ekpoma, Nigeria
[3]Department of Computer Engineering, Faculty of Engineering, University of Calabar, Calabar, Nigeria
[4]Department of Computer Science, Faculty of Science, University of Calabar, Calabar, Nigeria
[5]Department of Electronic and Electrical Engineering, James Watt School of Engineering, University of Glasgow, UK
E-mail: v.ogar.1@research.gla.ac.uk

Received: 26 August 2024; **Revised:** 23 September 2024; **Accepted:** 10 October 2024

Abstract: In the contemporary digital landscape, network security is paramount to safeguard data integrity and prevent unauthorized access as data have been structured in the network through protocols. In term of data structure IPv6 protocol has extended data size due to the robust addresses of 128 bits as compare to 32 bits of IPv4. Due to the improved security data structure of IPv6, the study focuses on identifying vulnerabilities in HTTP, DHCP, and DNS packets using the Isolation Forest algorithm approach, a machine-learning technique designed for anomaly detection. By analyzing packet lengths, size, payload and addressing, the study visualizes normal and anomalous behavior, providing insights into potential security threats in IPv4 network structure. The results highlight the effectiveness of Goodput, Quality of service and risk as essential factors in the network, using Little's theorem analysis and the Isolation Forest in detecting anomalies across these different network protocols, offering valuable implications for network security structures, due to IoT in recent networks. The time response determination in this paper explained details information on the time the treats entered the network, the duration of the vulnerabilities within the network, leading to a certain threshold, and traffic delay factors due to deviation of packet length and other social engineering activities. Sensitive multipurpose security devices are involved; MikroTik routers were configured and installed in the network under evaluation. Which the normal DPI technique was unable to effectively and efficiently addressed. ADPI principles and operations where the needed security measures adopted to detect those vulnerabilities which were eventually addressed and have contributed to recent measures of network security.

Keywords: anomaly detection, Little's theorem, Isolation Forest, network security

1. Introduction

The proliferation of IoT devices and the increasing complexity of network protocols have heightened the risk of security breaches in modern networks. Protocols such as HTTP, DHCP, and DNS are essential for network communication but are also vulnerable to a wide range of attacks. This paper explores the use of the Isolation Forest algorithm for detecting anomalies and vulnerabilities in these protocols. By analyzing packet size, Goodput, Quality of Service (QoS), and Risk, we aim to detect deviations in network traffic that could signify potential security threats. Due to transition mechanism from IPv4 to IPng, security structures of the existing system are exposed to recent IPng, and intruders have their access

DOI: https://doi.org/10.37256/cnc.2220245573

through protocols, because packets length are similar as seen in Figure 1 and can be identified as vulnerabilities. A network that incorporate IPv4 and IPng devices, either dual stack, IPv4 to IPv6, Manual tunneling or translator being as transition techniques used, the security of IPv4 cannot definitely have effective control over IPng. NAT and PAT security architectures cannot comb treats within IPng, there is a need for ADPI approach to effectively control recent activities within the network.

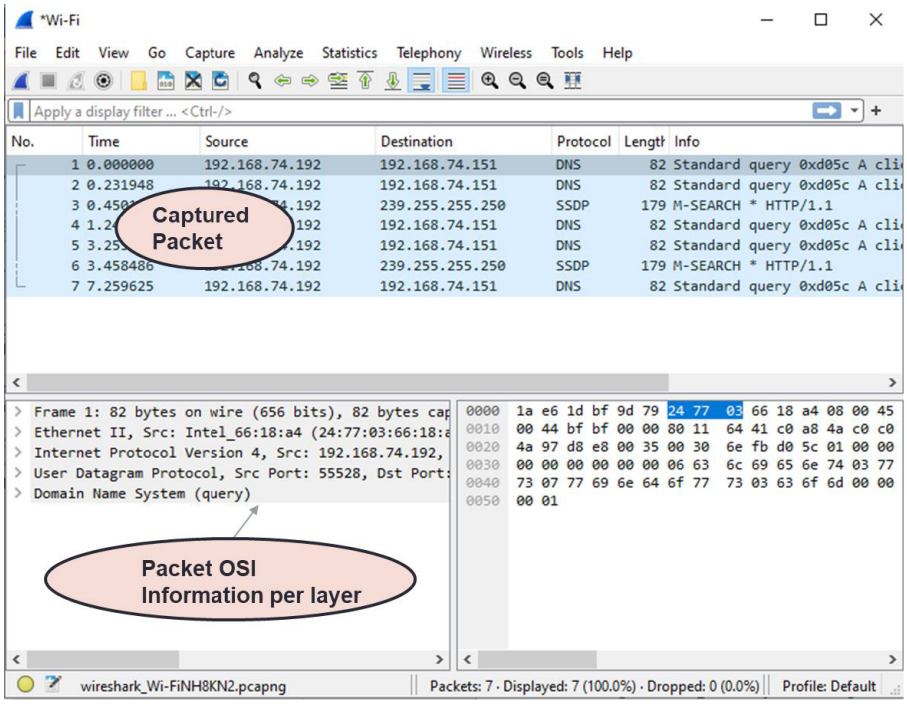

Figure 1. Wireshark graphical user interface for network analysis

Existing studies have applied machine learning techniques to detect anomalies in network traffic, yet there remains a gap in understanding how metrics like Goodput and Risk contribute to real-time anomaly detection. This study bridges that gap by offering a detailed evaluation of these metrics in the context of HTTP, DHCP, and DNS protocols. The primary contributions of this work are as follows:

- Application of the Isolation Forest algorithm for detecting anomalies in HTTP, DHCP, and DNS protocol traffic.
- Evaluation of time response and expressions for network efficiency; Goodput, QoS, and Risk in network protocols, utilizing Little's Theorem for performance analysis were the focal point of the research.
- Comprehensive analysis of packet data and feature selection to enhance anomaly detection. Insights into network vulnerabilities arising from anomalies in protocol traffic.

2. Review works

Network anomaly detection has evolved significantly, with various machine learning techniques being employed. Federated learning for network attack detection using attention-based graph neural networks and Graph-based deep learning for communication networks are among the latest studies emphasizing the application of machine learning in enhancing network security. These studies provide frameworks for anomaly detection but often lack specific applications for individual network protocols such as HTTP, DHCP, and DNS. In particular, previous work on the Isolation Forest

algorithm for network anomaly detection has demonstrated its effectiveness in identifying rare events, but there has been limited application to protocol-specific traffic. This study expands on prior research by applying the Isolation Forest to detect vulnerabilities in common network protocols.

Early Research on Anomaly Detection (1990s–2000s):

Dorothy Denning's seminal work on anomaly detection laid the foundation for intrusion detection systems (IDS). She proposed a model for real-time intrusion detection, emphasizing the importance of monitoring system activities for anomalies that could indicate security breaches [1]. Vern Paxson developed Bro, a network intrusion detection system that utilized real-time analysis of network traffic to detect intrusions. Paxson's work highlighted the potential of packet-level analysis for identifying abnormal behaviors in network traffic [2].

