EAI Endorsed Transactions
on Industrial Networks and Intelligent Systems

Research Article **EAI.EU**

A novel approach for graph-based real-time anomaly detection from dynamic network data listened by Wireshark

Muhammed Onur Kaya[1], Mehmet Ozdem[2], Resul Das[1,*]

[1]Department of Software Engineering, Technology Faculty, Firat University, 23119 Elazig, Turkiye
[2]Türk Telekom, Ankara, Turkiye.

Abstract

This paper presents a new approach for real-time anomaly detection and visualization of dynamic network data using Wireshark, known as the most widely used network analysis tool. As the complexity and volume of network data increases, effective anomaly detection has become important to maintain network performance and improve security. The proposed method uses comprehensive network packet information from Wireshark for fast and accurate anomaly detection. The collected network packets enable easy detection and interpretation of anomalies in a dynamic manner. The presentation is enriched with graph-based representations. Thanks to this method, network administrators can understand the data more easily and make more sound decisions. The results of our study show significant improvements in the effectiveness of anomaly detection and practical applicability of visualization tools in real-time scenarios. The present study integrates the advances in the network approach and various visualization methods to provide the new ideas for network security and management for dynamic network management improvement.

Keywords: Cyber Attack, Information Security, Graph Visualization, Temporal Dynamic Networks, Wireshark.

Received on 19 10 2024; accepted on 19 12 2024; published on 07 01 2025

doi:10.4108/eetinis.v12i2.7616

1. Introduction

In the digital world, cyber attacks, which threaten all connected elements, are extensively present in every field [1]. With the rapid increase in the number of wireless communication and internet connections due to IoT-based networks, the amount of data in cyberspace has also grown significantly. Moreover, cybercriminals exploit new vulnerabilities, causing harm to individuals, businesses, and governments.

As technology advances, the importance of advanced cybersecurity measures, especially Heterogeneous Networks and Multi-Layered Security (MLM) strategies, is increasing in the evolving cyber threat environment [2]. Heterogeneous networks provide data transmission using different protocols and technologies, which creates various challenges in network defense. In such networks, detecting and closing vulnerabilities is more complex and challenging than in traditional networks. Likewise, multi-layered security strategies require the implementation of security policies and controls at multiple layers (e.g., data link layer, network layer, and application layer), which makes it more difficult for attackers to access the system. However, these strategies should also be constantly updated and improved against the sophisticated attack techniques of cybercriminals. As part of security measures, Real-Time Network Monitoring (RTNM) and Intelligence-Based Threat Detection (CTI) systems also play an important role [3]. These systems try to identify abnormal activities by constantly monitoring network traffic. For example, a sudden and unexpected increase in traffic on the network may be a sign of a possible attack. Such systems enable early detection of potential threats and intervention. In addition, post-incident forensics help develop cybersecurity strategies, thus providing information on how to prevent similar attacks in the future.

*Corresponding author. E-mail: rdas@firat.edu.tr

Cyber attacks are usually initiated by experienced hackers and are characterized by long-term behaviors characterized as hybrid Tactics, Techniques, and Procedures (TTPs) [4]. An example of such a cyber attack is Distributed Denial of Service (DDoS) attacks. DDoS attacks are easily initiated and highly destructive network attacks [5]. An attacker usually infiltrates vulnerable nodes on the Internet, turns them into a botnet, and uses these large-scale distributed hosts to launch access attacks on victim servers by generating large packets with spoofed IP addresses. This attack can quickly exhaust the resources of victim servers, making them unable to respond to legitimate requests, thus causing a denial of service [6].

1.1. Problem statement

Cyber attacks are very difficult to detect by traditional intrusion detection methods. Traditional detection methods have difficulty coping with the ever-evolving nature of threats. Therefore, it becomes imperative to develop innovative solutions in the field of cybersecurity. The rise of artificial intelligence further emphasizes this critical need by making complex threats more powerful [7]. This situation requires sustainably strengthening defense strategies and choosing more proactive approaches to prevent attacks in advance.

The main problems encountered in the analysis and visualization process of network data can be summarized as follows:

- In current research, the inadequacy of studies on visualization of cyber attacks is striking [8].
- Cyber security experts and researchers do not benefit sufficiently from different visualization techniques to better understand attacks and develop effective defense strategies [9, 10].
- Traditional methods are inadequate in analyzing and visualizing large and complex cyber attack graphs.
- Processing live data and creating visualizations from this data are critical in attack detection [11].
- Data from dynamic analysis tools such as Wireshark alone are not sufficient to create effective live attack detection systems.

1.2. Motivation

Dynamic network data forms the foundation of contemporary communication systems, resulting in a continuously growing and increasingly intricate environment. In this landscape, real-time anomaly detection has become essential for ensuring network performance and addressing security challenges. The rapid detection of anomalies allows network administrators to take timely action, minimizing service disruptions and optimizing resource utilization.

In our study, we utilize Wireshark, the most prevalent network analysis tool worldwide, to introduce a novel method for detecting anomalies in dynamic network data. The comprehensive data collection and analysis features of Wireshark support real-time monitoring and detection efforts, while its intuitive interface aids in simplifying the identification of anomalies. Our approach not only targets anomaly detection but also incorporates an innovative method for graphically representing the resulting data. This visualization improves the quick understanding of network statuses, thus facilitating faster decision-making.

The findings from our research significantly advance the fields of network security and management, providing enhancements to the efficiency of real-time anomaly detection. Additionally, this work aspires to pave the way for new strategies in managing dynamic networks, contributing valuable insights to both academic discourse and practical applications.

1.3. Main contributions

In this paper, a comprehensive database for detecting attack techniques, called CyberThreatDB, is created by combining data obtained from the Wireshark tool with information from AlienVault Open Threat Exchange (OTX) and VirusTotal, which are used to analyze different threats [12]. Additionally, network traffic data analysis is supported through the use of graph visualization techniques, allowing for the visualization of relationships between network assets. Graph techniques stand out as a powerful method in attack detection and network traffic analysis, making complex attack scenarios more comprehensible [13]. These processes are carried out more effectively with an attack detection application that enables the live analysis and monitoring of data, thereby allowing security experts to respond to incidents more quickly.

The main contributions of this paper are as follows:

- A comprehensive threat intelligence database, CyberThreatDB, is presented by aggregating threat intelligence from various sources.
- The proposed system models connections between attack sequences through attack graphs and highlights anomalous points.
- Interactions and communications on network traffic are visualized in detail, enabling security experts to better understand complex attack scenarios.

- With customizable filtering and search options for users, the system provides the ability to quickly identify and analyze specific threats.
- Static and dynamic data integration is provided to analyze both historical datasets and dynamically obtained live network traffic.

The proposed system plays a critical role in enhancing the ability to identify and develop attack techniques that an attacker might use, thereby increasing the level of automation in forensic investigations. These capabilities contribute to organizations' cyber defenses and provide insights into the security posture of their systems. The proposed system can improve the analyst's understanding of an investigated attack and offer actionable insights related to the attack. Additionally, it can address the "lack of published or accessible methods" for threat hunting based on high-level attack patterns.