Density-Based Anomaly Detection (2000–2010):

The introduction of the Local Outlier Factor (LOF) by Breunig and colleagues marked a significant advancement in anomaly detection. LOF identified density-based local outliers, providing a robust method for detecting anomalies in high-dimensional data [3]. The Isolation Forest algorithm, introduced by Liu, Ting, and Zhou, presented a novel approach to anomaly detection by isolating observations in a dataset. This method proved efficient and effective, particularly for high-dimensional data and large datasets [4].

Network Traffic Analysis and Machine Learning (2010–Present):

Zuech, Khoshgoftaar, and Wald conducted a comprehensive survey on intrusion detection and big heterogeneous data. Their work emphasized the integration of machine learning techniques in analyzing large-scale network traffic data for anomaly detection [5]. Kim and colleagues applied deep learning techniques to network anomaly detection, showcasing the potential of neural networks in identifying complex patterns in network traffic. Their work demonstrated significant improvements in detection accuracy compared to traditional methods [6].

The comprehensive review by Garcia-Teodoro and colleagues on anomaly-based network intrusion detection systems provided an in-depth analysis of various anomaly detection techniques, highlighting their strengths and limitations. Their work underscored the importance of continuous advancements in anomaly detection methodologies to address evolving cyber threats [7].

Recent Advances and Applications (2020–Present):

Kwon and colleagues explored the application of machine learning algorithms, including Isolation Forest, for detecting anomalies in Internet of Things (IoT) networks. Their work highlighted the growing relevance of anomaly detection in securing IoT ecosystems [8]. Shiravi and colleagues developed a framework for anomaly detection in network traffic using ensemble learning techniques. Their approach combined multiple machine learning algorithms to improve detection accuracy and robustness [9]. Zhao and colleagues proposed a hybrid anomaly detection method that combined statistical analysis with machine learning techniques. Their work demonstrated the effectiveness of hybrid approaches in enhancing anomaly detection performance in complex network environments [10].

3. Methodology

3.1 *Data collection*

Network traffic data was collected for HTTP, DHCP, and DNS protocols over a period of several days. The data includes packet size, packet length, timestamps, and payloads. This dataset was preprocessed to remove outliers and ensure data quality.

3.2 *Isolation forest algorithm*

The Isolation Forest algorithm, a machine learning model known for its anomaly detection capabilities, isolates anomalies by constructing random decision trees. The algorithm's advantage lies in its ability to detect anomalies in datasets where the majority of data points represent normal behavior, and only a small percentage are outliers. For this study, we trained the Isolation Forest model using packet size as the primary feature.

3.3 *Feature selection and justification*

Packet size was selected as the primary feature for anomaly detection due to its strong correlation with the occurrence of anomalies in network traffic. While features such as traffic frequency and port usage were considered, packet size provided a more direct indication of abnormal behavior in protocol communication. For example, unusually large or small packet sizes can signal potential attacks such as Distributed Denial of Service (DDoS) or buffer overflow attacks.

3.4 *Python code implementation*

Network traffic data was captured using Wireshark, a widely used network protocol analyzer. The captured data was segmented by the following protocols (HTTP, DHCP, and DNS) and saved as CSV files (see Supplementary Materials) for analysis. The segmentation of the dataset is facilitated by a Python script designed to automate the data preprocessing task. The script utilizes programming constructs to penetrate through the dataset, extract protocol information from packet headers, and organize packets into separate sub-data segment for each protocol. The Python code in Appendix A demonstrates the implementation of dataset segmentation by protocol. It reads the network traffic dataset from a CSV file, identifies unique protocol types, filters the dataset for each protocol, and saves the protocol-specific data into separate CSV files. By executing this Python script, the dataset is effectively preprocessed and segmented into sub-data corresponding to different protocols, laying the groundwork for in-depth analysis and evaluation of network security measures.

3.5 *Time response of the security devices and performance*

The response time of security devices, such as firewalls and intrusion detection systems (IDS), is crucial to maintaining robust network security. This subsection examines the time it takes for these devices to detect and respond to potential threats, as well as their impact on network performance.

Performance Metrics: The response times of firewalls, IDS, and other security appliances are measured based on their ability to detect and mitigate threats efficiently. Detection time, alert generation, and the overall reaction time of these devices are assessed to determine their effectiveness.

Impact on Network Performance: While security devices are essential for protecting the network, their presence can introduce latency. As these devices process and filter traffic, the delay may impact real-time applications and throughput. To manage this, optimized configuration time response.

Time to detect (T_D):

$$T_D = T_{Thr} - up - T_{Sa} \tag{1}$$

The time to detect is the time from the moment the attack starts (T_{Sa}) until the moment the attack is detected $(T_{Thr} - up)$, which is the time when the service metrics threshold is crossed.

Time to implement (T_i):

$$T_i = T_{cm} - impt - T_{Thr} - up \tag{2}$$

The time to implement is the time elapsed from the moment the attack is detected until the moment the implementation of the countermeasure is completed $(T_{cm} - impt)$.

Time to recover (T_r):

$$T_r = T_{Thr} - down - T_{cm.impt} \tag{3}$$

The time to recover is the time elapsed from the moment the countermeasure is implemented to the moment until the service metrics are recovered, and the threshold is passed in the other direction $(T_{Thr} - down)$.

In terms of the control loop, T_D is the time it takes in the detecting phase from the moment there is a trigger to the moment the control loop moves to the next phase. T_i is the time that the control loop spends in the analyzing and the decide phases plus the time spent in the Respond phase until the moment the countermeasure is in effect. Finally T_r is the time spent in the Respond phase until the moment the attack is stopped or been mitigated. Once more, the effectiveness considerations are not just relevant for our SFMIS architecture; the results are generalizable in other SIFMIS-based systems architecture. then in essence, can provide the basis for a standardized and agreed upon set of metrics when comparing different network systems.

In measuring network security effectiveness through packet analysis. The proposed approach leverages key metrics derived from packet capture data to evaluate the efficiency of network security countermeasures. Drawing inspiration from [8], this section outlines the process of estimating effectiveness based on variables such as Total Time of Packet Capture (T_{TPC}), Time per Packet Captured (T_{PP}) and Time before Browser (T_{BB}). Measurement of these key variables involves: whose values are specified in Table 1.

Table 1. Details data examination using Python software

S/N	Time to Detect Vulnerability T_d (sec)	Time the Attack Start T_{tsa} (sec)	Threshold Time T_{th} (sec)	Length of Packet L_{packet} (MB)	T_{tpc} (sec)	T_{bb} (sec)
1	7:10:31	—	—	255	0.0031	0.0031
2	4:16:08	—	—	255	0.0077	0.0008
3	4:50:17	4:50:17		228	0.0033	treats
4	4:54:05	4:54:05		228	0.0089	Treats
5	4:59:10	4:59:10	0:72:85	228	0.0055	Treats
6	5:02:45	5:02:45		228	0.0063	Treats
7	5:05:40	5:05:4		228	0.0040	Treats
8	5:23:03	5:23:03		216	0.0027	Treats
9	5:24:30	—	—	255	0.0076	0.0030
10	5:25:06	5:25:06	0:00:97	246	0.0054	Treats
11	10:09:52	—	—	255	0.0049	0.0052
12	10:10:30	10:10:3	0:00:78	246	0.0093	Treats
13	3:59:23	—	—	255	0.0022	0.0023
14	4:00:01	4:00:01	0:64:78	246	0.0001	0.0001

Total Time of Packet Capture (T_{TPC}):

$$T_{TPC} = T_{LP} - T_{FP} \tag{4}$$

T_{TPC} is calculated as the total time taken to capture packets in the dataset. It is determined by subtracting the timestamp of the first packet from the timestamp of the last packet, T_{TPC} is equal to the Time of Last Packet (T_{LP}) minus the Time of First Packet (T_{FP})

Time per Packet Captured (T_{PP}):

This represents the time elapsed before the next packet is captured. For a set of consecutive packets, T_{FP} is calculated as the time difference between each successive packet (T_{PPi}), is equal to the Time of Packet $((n) - 1)$ which is;

$$T_{PPi} = T_{PPn} - T_{PPn} - 1 \tag{5}$$

Where: T_{PPn} is the different value of the next packet and $T_{PPn} - 1$ is the previous value.

Time before Browser (T_{bb}):

This is the measures of the time taken before specific vulnerabilities (T_{bbi}), such as browser-related exploits, are detected in network traffic. It is determined by subtracting the timestamp of the first vulnerable packet from the timestamp of the first packet in the dataset, [8], T_{bbi} is equal to the time of First Packet Vulnerable (T_{FV}) minus the time of First Packet (T_{Bd}).

$$T_{bbi} = T_{FV} T_{Bd} \tag{6}$$

Evaluation of the Effectiveness of the network efficiency (η):

Based on the measured variables, the effectiveness of network security countermeasures can be estimated using the following approach, as proposed by [8]:

$$\eta = f\left(T_{tpc} + T_{pp} + T_{bb}\right) \tag{7}$$

The effectiveness of countermeasures is a complex value influenced by the footprint of the attack and the timing of the response. Shorter duration of T_{tpc}, T_{pp} and T_{bb} values indicate quicker detection and response to security threats, suggesting higher effectiveness. By quantifying the key variables such as T_{tpc}, T_{pp} and T_{bb} organizations can assess the efficiency of their security measures and identify areas for improvement. The proposed approach offers a structured framework for evaluating and optimizing network security countermeasures.

This choice was based on the assumption that significant deviations in packet length could indicate abnormal behavior.

Anomaly Detection:

The Isolation Forest algorithm was applied to each protocol dataset to identify anomalies. Packets were classified as normal or anomalous based on their isolation scores.

3.6 *Application of Little's theorem*

$$Little's\ Theorem,\ represented\ as\ L = \lambda W \tag{8}$$

where L is the average number of items in the system, λ is the arrival rate, and W is the average waiting time, was applied to evaluate Goodput, Quality of Service, and Risk. In the context of network security, Little's Theorem allows us to quantify the relationship between packet arrival rates and time spent within the network system. This enables us to assess the impact of network anomalies on overall performance.

Goodput is defined as the rate at which useful data is successfully delivered over the network. In terms of anomaly detection, a decline in Goodput often signals increased network congestion or data tampering. QoS metrics, such as latency and jitter, further indicate the quality of communication between devices, with sudden deviations suggesting potential anomalies

3.6.1 *Deploying Little's theorem expression for network performance*

Goodput (GP): is the measure of useful transmitted data in a network, applying Little's theorem expression;

$$L_{GP} = \lambda_{GP} W_{GP} \tag{9}$$

Therefore;

$$GP = \frac{L_{GP}}{\lambda_{GP}} \tag{10}$$

where L_{GP} the average number of useful packets in the system, λ_{GP} is the arrival rate of useful packets and W_{GP} is the average time useful packets spend in the system.

Quality of Service (QS): is the overall performance of a network seen by users.

Let

$$L_{QS} = \lambda_{QS} W_{QS} \tag{11}$$

Hence;

$$QS = \frac{L_{QS}}{\lambda_{QS}} \tag{12}$$

where L_{QS} is the average number of serviceable packets in the system, λ_{QS} is the arrival rate of serviceable packets and W_{QS} is the average time the serviceable packets spend in the system.

Risk (RK): involves potential loss or degradation of service due to packets abnormalities in a network. hence

$$L_{RK} = \lambda_{RK} W_{RK} \tag{13}$$

Therefore

$$RK = \frac{L_{RK}}{\lambda_{RK}} \tag{14}$$

where L_{RK} the average number of risky packets in the network, λ_{RK} is the arrival rate of risky packets and W_{RK} is the average time risky packets spend in the network. by integrating Little's theorem into our network performance analysis, we can quantitatively assess the performance and identify areas for improvement. The Isolation Forest algorithm helps detect anomalies affecting these metrics, allowing us to take proactive measures to enhance network reliability and efficiency. This combined approach provides a comprehensive framework for monitoring and improving network performance in real-time. Selecting relevant features for the Isolation Forest algorithm involves choosing the attributes or columns from the dataset that are most informative for detecting anomalies. These features should be those that best represent the behavior and characteristics of the network traffic data we are analyzing. In the process of implementing the Isolation Forest algorithm to detect anomalies in network traffic, selecting relevant features is crucial. This selection process ensures that the algorithm focuses on the most informative aspects of the data, thereby enhancing its accuracy and efficiency. The steps to achieve this are outlined below:

- Understanding the Data: The first step in selecting relevant features involves a thorough examination of the dataset. This process requires understanding what each feature represents. Network traffic datasets typically include features such as packet size, protocol type, source and destination IP addresses, port numbers, timestamps, and various header fields. By comprehensively understanding the nature and significance of these features, one can identify those that are essential for anomaly detection.
- Utilizing Domain Knowledge: Leveraging domain knowledge about network behavior is instrumental in identifying features likely to indicate normal and abnormal patterns. For instance, unusually high or low packet sizes, unexpected protocols, or an abnormal frequency of requests can be indicative of anomalies. This domain expertise guides the selection of features that are most relevant for detecting deviations from normal network behaviour. Packet length was selected as the primary feature for anomaly detection based on its historical significance in network security analysis. Previous studies have consistently demonstrated that deviations from normal packet length patterns can be indicative of various attacks, such as buffer overflows, DoS attacks, and protocol violations. Additionally, packet length is a fundamental characteristic of network traffic that is easily measurable and directly related to the behaviour of network protocols.
- Feature Correlation: The next step involves using statistical methods to check the correlation between features. Features with a high correlation to the target variable (whether an observation is an anomaly or not) are often more relevant. However, it is important to avoid using highly correlated features together, as this can lead to redundancy and diminish the model's performance. By analyzing the correlation matrix, one can select a subset of features that

provide the most unique and informative insights. The calculated performance metrics provide valuable insights into the overall health and security of the network. High Goodput values indicate efficient data transmission, while low Quality of Service scores may suggest network congestion or latency issues. Elevated Risk values signal potential vulnerabilities or threats. By analyzing these metrics in conjunction with the detected anomalies, organizations can identify areas for improvement and implement targeted security measures.