1.4. Paper organization

The remainder of the paper is structured as shown in Figure 1. In Section 2, the literature is reviewed, and existing systems are examined. Section 3 focuses on data processing steps, graphing methods, and techniques used specifically in the cyber attack detection process. In Section 4, the experimental results on the effectiveness and performance of these techniques are discussed in detail. Finally, Section 5 presents the conclusions.

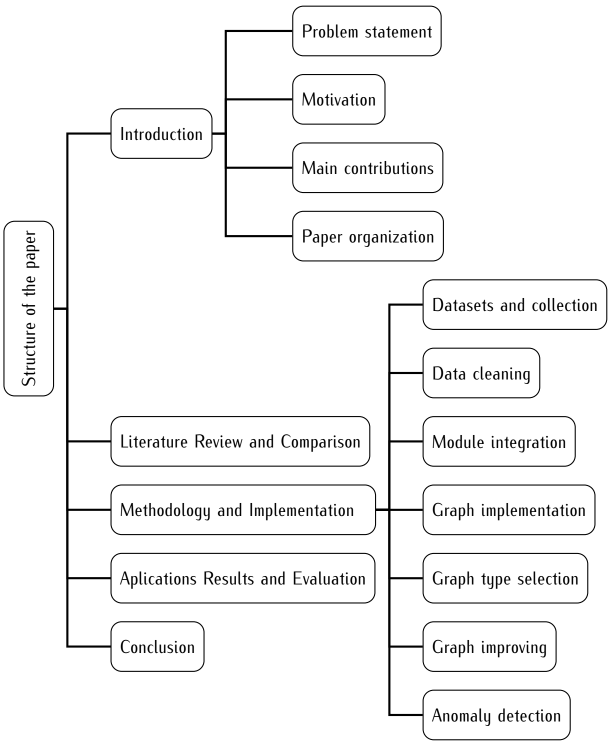

Figure 1. Organization of the paper

2. Literature Review and Comparison

Strengthening the resilience of information systems primarily relies on detecting and preventing cyber threats. Since the focus of this paper is network traffic, we have designed the detection of cyber threats to assist experts in this field. To analyze network traffic and detect attacks, it is essential to understand basic network concepts, such as the Transmission Control Protocol (TCP) three-way handshake used to initiate a connection between two devices. Network data capture and analysis tools present standard information that analysts are familiar with. At this point, visualization techniques should be applied to aid analysis and facilitate anomaly detection. In recent years, innovative approaches and ideas have emerged, particularly for detecting and predicting such threats. Fan [1] proposes a new approach to address the shortcomings of traditional cyberattack detection techniques. By linking system assets with the existing graph, a small and compact local graph is created to detect cyberattacks. However, more information is needed on how the method can scale in terms of memory and computational requirements for large-scale systems. Aryeh et al. [11] proposed an adaptable framework to address attack detection. The framework includes a GAN-based approach to generate multiple attacks and examples through the use of graphs, employing different attack strategies. Moreover, with the inclusion of unique components such as similarity transformation rate, more covert attacks can be executed and detected through this paper's methodology. However, it is noted that attack techniques focusing on specific attack strategies may be insufficient for generating broad and diverse attack examples. No data is provided on the performance of this research in real-world applications. Shen [4] conducted experiments on the DeepAG framework to simultaneously detect threats, predict future attacks, and construct the graphical structure of the attacks. This approach could take a step forward in effectively identifying and monitoring sophisticated threats. The study by Dodiya and Singh [18] examines methods for analyzing malicious traffic and collecting IoCs using Wireshark. These methods are particularly critical for malware detection. The work conducted by Qitao Zheng and colleagues [19] developed the SR-INT system, which combines software-defined networks with protocol-agnostic forwarding to enhance the efficiency of network monitoring processes.

The literature review highlights the importance of different methods developed for detecting cyber threats and analyzing network traffic. The studies

Table 1. Related works and comparison with our work.

Ref.	Tools	Purpose	Method	Analysis	Findings
[4]	HDFS, OpenStack, PageRank	Detection and prediction of Advanced Persistent Threats (APTs).	Machine learning techniques such as Bayesian Networks and Hidden Markov Models (GMM) were applied.	The limitations and challenges of traditional systems were identified.	The DeepAG framework was tested and proposed for simultaneous detection, prediction, and creation of attack graph structures.
[11]	UMaT network traffic dataset	Capture and analyze network packets using Python libraries and data science techniques.	Data was captured with Python's Scapy library and processed by converting it into a Pandas DataFrame.	Emphasis was placed on high-volume data transfers and suspicious IP addresses.	Python and data science were shown to accelerate and make network security analysis more effective.
[14]	NSL-KDD	Create a cyberattack detection model using deep learning and data visualization techniques in network logging systems.	Data filtering and transformation, pixel-based visualization, graphing, and coordinated multiple views were used.	A series of prototype visualizations were developed and an applied review was conducted to assess the effectiveness and limitations of the visualization approach.	Visualization approaches were tested on widely used datasets like Kyoto 2006+ and UGR'16 to evaluate their effectiveness and limitations.
[15]	Elasticsearch	Develop a cyberattack detection model using deep learning.	Advanced cyberattack detection models were developed using deep learning models like DNN, RNN, and LSTM.	The system set up with ELK Stack monitored network logs and cyberattacks, enabling rapid responses to security threats through in-depth analysis.	A system for managing and analyzing network log data was established. Network log data and attacks were made traceable through visualization.
[16]	GrafAttacker	Flexibly adjust attack strategies in graph analysis steps.	Generative adversarial networks (GAN)-based attack samples were produced using the GrafAttacker framework.	The features and responses of different graph analysis tasks were analyzed.	The effectiveness and general applicability of GrafAttacker were tested and evaluated across various graph analysis tasks.
[1]	Provenance database, Linux Auditd, Windows ETW	Develop an intelligent cyberattack detection system utilizing graph learning.	A heterogeneous graph was used to model system assets and events, and Graph Neural Networks (GNNs) were applied to detect cyberattacks.	The proposed graph learning method was compared with traditional rule-based and machine learning models on two real-world datasets.	The proposed model performed better than traditional detection methods and was effective at recognizing patterns, like advanced rule-based systems.
[7]	ExploitDB	Develop a dynamic cyber threat prediction model using publicly available data.	Cyber threat prediction was approached as a dynamic link prediction problem using heterogeneous graphs, employing the DHGL (Dynamic Heterogeneous Graph Learning) method.	The model was tested against other models on real-world datasets and demonstrated high effectiveness in learning evolutionary patterns of cyber threats.	The developed model outperformed traditional static and dynamic methods and effectively predicted cyber threats based on learned evolutionary patterns.
[9]	Scopus	Classify and visualize adversarial attacks and defense strategies.	A knowledge graph based on citation data was created using VOSviewer software.	A visualization and trend analysis of recent studies on adversarial attacks were performed.	Visualization studies revealed the potential for advancement in the field of adversarial attacks and identified new research directions.
[17]	UNSW-NB15, CIC-IDS2017	Develop a tool to extract and label packet capture files from modern network intrusion detection datasets.	With Payload-Byte, metadata information was used to label raw traffic captures and convert them into byte-level feature vectors.	Comparative analysis was performed using packet-based and flow-based data to train machine learning models.	Payload-Byte addressed the issues of comparability and reproducibility by providing a standardized basis for packet-based approaches.
This paper	Wireshark, OTX, VirusTotal	Develop a tool that includes a graph application and an attack detection module for network packets.	Graph applications were made using static and dynamic integration through the creation of the CyberThreatDB database and a Python module.	Cyberattack detection was achieved by visualizing live stream packets through dynamic integration.	A live network traffic visualization and attack detection application was developed to facilitate the work of cybersecurity experts, and various graph applications were demonstrated.

emphasize the roles of both traditional and innovative approaches in cybersecurity while also underscoring some limitations of existing methods. The proposed system focuses on overcoming these limitations and developing more effective detection and analysis methods.