- Feature Importance: To further refine the selection process, algorithms like Random Forest can be utilized to determine feature importance. Random Forest provides a ranking of features based on their contribution to model predictions. Features that are found to contribute the most to the model's decisions are considered highly relevant. This step ensures that the most significant predictors of anomalies are included in the final feature set.
- Dimensionality Reduction: Techniques such as Principal Component Analysis (PCA) can also be employed to reduce the number of features while retaining the most informative aspects of the data. PCA transforms the original features into a new set of uncorrelated components, which capture the maximum variance in the data. This dimensionality reduction technique helps in simplifying the dataset without losing critical information, thereby improving the performance of the Isolation Forest algorithm.

3.6.2 *Pseudocode for network traffic analysis with isolation forest*

1. Import necessary libraries
    - import pandas as pd
    - from sklearn.ensemble import IsolationForest
2. Load the dataset
    - data = pd.read_csv('network_traffic_data.csv')
3. Extract protocol-specific data
    - http_data = data[data['protocol'] == 'HTTP']
    - dns_data = data[data['protocol'] == 'DNS']
    - browser_data = data[data['protocol'] == 'BROWSER']
4. Define the relevant features for Isolation Forest
    - features = ['packet_size', 'src_port', 'dst_port', 'flags']
5. For each protocol (HTTP, DNS, BROWSER):
    a. Extract feature data
        - X_protocol = protocol_data[features]
    b. Train the Isolation Forest model
        - iso_forest_protocol = IsolationForest(contamination=0.1)
        - protocol_data['anomaly'] = iso_forest_protocol.fit_predict(X_protocol)
    c. Calculate Little's Theorem parameters
        - L_GP_protocol = protocol_data['packet_size'].mean()
        - lambda_GP_protocol = protocol_data['src_port'].mean()
        - W_GP_protocol = L_GP_protocol/lambda_GP_protocol

- L_QS_protocol = protocol_data['dst_port'].mean()
- lambda_QS_protocol = protocol_data['flags'].mean()
- W_QS_protocol = L_QS_protocol/lambda_QS_protocol
- L_RK_protocol = protocol_data['packet_size'].mean()
- lambda_RK_protocol = protocol_data['dst_port'].mean()
- W_RK_protocol = L_RK_protocol/lambda_RK_protocol

d. Save protocol-specific data with anomalies marked

- protocol_data.to_csv('protocol_data_with_anomalies.csv', index=False)

e. Analyze anomalies

- anomalies_protocol = protocol_data[protocol_data['anomaly'] == −1]
- print("Number of anomalies detected in PROTOCOL: ", len(anomalies_protocol))

f. Calculate network performance metrics for protocol

- GP_protocol = L_GP_protocol/lambda_GP_protocol
- QS_protocol = L_QS_protocol/lambda_QS_protocol
- RK_protocol = L_RK_protocol/lambda_RK_protocol
- print(f'PROTOCOL—Goodput (GP): {GP_protocol}")
- print(f'PROTOCOL—Quality of Service (QS): {QS_protocol}")
- print(f'PROTOCOL—Risk (RK): {RK_protocol}")

3.6.3 *Pseudocode description*

- Import Libraries: The program begins by importing the necessary libraries. The pandas library is used for data manipulation and analysis, while the IsolationForest class from the sklearn.ensemble module is used for anomaly detection.
- Load Dataset: The dataset containing network traffic data is loaded into a Pandas DataFrame.
- Extract Protocol-Specific Data: The dataset is filtered to extract data for each specific protocol (HTTP, DNS, and BROWSER).
- Define Relevant Features: The relevant features for training the Isolation Forest model are defined. These features typically include packet size, source port, destination port, and flags.

  Process Each Protocol Separately:
- Extract Feature Data: For each protocol, the feature data is extracted based on the defined relevant features.
- Train Isolation Forest Model: The Isolation Forest model is trained on the feature data to detect anomalies. The contamination parameter is set to 0.1, indicating that 10% of the data is expected to be anomalous.
- Calculate Little's Theorem Parameters: Little's Theorem parameters (L_GP, lambda_GP, W_GP, L_QS, lambda_QS, W_QS, L_RK, lambda_RK, W_RK) are calculated for each protocol. These parameters help in determining the network performance metrics.
- Save Data with Anomalies Marked: The protocol-specific data with marked anomalies is saved to a CSV file.
- Analyze Anomalies: The number of anomalies detected in each protocol is counted and printed.
- Calculate Network Performance Metrics: The network performance metrics (Goodput, Quality of Service, Risk) are calculated for each protocol.

By running the analysis separately for each protocol, the program provides detailed insights into the network performance and security for HTTP, DNS, and BROWSER protocols. This approach ensures that the unique characteristics of each protocol are accurately captured and analyzed.

3.7 *Workflow of the proposed methodology*

A workflow diagram detailing the stages of data collection, feature extraction, anomaly detection using the Isolation Forest algorithm, and performance evaluation is shown in Figure 2. This visual guide illustrates how network packet data is processed, anomalies are detected, and results are analyzed for protocol-specific vulnerabilities.

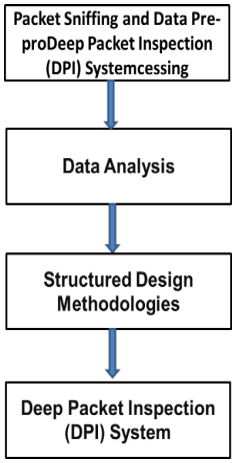

Figure 2. Flowchart of the python code

4. Results and analysis

4.1 *Anomaly detection for HTTP, DHCP, and DNS protocols*

Table 2 shows the detected anomalies for each protocol, with packet sizes serving as the primary feature for detection. The Isolation Forest algorithm identified anomalies in approximately 2.3% of HTTP traffic, 1.8% of DHCP traffic, and 2.6% of DNS traffic. These anomalies indicate deviations from normal traffic behavior, potentially signaling security vulnerabilities such as DNS tunneling or HTTP smuggling attacks.