3. Methodology and Implementation

This section aims to provide a detailed explanation of the methods used to analyze and visualize cyberattack data.

First, the challenges encountered during the data preprocessing phase and the methods used to overcome

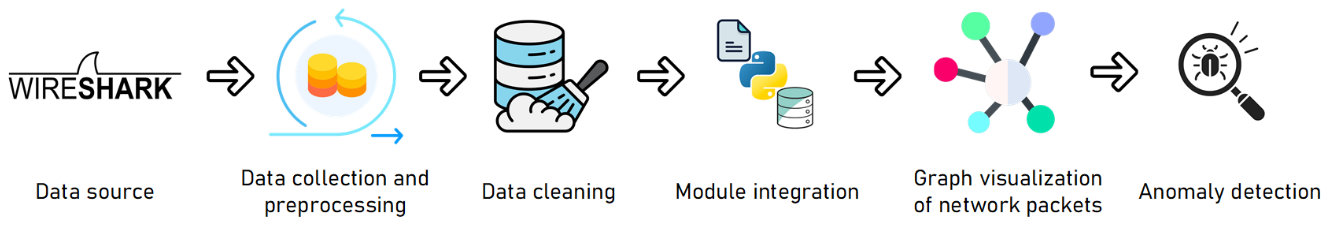
WIRESHARK
Data source
Data collection and preprocessing
Data cleaning
Module integration
Graph visualization of network packets
Anomaly detection

Figure 2. The general flow chart of the proposed approach.

these challenges will be explained. Various preprocessing steps will be utilized to understand the heterogeneous nature of cyberattack data and convert this data into a more consistent format. These steps will enable more robust data analysis and ensure that the data is suitable for the model.

3.1. Data collection

The primary objective of this study is to develop a tool for cybersecurity experts and system administrators to visualize and analyze network traffic for cyberattack detection and management. Initially, static data integration was achieved by capturing different network traffic over a specified time period using the Wireshark tool. A dataset was obtained, containing scenarios that included various network protocols, Internet Protocol (IP) addresses, port numbers, and timestamps. This dataset forms the basis by incorporating packets that represent both typical network traffic scenarios and anomalous conditions. Additionally, packets that constitute abnormal conditions, recorded through network scans conducted with the Wireshark network monitoring tool, were examined using the NetworkMiner software to assess their suitability for the dataset [20]. This dataset was further enhanced by merging data from AlienVault Open Threat Exchange and VirusTotal, which are used for the analysis of different threats, forming a comprehensive database known as CyberThreatDB.

Table 2. Sample segment of the network traffic dataset.

No	Time	Source IP	Destination IP	Protocol	Size
4	0.080058	192.168.1.105	192.168.1.109	ICMP	1042
7	1.031025	192.168.1.105	192.168.1.109	ICMP	1042
8	1.607525	192.168.1.100	192.168.1.255	UDP	121
17	4.094398	192.168.1.105	192.168.1.109	ICMP	1042
22	5.858237	23.99.80.186	192.168.1.109	TCP	54
26	7.035089	192.168.1.101	239.255.255.250	UDP	77
38	9.145422	192.168.1.109	192.168.1.105	ICMP	1042

PCAP is one of the most popular formats for capturing network traffic. In this study, two data formats were selected for evaluation and profiling: one is PCAP, and the other is CSV (Comma-Separated Values) format [21]. For dynamic data integration, the PCAP format was chosen using the automatic saving feature of the Wireshark tool. Through this method, packets are continuously recorded live. A sample segment of the recorded dataset is provided in Table 2. This format was later used in conjunction with modules like PyShark. Static integration, on the other hand, utilized the CSV format, another saving option in Wireshark, which was merged with CSV data from other sources to create CyberThreatDB.

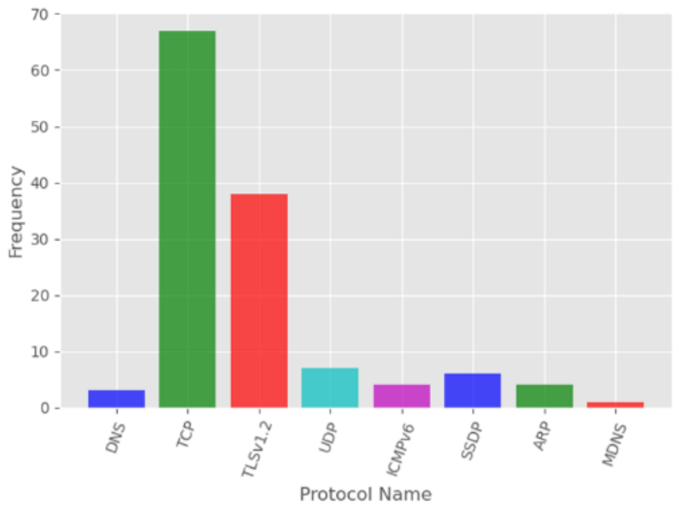

Figure 3. Bar chart illustrating the frequency of protocol usage in a specific section of the dataset.

In the generated dataset, attention has been paid to the use of various network protocols. In addition to fundamental protocols like Domain Name System (DNS), Transmission Control Protocol (TCP), and User Datagram Protocol (UDP), different other protocols

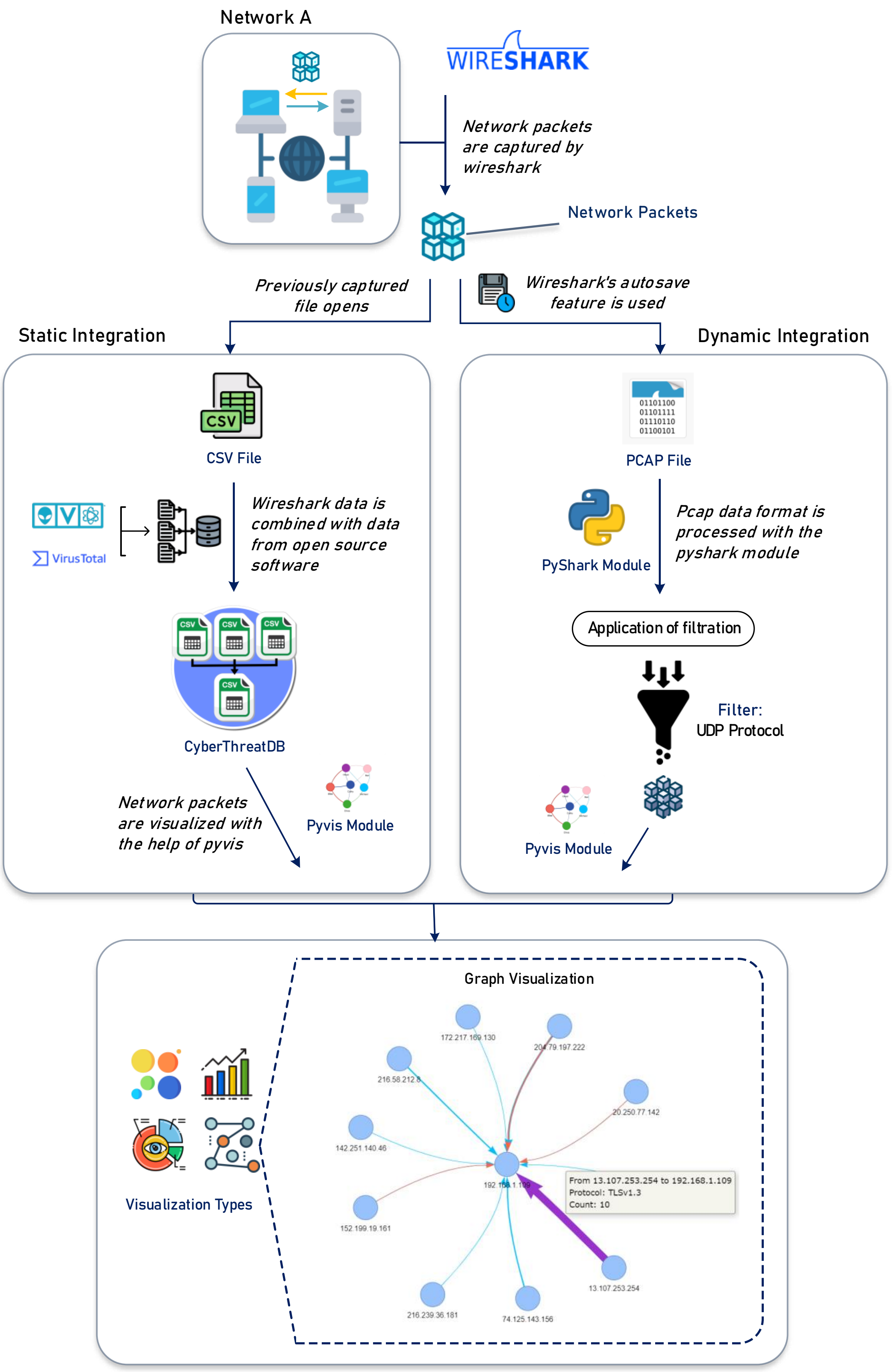

Figure 4. The detailed flow chart of the proposed approach.

Table 3. Comparison of static and dynamic data integration in network traffic analysis.

Features	Static Integration	Dynamic Integration
Description	Pre-recorded CSV files are integrated into the software.	A continuous analysis is performed by monitoring live network traffic in real-time with the help of PCAP files.
Purpose	Detailed analysis is conducted on pre-recorded data.	Detects real-time security threats and monitors network traffic in real-time.
Flexibility and Speed	Provides detailed analysis but has limited real-time intervention speed.	Offers the advantage of responding quickly to real-time changes.
Modules Used	Pandas, NetworkMiner, Pyvis	PyShark, Pyvis
Data Format	CSV	PCAP, CSV
Application Examples	Historical attacks, network performance analyses.	Real-time security breaches, real-time changes in network performance.

were used in the live stream with dynamic data integration to help the tool handle various abnormal scenarios. In Figure 3, a bar chart showing the frequency of protocol usage in a specific segment of the dataset is presented.

By observing the bar chart in Figure 3, the importance of protocol diversity can be understood. Protocol diversity allows the project to test over a broader range of network traffic and increases its resilience to potential security threats.

3.2. Data cleaning

The data cleaning phase was implemented for static data integration. Therefore, this phase was implemented for the data in the CyberThreatDB database in CSV format. The reason why this phase was not implemented for the data in dynamic integration is that the PyShark library, which will be used in the next phase, performs this phase automatically. In this process, the data set was read using Python's pandas library and unnecessary fields that would not be of any help in cyber attack detection and visualization were removed. Then, missing or erroneous data was detected and corrected or removed. This step was performed so that experts do not waste time with unnecessary data while using the application.

The data from AlienVault Open Threat Exchange and VirusTotal were in an irregular format. These irregularities were completely eliminated and the data was made processable. With this step, the data received from the Wireshark tool was made ready for python module integration. Before the data cleaning process, the following steps were made sure to be performed:

- *Data Standardization:* All data formats have been converted to a specified format.
- *Outlier Packet Detection:* Packets that deviate from the norm among Wireshark packets have been flagged.
- *Duplicate Data Prevention:* Duplicate packets were identified, and unnecessary repetitions were removed.
- *CRC (Cyclic Redundancy Check) Calculation:* Errors in packets were detected and corrected using error detection algorithms like CRC.
- *Noise Reduction:* Background noise irrelevant for analysis was filtered and reduced.
- *Data Density Control:* High-volume packets were compressed or deleted.
- *Incompatibility Detection:* Incompatibilities between nested protocols were identified and resolved.

The data cleaning phase ensures that unnecessary information is filtered out, and missing or erroneous data is corrected. This process prepares the dataset for efficient and accurate analysis in cybersecurity contexts. The cleaned version of each packet P_{clean} is expressed in Equation 1.

$$P_{\text{clean}} = P_{\text{orig}} \times \left(1 - \frac{F_{\text{noise}} + F_{\text{dup}} + F_{\text{err}}}{P_{\text{orig}}}\right) \quad (1)$$

Let P_{clean} represent the number of cleaned packets in the dataset, while P_{orig} denotes the original number of packets in the dataset. Additionally, F_{noise} refers to the packets identified as noise and subsequently removed, F_{dup} signifies the duplicate packets that have been identified and removed, and F_{err} indicates the erroneous packets that have been detected and either corrected or removed [22]. This formula encapsulates the core steps of the data cleaning phase, ensuring that the resulting dataset is cleaner, more accurate, and ready for further analysis. With the completion of these steps, the aim is to provide cybersecurity experts with cleaner and more processable data, accelerating their analysis processes and improving accuracy. This optimization can be considered within the scope of the concept of "data hygiene."

3.3. Module integration

This step is one of the fundamental stages for dynamic integration. As shown in Figure 4, the module integration phase effectively analyzes the dataset using a special module named PyShark, which is implemented in Python. Additionally, this stage involves adapting the data in both formats to be compatible with graph algorithms. Figure 5 illustrates the dynamic integration process of the PCAP file format, which is used for live network traffic data. PyShark is a Python library that captures network traffic and presents packets with various features [23]. PyShark integrates with tshark for efficient packet parsing, enabling flexible network traffic analysis through Python and tshark's Extensible Markup Language (XML) outputs.