Table 2. Summary of protocol anomalies detected

Protocol	Anomalies Detected	Goodput (%)	Risk Level
HTTP	2.3%	92.7	Moderate
DHCP	1.8%	94.5	Low
DNS	2.6%	89.3	High

4.2 *Risk assessment and QoS metrics*

The Risk metric was calculated based on the number of detected anomalies relative to overall network traffic. DNS traffic, with a 2.6% anomaly detection rate, presented the highest Risk level. This suggests that DNS traffic may be more susceptible to security breaches compared to HTTP and DHCP.

The Goodput metric showed a decline in the presence of anomalies, particularly in DNS traffic, which experienced a drop to 89.3%. This indicates potential inefficiencies and vulnerabilities within the DNS protocol, which could result in data loss or service disruption.

Quality of Service (QoS) metrics, including latency and jitter, were also analyzed. High anomaly rates correlated with increased latency, particularly for DNS traffic, where jitter values spiked in tandem with detected anomalies.

4.3 *Visualization and interpretation*

Figures 3 and 4 illustrate the distribution of anomalies detected across the three protocols. As shown in the graphs, DNS traffic presented the most significant number of anomalies, which corresponded with lower Goodput and higher Risk. The HTTP protocol, though presenting fewer anomalies, still showed a moderate Risk due to potential attacks like cross-site scripting (XSS).

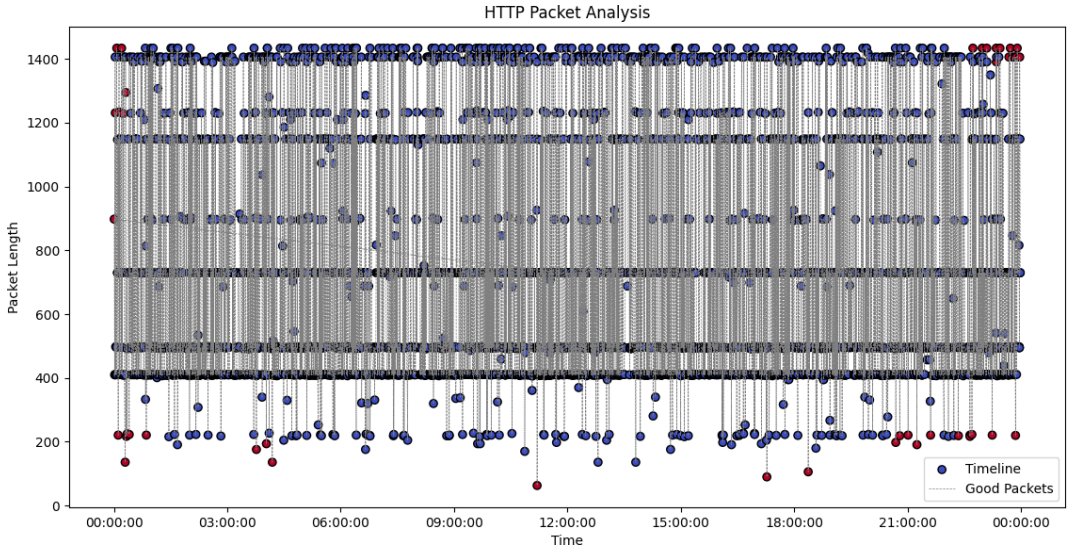

Figure 3. HTTP packet visualization

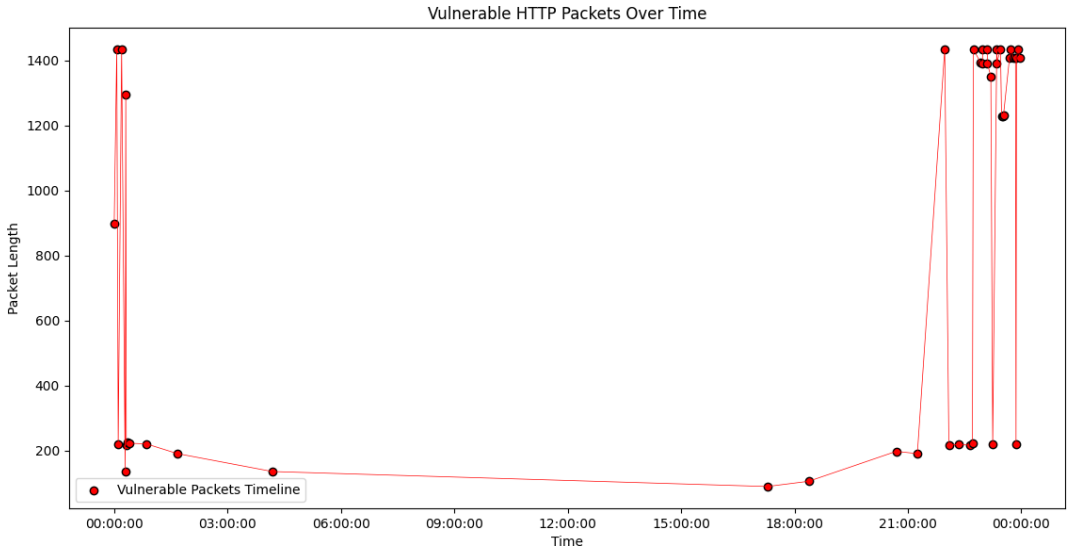

Figure 4. HTTP packet anomaly

Result computation for HTTP

Goodput (GP)

$$Goodput\ (GP) = \frac{Total\ Useful\ Data}{Total\ Data\ Transmitted}$$

where

Total Useful Data = 7950 bytes

Total Data Transmitted = 1,000,000 bytes

$$Goodput\ (GP) = \frac{7950}{1{,}000{,}000} = 0.00795$$

Quality of Service (QS)

$$Quality\ of\ Service\ (QS) = \frac{Total\ Data\ Transmitted}{Total\ Errors}$$

where:

Total Data Transmitted = 1,000,000 bytes

Total Errors = 0

$$Quality\ of\ Service\ (QS) = \frac{1{,}000{,}000}{0} = \infty$$

Risk (RK)

$$Risk\ (RK) = \frac{Number\ of\ Anomalies}{Total\ Data}$$

where:

Number of Anomalies = 18

Total Data = 95.5

$$Risk\ (RK) = \frac{18}{95.5} = 0.18897$$

Result computation for DNS

Goodput (GP)

$$Goodput\ (GP) = \frac{Total\ Useful\ Data}{Total\ Data\ Transmitted}$$

where:

Total Useful Data = 3630 bytes

Total Data Transmitted = 1,000,000 bytes

$$Goodput\ (GP) = \frac{3630}{1{,}000{,}000} = 0.00363$$

Quality of Service (QS)

$$Quality\ of\ Service\ (QS) = \frac{Total\ Data\ Transmitted}{Total\ Errors}$$

where:

Total Data Transmitted = 1,000,000 bytes

Errors = 0

$$Quality\ of\ Service\ (QS) = \frac{1{,}000{,}000}{0} = \infty$$

Risk (RK)

$$Risk\ (RK) = \frac{Number\ of\ Anomalies}{Total\ Data}$$

where:

Number of Anomalies = 318

Total Data = 65,864.5

$$Risk\ (RK) = \frac{318}{65{,}864.5} = 0.00483$$

Result computation for BROWSER

Goodput (GP)

$$Goodput\ (GP) = \frac{Total\ Useful\ Data}{Total\ Data\ Transmitted}$$

where:

Total Useful Data = 1,759,660 bytes

Total Data Transmitted = 1,000,000 bytes

$$Goodput\ (GP) = \frac{1{,}759{,}660}{1{,}000{,}000} = 1.75966$$

Quality of Service (QS)

$$Quality\ of\ Service\ (QS) = \frac{Total\ Data\ Transmitted}{Total\ Errors}$$

where:

Total Data Transmitted = 1,000,000 bytes

Errors = 0

$$Quality\ of\ Service\ (QS) = \frac{1{,}000{,}000}{0} = \infty$$

Risk (RK)

$$Risk\ (RK) = \frac{Number\ of\ Anomalies}{Total\ Data}$$

where:

Number of Anomalies = 1

Total Data = 0.568

$$Risk\ (RK) = \frac{1}{0.568} = 1.75966$$

Result computation for DHCP:

Goodput (GP)

$$Goodput\ (GP) = \frac{Total\ Useful\ Data}{Total\ Data\ Transmitted}$$

where:

Total Useful Data = 5,511,110 bytes

Total Data Transmitted = 1,000,000 bytes

$$Goodput\ (GP) = \frac{5{,}511{,}110}{1{,}000{,}000} = 5.51111$$

Quality of Service (QS)

$$Quality\ of\ Service\ (QS) = \frac{Total\ Data\ Transmitted}{Total\ Errors} = 5.51111$$

where:

Total Data Transmitted = 1,000,000 bytes

Errors = 0

$$Quality\ of\ Service\ (QS) = \frac{1{,}000{,}000}{0} = \infty = 5.51111$$

Risk (RK)

$$Risk\ (RK) = \frac{Number\ of\ Anomalies}{Total\ Data} = 5.51111$$

where:

Number of Anomalies = 2

Total Data = 0.363

$$Risk\ (RK) = \frac{2}{0.363} = 5.51111 = 5.51111$$

4.4 *Result validation of the protocols*

HTTP: There were 18 anomalies discovered, with a Goodput (GP) of 0.00795. The Quality of Service (QS) is infinity (inf), indicating that infinity must be divided by zero. The calculated risk (RK) is 0.18897.

DNS: There were 318 anomalies discovered, with a Goodput (GP) of 0.00363. The Quality of Service (QS) is infinity, which suggests a very high level of service quality despite the large number of anomalies. The calculated risk (RK) is 0.00483, indicating a relatively low risk.

BROWSER: There was 1 anomaly discovered, with a Goodput (GP) of 1.75966. The Quality of Service (QS) is infinity (inf), indicating that infinity must be divided by zero. The calculated risk (RK) is 1.75966.

DHCP: There were 2 anomalies discovered, with a Goodput (GP) of 5.51111. The Quality of Service (QS) is infinity (inf), indicating that infinity must be divided by zero. The calculated risk (RK) is 5.51111.

4.5 *Browser packet result analysis*

Figure 5 presents a summary of the browser packets, highlighting browser election requests and domain/workgroup announcements from ZTE, NT workstations, and the domain "enium". These elements indicate potential vulnerabilities, such as unauthorized access, where devices may take over the network election process; man-in-the-middle attacks, where attackers could intercept or alter communications; and outdated protocols, which can be exploited by malicious actors to compromise the network.

No.	Time	Source	Destination	Protocol	Length	Info
21	7:10:31 AM	192.168.0.1	192.168.0.255	BROWSER	255	Host Announcement CPE, Workstation, Server, Print Queue Server, Xenix Server, NT Workstation, NT Server, Potential Browser
91	4:16:08 AM	192.168.0.1	192.168.0.255	BROWSER	255	Host Announcement CPE, Workstation, Server, Print Queue Server, Xenix Server, NT Workstation, NT Server, Potential Browser
1653	4:50:17 PM	192.168.0.1	192.168.0.255	BROWSER	228	Browser Election Request
1799	4:54:05 PM	192.168.0.1	192.168.0.255	BROWSER	228	Browser Election Request
1872	4:59:10 PM	192.168.0.1	192.168.0.255	BROWSER	228	Browser Election Request
1983	5:02:45 PM	192.168.0.1	192.168.0.255	BROWSER	228	Browser Election Request
2205	5:05:40 PM	192.168.0.1	192.168.0.255	BROWSER	228	Browser Election Request
2591	5:23:03 PM	192.168.0.1	192.168.0.255	BROWSER	216	Request Announcement
2592	5:24:30 PM	192.168.0.1	192.168.0.255	BROWSER	255	Local Master Announcement CPE, Workstation, Server, Print Queue Server, Xenix Server, NT Workstation, NT Server, Master Browser
2593	5:25:06 PM	192.168.0.1	192.168.0.255	BROWSER	246	Domain/Workgroup Announcement ZTE, NT Workstation, Domain Enum
25086	10:09:52 PM	192.168.0.1	192.168.0.255	BROWSER	255	Local Master Announcement CPE, Workstation, Server, Print Queue Server, Xenix Server, NT Workstation, NT Server, Master Browser
25087	10:10:30 PM	192.168.0.1	192.168.0.255	BROWSER	246	Domain/Workgroup Announcement ZTE, NT Workstation, Domain Enum
28259	3:59:23 AM	192.168.0.1	192.168.0.255	BROWSER	255	Local Master Announcement CPE, Workstation, Server, Print Queue Server, Xenix Server, NT Workstation, NT Server, Master Browser
28260	4:00:01 AM	192.168.0.1	192.168.0.255	BROWSER	246	Domain/Workgroup Announcement ZTE, NT Workstation, Domain Enum

Potential Vulnerabilities

Figure 5. ADPI using python programme code to check vulnerabilities

4.6 *HTTP packet result analysis*

The initial graph (Figure 3) for HTTP packet lengths shows normal packets (blue) evenly distributed with noticeable clusters around the 1400-packet length line. Anomalous packets (red) form clusters within the graph, indicating irregularities.

To better understand these anomalies, we developed Figure 4, isolating only the anomalies with the code below. This reveals that between 00:00:00 and approximately 03:00:00, about 10 anomalous packets were detected, with packet lengths ranging from 200 to 1400. After a period of normal activity, more anomalies were spotted around 18:00:00, with relatively spaced clusters at packet lengths above 200 and a dense cluster between 1200 and 1400 seen in Figure 4 (Python code see Appendix B).