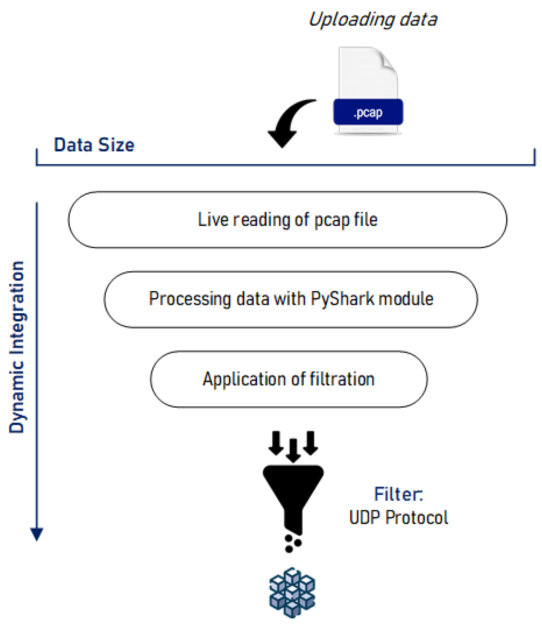
Uploading data
.pcap
Data Size
Live reading of pcap file
Processing data with PyShark module
Application of filtration
Dynamic Integration
Filter:
UDP Protocol

Figure 5. PCAP data format processing and dynamic integration process.

With its live capture and file capture features, PyShark can capture packets in real-time or from pre-recorded files, thus increasing its suitability for various scenarios [24]. The PyShark module has become a powerful tool that allows access to data in PCAP files created by the Wireshark tool in Python. By performing a detailed examination of the data it collects, the PyShark module analyzes factors such as packet headers, usage of communication protocols, data sizes, and communication durations. Filtering is used to handle network traffic more effectively.

The filtering process shown in 5 offers customizable options to the user, allowing them to focus on specific IP addresses or communication protocols. The filtering formula is shown in Equation 2. Following this filtering process, interesting data that meets the defined criteria emerges, which has been prepared through the integration of static and dynamic data. This output is then transformed into graph applications and visualized to enhance its suitability for attack detection [25].

$$P_{\text{filtered}} = P_{\text{total}} \times \left(\frac{C_{\text{IP}} + C_{\text{protocol}} + C_{\text{packets}}}{P_{\text{total}}} \right) \qquad (2)$$

Where:

– P_{filtered}: Number of packets that meet the filtering criteria.

– P_{total}: Total number of packets captured by PyShark.

– C_{IP}: Count of packets filtered based on specific IP addresses.

– C_{protocol}: Count of packets filtered based on specific communication protocols.

– C_{packets}: Total number of packets that meet both IP and protocol criteria.

After filtering, the refined dataset P_{filtered} is prepared for further analysis. The filtered data can be visualized using network graphs, where, as shown in Equation 3.

$$G(V, E) = \{V_{\text{IP}}, E_{\text{protocol}}, C_{\text{packets}}\} \qquad (3)$$

The graph $G(V, E)$ represents nodes, where V denotes the IP addresses, and edges E illustrate the relationships between these IP addresses. Within this context, V_{IP} is the set of IP addresses involved in the communication, while E_{protocol} consists of edges that represent the communication protocols or other features connecting these IP addresses [25, 26]. Additionally, C_{packets} refers to the total number of packets transmitted for each edge, indicating the strength of communication between the nodes. This formulation illustrates how the filtering process translates raw network traffic into meaningful graph representations, with nodes representing IP addresses and edges representing the relationships defined by protocols and packet counts. This aids in effective attack detection.

3.4. Graph implementation

The graph application includes the steps of converting the data set into a graph and applying various graph algorithms. In this step, it is important to properly create and process the graph data. For this purpose, graph visualization is preferred by using the Pyvis library, which is a versatile graph processing and visualization library, together with the Customtkinter interface creation library. The integration of Pyvis with CustomTkinter not only enhances the visual representation of the graph but also allows for an interactive user experience [11]. By enabling real-time manipulation of the graph, users can better understand the relationships within the data and efficiently analyze the results of various graph algorithms, leading to more informed decision-making and insights.

3.5. Graph type selection

The selection of the graph type was made based on the structure of the dataset and the analysis requirements. The most suitable graph type for an application where nodes represent IP addresses and edges represent protocols, packet counts, and other features is network graphs, which are used to visualize network data. These graphs have a structure where nodes represent entities in the network (IP addresses), and edges represent the relationships between these entities (protocols, packet counts, and other features). Among the subtypes of network graphs, the most appropriate for this kind of graph is "Node-Link" graphs, a specific type of network graph [27]. Additionally, by using "Circular Graphs" and "Force-Directed Graphs," the visualization is enriched, making attack detection easier.

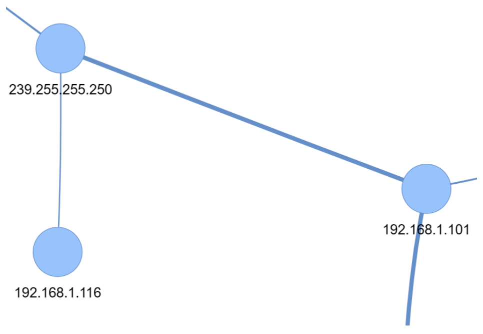

Figure 6. A type of graph where the edge weights are determined by the number of packets.

In the graph shown in Figure 6, the thickness of the edges is determined by the density of packet counts. For instance, since there is less packet flow between the IP addresses "192.168.1.116" and "239.255.255.250," a thinner edge is drawn. Conversely, a thicker edge is drawn between the IP addresses "239.255.255.250" and "192.168.1.101" due to a higher number of packets in that communication. This type of graph automatically determines the thickness of the edges while creating the nodes and edges based on packet counts.

To further refine this analysis, we consider a more comprehensive edge weighting formula, which incorporates not only the packet count but also other network characteristics such as latency and error rate [28]. The edge weight W_{ij} between node i and node j is calculated as shown in Equation 4.

$$W_{ij} = \alpha \cdot \frac{P_{ij}}{P_{\max}} + \beta \cdot \frac{L_{ij}}{L_{\max}} + \gamma \cdot \left(1 - \frac{E_{ij}}{E_{\max}}\right)^{\delta} \tag{4}$$

Where:

– W_{ij} represents the weight of the edge between nodes i and j, which dictates the thickness of the edge in the graph.

– P_{ij} is the number of packets exchanged between nodes i and j, normalized by the maximum packet count $P_{\max}$.

– L_{ij} denotes the latency between nodes i and j, normalized by the maximum latency $L_{\max}$.

– E_{ij} is the error rate for communication between nodes i and j, with $E_{\max}$ being the maximum observed error rate. The error rate is subtracted from 1 to ensure that higher error rates reduce the edge weight.

– α, β, and γ are weight coefficients that adjust the relative influence of packet count, latency, and error rate, respectively, on the edge weight.

– δ is a power factor to control the non-linearity of the error rate's influence on the final edge weight.

This formula ensures that the visualization not only reflects the density of packet exchange but also captures other key aspects of network performance, such as delays and reliability, enriching the analysis for more accurate detection of network anomalies or attacks.

3.6. Graph improving

Various techniques were employed to improve the graph. These techniques involved modifications to the graph to make it more meaningful or enhancements to the analysis results. For instance, techniques such as graph visualization or the identification of substructures were utilized. When hovering over any node, information such as the total packet count, node number, and IP address is displayed, while hovering over an edge reveals which node the packet originated from, making it interactive as shown in Figure 7.

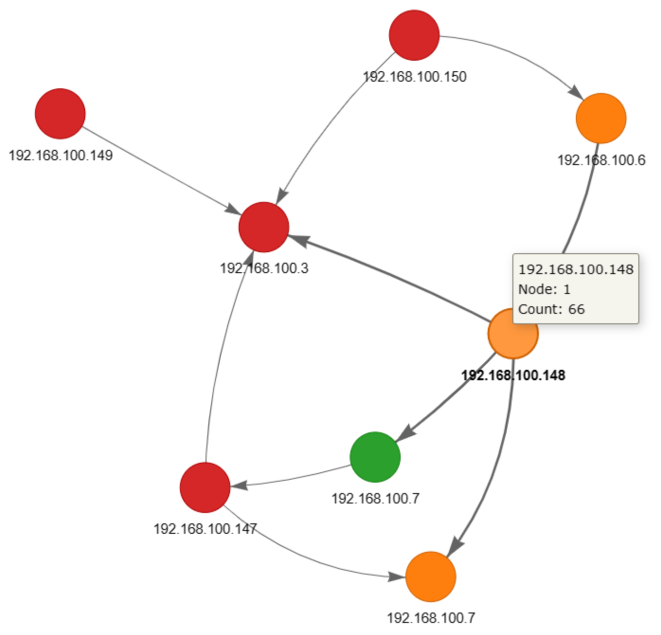

Figure 7. A graph output of network traffic representing a specific portion of the dataset.

To aid visualization, the graph nodes are shown in different colors, as seen in Figure 7. Nodes with the highest packet counts are displayed in red, those with the lowest are shown in green, and the remaining packets are represented in orange.

The coloring of nodes can be mathematically represented as shown in Equation 5.

$$C_i = \begin{cases} \text{Red} & \text{if } P_i = \max(P) \\ \text{Green} & \text{if } P_i = \min(P) \\ \text{Orange} & \text{otherwise} \end{cases} \tag{5}$$

Where:

– C_i: Color of node i.

– P_i: Packet count of node i.

– P: Set of packet counts of all nodes.

This formulation establishes a clear method for determining the color of each node based on its packet count. By implementing such visual cues, the graph becomes more intuitive and informative, facilitating better understanding and analysis of network traffic.

3.7. Anomaly detection

By utilizing various graph types and applications, experts can dynamically manage and analyze network traffic, providing a robust framework for anomaly detection [29]. These techniques leverage the power of visualization to easily identify patterns and outliers within complex datasets. In particular, these methods can be employed to detect common threats such as DDoS attacks, malware infections, phishing attempts, Man-In-The-Middle (MITM) attacks, and IP spoofing. Graphs assist in identifying such attacks by visualizing and analyzing unusual patterns in network traffic [30]. Moreover, the integration of real-time data into these graphical analyses allows for immediate responses to potential threats, enhancing the overall security framework. By continuously updating the visual representations, experts can adapt their strategies based on the most current network behaviors, ensuring a proactive stance against emerging risks.

Table 4. Anomaly detection in network traffic data.

Source IP	Destination IP	Protocol	Size	Status
192.168.1.105	192.168.1.109	ICMP	1042	**Abnormal**
192.168.1.105	192.168.1.109	ICMP	1042	**Abnormal**
192.168.1.100	192.168.1.255	UDP	121	Normal
192.168.1.105	192.168.1.109	ICMP	1042	**Abnormal**
23.99.80.186	192.168.1.109	TCP	54	Normal
192.168.1.101	239.255.255.250	UDP	77	Normal
192.168.1.109	192.168.1.105	ICMP	1042	Normal

The dataset used in Table 2 is processed into graphs. Experts can utilize these graphs to make detections as shown in Table 4. By analyzing specific attributes of the detected anomalies, such as source and destination IP addresses and the protocols used, experts can gain a deeper understanding of the nature of these threats and implement more effective countermeasures. To assess anomalies within network traffic, we calculate an anomaly score for each data packet based on its size, communication duration, and communication type, using the formula provided in Equation 6.

$$A_i = \frac{(P_i - \mu) \cdot w_P + (C_i - \mu_C) \cdot w_C + (T_i - \mu_T) \cdot w_T}{\sqrt{(w_P^2 \cdot \sigma_P^2) + (w_C^2 \cdot \sigma_C^2) + (w_T^2 \cdot \sigma_T^2)}} \tag{6}$$

In this context, the weight W_{ij} of the edge connecting nodes i and j determines the thickness of the edge in the graph, reflecting the relationship between these two nodes [31]. This weight is influenced by several factors, including the number of packets exchanged P_{ij}, which is normalized by the maximum packet count $P_{\max}$. Additionally, the latency L_{ij} between the nodes is

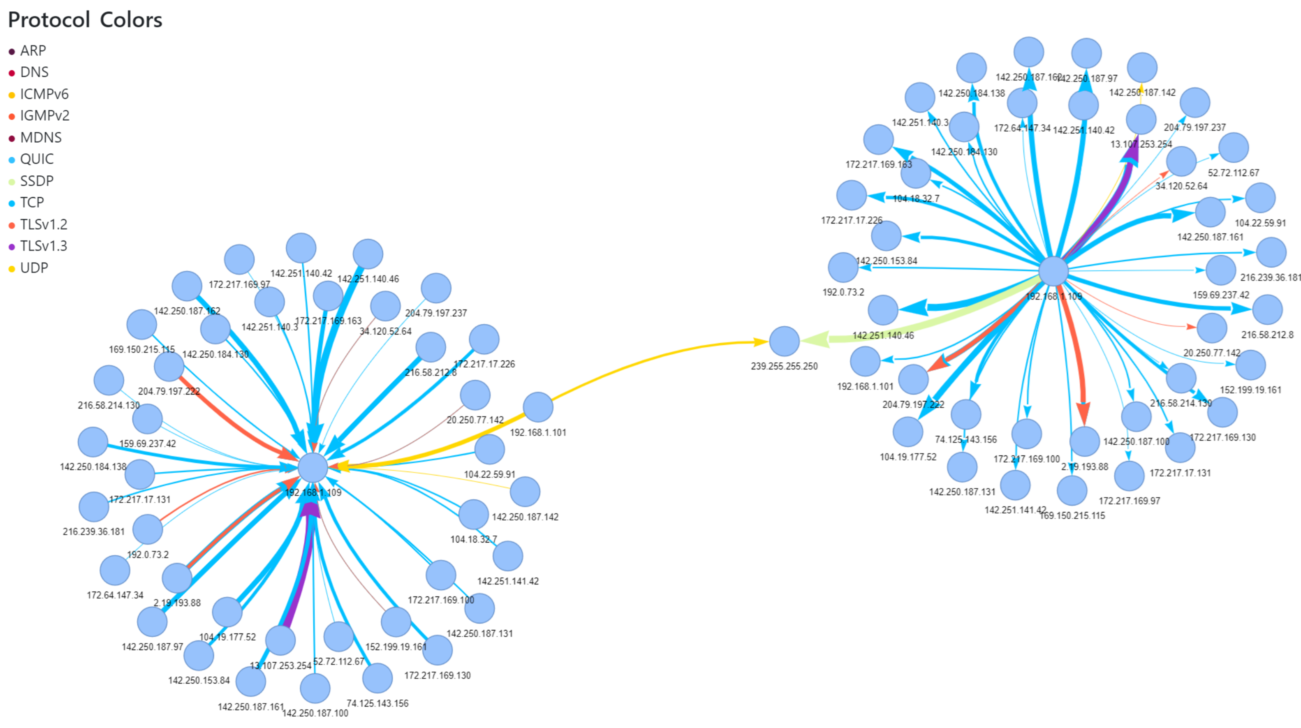

Figure 8. Graph representation of network traffic over a certain period of time.