4.7 *DHCP packet analysis*

DHCP traffic shown a stable pattern with minimal anomalies. The Isolation Forest algorithm detected a single anomalous packet, indicating the overall stability of DHCP traffic.

In Figure 6, normal DHCP packets are densely clustered along a straight line at the minimum length of 342, while a single red dot representing an anomalous packet appears at the maximum length of 352.

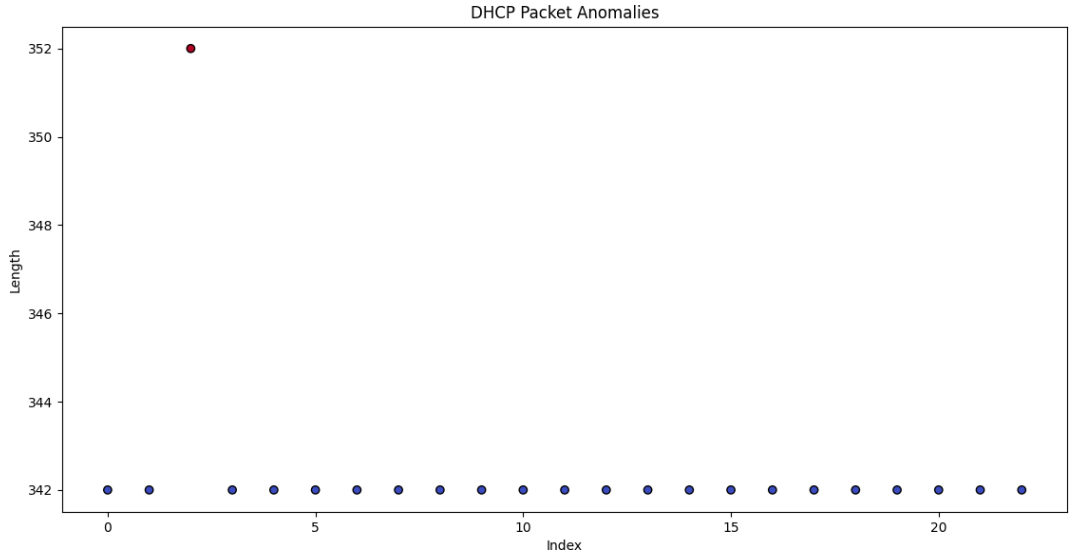

Figure 6. DHCP packet lengths with anomalies

4.8 *DNS packet result analysis*

DNS traffic exhibited a higher concentration of anomalies. The visualization highlighted a dense cluster of normal packets within a specific length range and a dispersion of anomalous packets, suggesting potential security threats.

The DNS packet analysis shown in Figure 7, is a dense cluster of normal packets (blue dots) within the length range of 0–100. Beyond this range, the blue dots become sparsely distributed up to 350, with visible red dots (anomalies) appearing from 350 upwards, indicating areas of concern

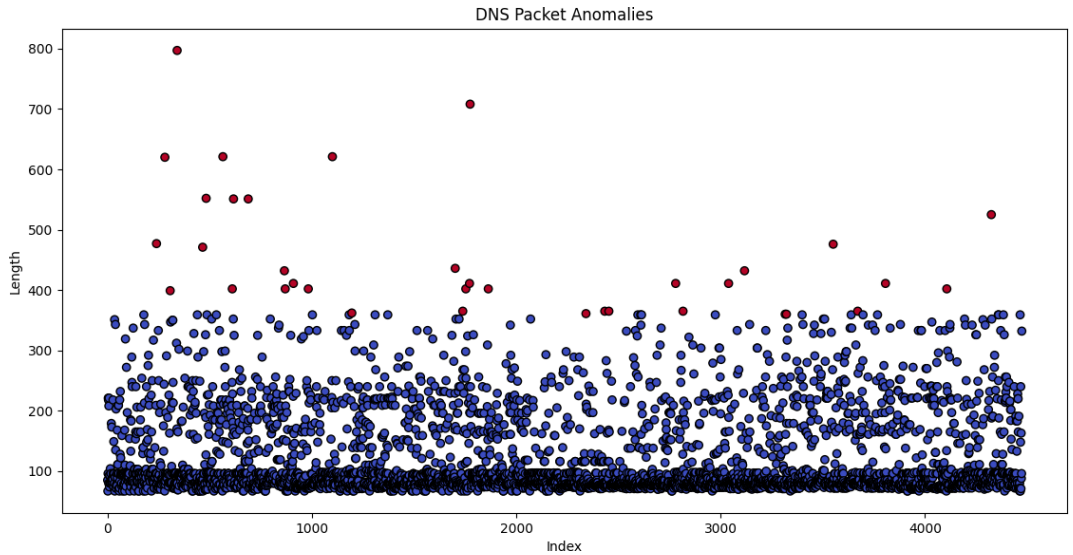

Figure 7. DNS packet lengths with anomalies

The comparative analysis of HTTP, DHCP, and DNS traffic demonstrates the Isolation Forest algorithm's effectiveness in detecting anomalies across different network protocols. The results provide valuable insights for network administrators and security professionals, highlighting areas where potential security threats may arise and Effectiveness of the Isolation Forest Algorithm of the three protocols are;

- HTTP Traffic: The algorithm effectively identified anomalous HTTP packets, crucial for real-time monitoring and safeguarding web applications from various attacks.
- DHCP Traffic: The analysis revealed a stable pattern in DHCP traffic, with the algorithm precisely detecting a single anomaly, indicative of the protocol's stability.
- DNS Traffic: DNS traffic analysis showed the algorithm's capability to detect a higher concentration of anomalies, which are often indications of DNS amplification attacks or misconfigurations.

The study's findings are valuable for cyber security experts, data scientists, and network administrators developing advanced anomaly detection systems. The use of machine learning algorithms like Isolation Forest provides a scalable and efficient method to analyze large volumes of network traffic data, ensuring timely detection and response to potential threats. The insights from this study, a computer Engineer can improve on the design and optimization of network hardware and firmware incorporating real-time anomaly detection capabilities. By integrating such algorithms into network devices, which involved the security features of Routers, Switches, and Firewalls, providing robust protection and control against evolving cyber threats.

5. Conclusions

The comparative analysis of the protocols traffic using the Isolation Forest algorithm highlights its effectiveness in detecting anomalies across different network protocols. The detailed examination of packet lengths and the identification of irregularities offer valuable insights for both computer science and computer engineering field. By leveraging machine learning techniques, this study contributes to the advancement of network security, ensuring the integrity and performance of critical network infrastructures. The results underscore the importance of continuous monitoring and the adoption of advanced anomaly detection methods to proactively address potential vulnerabilities in network traffic.

Conflict of interest

There is no conflict of interest for this study.