also normalized by the maximum latency $L_{\max}$. Another critical factor is the error rate E_{ij} for the communication between the nodes, where $E_{\max}$ denotes the highest observed error rate; in this case, higher error rates lead to a reduced edge weight as the error rate is subtracted from 1. To fine-tune the influence of each factor on the edge weight, weight coefficients α, β, and γ are used to adjust the relative significance of packet count, latency, and error rate, respectively [28, 32]. Lastly, the parameter δ serves as a power factor that controls the non-linearity of the error rate's impact on the final edge weight, allowing for a more nuanced representation of the network dynamics in the graph. This approach not only facilitates real-time detection but also contributes to the continuous improvement of network security protocols.

4. Application Results and Evaluation

In this section, the results of the applications are discussed in depth. Various graph types used in the dataset have been tested, and the results have been evaluated. The effectiveness of the techniques employed for visualization and network traffic management has been illuminated.

For attack detection, it is essential to identify abnormal behaviors and patterns in the network. Typically, a specific device does not communicate with others, but suddenly starts sending a high volume of communication traffic, which can be noticeable. This could be an indicator of a potential attack. Another example is Denial of Service (DoS) attacks. In this type of attack, an attacker causes an excessive amount of traffic to be sent to the network, which disrupts normal communication. Using node-link graphs, normal traffic has been compared with network traffic containing anomalies. Various types of graph applications have been developed to identify this excessive traffic and inform experts of a possible attack. These scenarios have been tested both statically and dynamically on real network traffic data.

Figure 8 shows a network traffic graph indicating direction and intensity between different nodes. In the protocol map on the left, colors have been assigned to the protocols used in the network traffic, and the edges in the graph application are colored accordingly. In Figure 8, the weighted graph type is used to determine edge thickness based on the number of packets flowing in the traffic. Additionally, directed graphs are utilized to show the direction of packets moving from one node to another, represented by arrows on the edges. As a result, the colors and arrows used in the graph represent different types of traffic

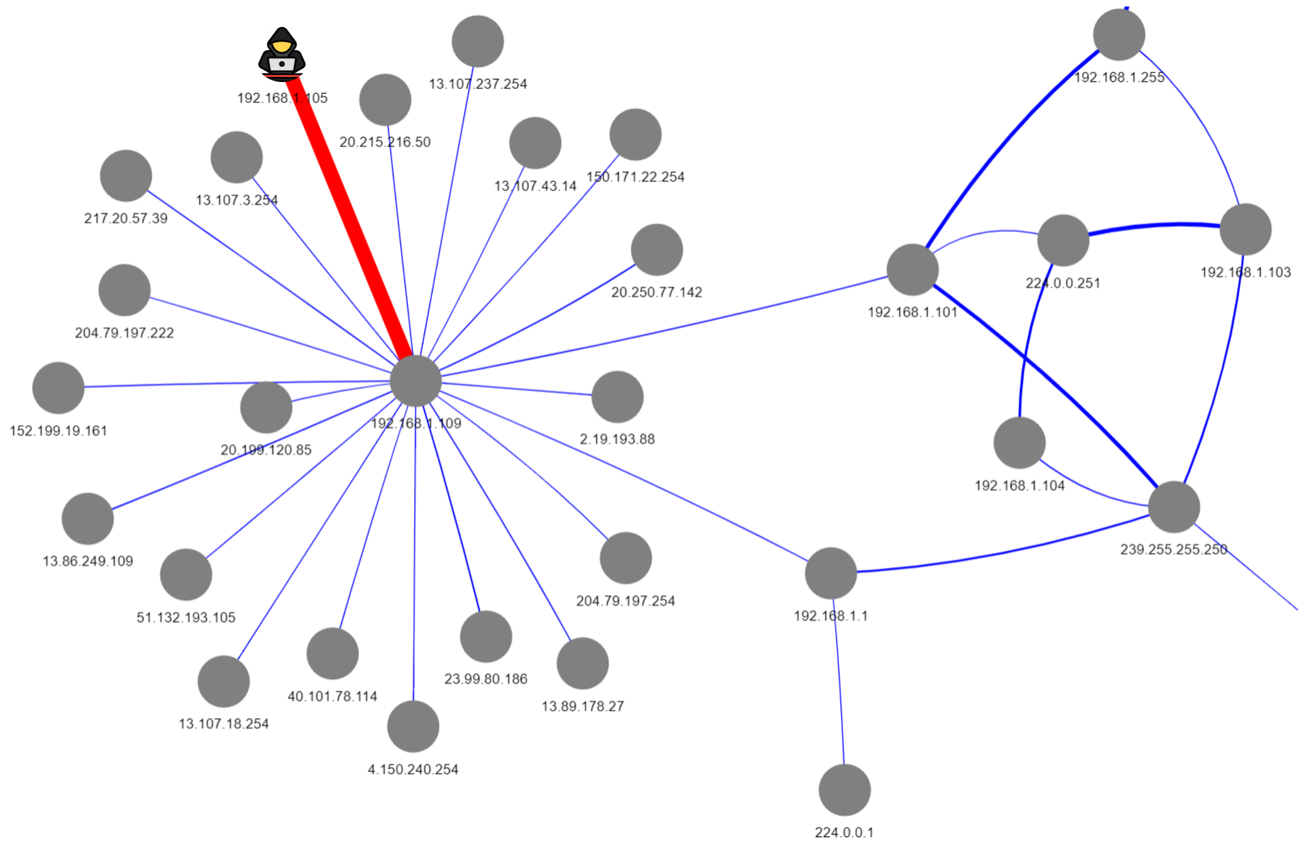

Figure 9. Graph implementation of network traffic subjected to DoS attack.

and their directions. The graph application clearly illustrates the direction and intensity of traffic, with different colors representing different traffic types. This information helps us understand which type of traffic is most prevalent in the network and which types may require specific optimizations. On the left side of Figure 8, incoming packets to the IP address "192.168.1.109" are displayed. By analyzing the density of incoming packets, we can gauge how actively the "192.168.1.109" device is being used and what types of services it is exposed to. It also helps us identify other devices communicating with the "192.168.1.109" device and the services they are utilizing. On the right side of Figure 8, outgoing packets from the IP address "192.168.1.109" are shown using a directed graph type.

As a result of the analysis, it can be observed in the graph application shown in Figure 8 that the number of packets coming in using the TLSv1.3 protocol is higher compared to those using other protocols. This indicates that the IP address "192.168.1.109" is communicating using a secure and modern protocol. It is also noted that a majority of the packets using the TLSv1.3 protocol are coming from a specific IP address. Furthermore, the graph on the right side of the figure shows that the packets exiting from the IP address "192.168.1.109" are also directed to the same IP address. This suggests that the IP address "192.168.1.109" frequently engages in secure communication with that specific address.