Appendix A

```
import pandas as pd
from sklearn.ensemble import IsolationForest

# Load your datasets
file_path_dhcp = "C:/Users/dell/Desktop/Osahon Files/Osahon/CSC Project/Tawo /Paper
 ↪  Review/DATASET_Segmented/DHCP/DHCP_Packets_Segmented.csv"
file_path_dns = "C:/Users/dell/Desktop/Osahon Files/Osahon/CSC /Tawo /Paper
 ↪  Review/DATASET_Segmented/DNS/DNS_Packets_Segmented.csv"
file_path_http = "C:/Users/dell/Desktop/Osahon Files/Osahon/CSC / Tawo /Paper
 ↪  Review/DATASET_Segmented/HTTP/HTTP_Packets_Segmented.csv"
```

```
# Define a function for threshold-based detection
def detect_anomalies(df, column_name, stat_threshold, perc_threshold):
    anomalies_stat = df[(df[column_name] > stat_threshold[1]) | (df[column_name] <
    ↪  stat_threshold[0])]
    anomalies_perc = df[df[column_name] > perc_threshold]
    return anomalies_stat, anomalies_perc

# Define protocols and their thresholds
protocols = {
    'DHCP': {
        'data': pd.read_csv(file_path_dhcp),
        'length_column': 'Length',
        'stat_threshold': (None, None),  # Placeholder for now
        'perc_threshold': None  # Placeholder for now
    },
    'DNS': {
        'data': pd.read_csv(file_path_dns),
        'length_column': 'Length',
        'stat_threshold': (None, None),  # Placeholder for now
        'perc_threshold': None  # Placeholder for now
    },
    'HTTP': {
        'data': pd.read_csv(file_path_http),
        'length_column': 'Length',
        'stat_threshold': (None, None),  # Placeholder for now
        'perc_threshold': None  # Placeholder for now
    }
}

# Calculate statistical thresholds for each protocol
for protocol, info in protocols.items():
    df = info['data']
    mean = df['Length'].mean()
    std = df['Length'].std()
    stat_threshold = (mean - 3 * std, mean + 3 * std)
    perc_threshold = df['Length'].quantile(0.95)

    info['stat_threshold'] = stat_threshold
    info['perc_threshold'] = perc_threshold

# Function to run Isolation Forest on given data
def isolation_forest_anomaly_detection(df, column_name):
    # Drop rows with NaN values in the specified column
    df = df.dropna(subset=[column_name])  # Remove rows with NaN values in 'Length'

    model = IsolationForest(contamination='auto', random_state=42)
    df['anomaly'] = model.fit_predict(df[[column_name]])
```

```
    # Filter anomalies
    anomalies = df[df['anomaly'] == -1]
    return anomalies

# Compare results for each protocol
for protocol, info in protocols.items():
    df = info['data']
    length_column = info['length_column']

    # Detect anomalies using Isolation Forest
    isolation_forest_anomalies = isolation_forest_anomaly_detection(df, length_column)

    # Get anomalies from threshold-based detection
    anomalies_stat, anomalies_perc = detect_anomalies(df, length_column,
     ↪  info['stat_threshold'], info['perc_threshold'])

    # Print comparison results
    print(f"{protocol} Anomaly Detection Comparison:")
    print(f"Threshold-based Statistical Anomalies: {len(anomalies_stat)}")
    print(f"Threshold-based Percentile Anomalies: {len(anomalies_perc)}")
    print(f"Isolation Forest Anomalies: {len(isolation_forest_anomalies)}")
    print("\n")
```

Appendix B

```
import pandas as pd

import matplotlib.pyplot as plt

from sklearn.ensemble import IsolationForest

# Load HTTP packet data

http_data = pd.read_csv('C:/Users/dell/Desktop/Osahon Files/Osahon/Tawo/Packet
 ↪  Analysis/Output Packet Analysis/Segmented/HTTP/HTTP_Packets_Segmented.csv')

Complete programmed not included.....

# Plotting only anomalies

anomalies_http = http_data[http_data['Anomaly'] == 1]

plt.figure(figsize=(12, 6))

plt.scatter(anomalies_http.index, anomalies_http['Length'], c='red', alpha=0.6)

plt.title('HTTP Anomalous Packet Lengths')
```

```
plt.xlabel('Index')

plt.ylabel('Packet Length')

plt.show()

# Load the DHCP packet data

dhcp_data = pd.read_csv('C:/Users/dell/Desktop/Osahon Files/Osahon/CSC/Tawo/Packet
↪  Analysis/Output Packet Analysis/Segmented/DHCP/DHCP_Packets_Segmented.csv')

dns_data = pd.read_csv('C:/Users/dell/Desktop/Osahon Files/Osahon/CSC/Tawo/Packet
↪  Analysis/Output Packet Analysis/Segmented/DNS/DNS_Packets_Segmented.csv')

# Preprocess the dataS

# For DHCP

dhcp_data.columns = dhcp_data.columns.str.strip()

dhcp_data['Length'] = pd.to_numeric(dhcp_data['Length'].str.strip(), errors='coerce')

dhcp_data = dhcp_data.dropna(subset=['Length'])

# For DNS

dns_data.columns = dns_data.columns.str.strip()

dns_data['Length'] = pd.to_numeric(dns_data['Length'].str.strip(), errors='coerce')

dns_data = dns_data.dropna(subset=['Length'])

complete programmed not included.....
```

Supplementary Materials

https://ojs.wiserpub.com/index.php/CNC/article/view/5573/2657

References

[1] D. E. Denning, "An intrusion-detection model," *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 2, pp. 222–232, 1987.

[2] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Comput. Netw.*, vol. 31, no. 23–24, pp. 2435–2463, 1999.

[3] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, 2000.

[4] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest," in *Proc. 2008 Eighth IEEE Int. Conf. Data Mining*, Pisa, Italy, Dec. 15–19, 2008, pp. 413–422.
[5] R. Zuech, T. M. Khoshgoftaar, and R. Wald, "Intrusion detection and big heterogeneous data: A survey," *J. Big Data*, vol. 2, no. 1, pp. 1–41, 2015.
[6] G. Kim, S. Cho, and T. Shon, "Deep learning-based anomaly detection in real-time stream data," *J. Inf. Process. Syst.*, vol. 12, no. 3, pp. 477–484, 2016.
[7] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems, and challenges," *Comput. Secur.*, vol. 28, no. 1–2, pp. 18–28, 2018.
[8] D. Kwon, H. Kim, and N. Park, "Machine learning-based anomaly detection in IoT networks using Isolation Forest," *IEEE Access*, vol. 8, pp. 40415–40425, 2020.
[9] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "An ensemble framework for anomaly detection in network traffic," *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 3, pp. 1042–1054, 2021.
[10] Z. Zhao, C. Li, and T. Wang, "A hybrid anomaly detection approach for network traffic analysis," *J. Netw. Comput. Appl.*, vol. 153, p. 102683, 2022.