In the network traffic, it is observed that there is a high volume of UDP traffic directed to the IP address "192.168.1.255." UDP is a connectionless data transfer protocol, meaning it does not guarantee the sequential and error-free delivery of data packets from the sender to the receiver. UDP is commonly used in applications where latency is critical, such as VoIP, gaming, and video streaming. The density of traffic may be associated with the increase in UDP traffic directed to the IP address "192.168.1.255." This intensity often results from devices in the network sending data to this broadcast address. This situation could indicate an increased need for communication among devices in the network or signal an error condition. It is crucial for network administrators to monitor such scenarios using graph applications.

In the graph on the right side of Figure 8, the Internet Control Message Protocol version 6 (ICMPv6) traffic directed from the address "192.168.1.109" to various targets is often indicative of a scanning or discovery process taking place in the network, which is clearly evident in this analysis. ICMPv6 is a protocol used for error reporting, discovery operations, and network routing communication in IPv6 networks. Protocols based on ICMPv6, such as the Neighborhood Discovery Protocol, facilitate tasks like enabling devices to find each other and discovering network topology. When a device sends ICMPv6 packets to different targets from the same IP address, it is generally interpreted as part of the process of devices communicating with each other and discovering other devices in the network. Analyzing this type of traffic with the aid of graph applications is crucial for experts to ensure network security when necessary.

To detect attacks and abnormal behaviors in the network, it is essential to obtain evidence from visualizations indicating a potential attack. In this scenario, data collected by monitoring traffic on the same network has been visualized and compared using graphs, distinguishing between normal and abnormal behavior. Continuous requests targeting a specific IP address caused network disruptions. Specifically, continuous requests were sent from the IP address "192.168.1.105", leading to interruptions in the network. This situation is visually represented in Figure 9. In the visualization, the high volume of traffic coming from the address "192.168.1.105" has been marked as an indicator of a Denial of Service attack. Furthermore, the effects and details of this attack are presented in the accompanying document.

5. Conclusion

Graphs have played a significant role throughout history as a model for solving and analyzing many unresolved problems. Today, graphs have become so popular that they can be used to model problems or algorithms in almost every field. Graphs are encountered in nearly all domains, including computer science, linguistics, engineering, mathematics, and medicine. In particular, their suitability for network applications forms the foundation of this paper. The graph-based application developed to analyze, manage, and detect potential cyberattacks in network traffic holds critical importance. This application allows cybersecurity experts to monitor and control network operations by visualizing network traffic. This paper has introduced the pervasive threat of cyberattacks and highlighted the need for innovative solutions due to the limitations of traditional detection methods. The developed tool aids cybersecurity experts in proactively enhancing network security and responding to cyberattacks more quickly and effectively by enabling them to visualize and monitor network traffic. Utilizing the Pyvis library, various types of graphs were developed to visualize network traffic data. Importantly, through dynamic integration, not only static data but also real-time network traffic was incorporated into the graph application. This study opens the door to monitoring and visualizing network attacks on the Internet, the world's largest information network.

Abbreviations

TTPs	Tactics, Techniques, and Procedures
MLM	Multi-Layered Security
RTNM	Real-Time Network Monitoring
CTI	Intelligence-Based Threat Detection
IP	Internet Protocol
DDoS	Distributed Denial of Service
OTX	Open Threat Exchange
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
DNS	Domain Name System
MITM	Man-In-The-Middle
XML	Extensible Markup Language
ICMPv6	Internet Control Message Protocol version 6

References

[1] M. Lv, C. Dong, T. Chen, T. Zhu, Q. Song, and Y. Fan, "A heterogeneous graph learning model for cyber-attack detection," 2021.

[2] X. Li, G. Xu, W. Lian, H. Xian, L. Jiao, and Y. Huang, "Multi-layer network local community detection based on influence relation," *IEEE Access*, vol. 7, pp. 89 051–89 062, 2019, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/document/8733023?denied=

[3] M. T. Alam, D. Bhusal, Y. Park, and N. Rastogi, "Looking beyond iocs: automatically extracting attack patterns from external cti," Jul. 2023, arXiv:2211.01753 [cs]. [Online]. Available: http://arxiv.org/abs/2211.01753

[4] T. Li, Y. Jiang, C. Lin, M. S. Obaidat, Y. Shen, and J. Ma, "DeepAG: Attack graph construction and threats prediction with bi-directional deep learning," *IEEE Trans. Dependable and Secure Comput.*, vol. 20, no. 1, pp. 740–757, Jan. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9684707/

[5] M. Baykara and R. Daş, "SoftSwitch: a centralized honeypot-based security approach usingsoftware-defined switching for secure management of vlan networks," *Turk J Elec Eng & Comp Sci*, vol. 27, no. 5, pp. 3309–3325, Sep. 2019. [Online]. Available: https://journals.tubitak.gov.tr/elektrik/vol27/iss5/4

[6] M. Husak, J. Komarkova, E. Bou-Harb, and P. Celeda, "Survey of attack projection, prediction, and forecasting in cyber security," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 1, pp. 640–660, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8470942/

[7] J. Zhao, M. Shao, H. Wang, X. Yu, B. Li, and X. Liu, "Cyber threat prediction using dynamic heterogeneous graph learning," *Knowledge-Based Systems*, vol. 240, p. 108086, 2022. [Online]. Available: https://sciencedirect.com/science/article/pii/S0950705121011564

[8] H. Chen, G. Chen, E. Blasch, M. Kruger, and I. Sityar, "Analysis and visualization of large complex attack graphs for networks security," *Proc SPIE*, pp. 657 004–657 004, Apr. 2007.

[9] T. Long, Q. Gao, L. Xu, and Z. Zhou, "A survey on adversarial attacks in computer vision: Taxonomy, visualization and future directions," *Computers & Security*, vol. 121, p. 102847, Oct. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404822002413

[10] U. Banerjee, A. Vashishtha, and S. Mukul, "Evaluation of the capabilities of wireshark as a tool for intrusion detection," *International Journal of Computer Applications*, vol. 6, Sep. 2010.

[11] F. L. Aryeh, B. K. Alese, and O. Olasehinde, "Graphical analysis of captured network packets for detection of suspicious network nodes," in *2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*. Dublin, Ireland: IEEE, Jun. 2020, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/9139672/

[12] S. Paakala, "Integrating open threat exchange data in a security operations center."

[13] R. Das and M. Soylu, "A key review on graph data science: The power of graphs in scientific studies," *Chemometrics and Intelligent Laboratory Systems*, vol. 240, no. 104896, Jun. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169743923001466

[14] S.-Y. Ji, B.-K. Jeong, and D. H. Jeong, "Evaluating visualization approaches to detect abnormal activities in network traffic data," *Int. J. Inf. Secur.*, vol. 20, no. 3, pp. 331–345, Jun. 2021. [Online]. Available: https://doi.org/10.1007/s10207-020-00504-9

[15] J.-C. Liu, C.-T. Yang, Y.-W. Chan, E. Kristiani, and W.-J. Jiang, "Cyberattack detection model using deep learning in a network log system with data visualization," *J Supercomput*, vol. 77, no. 10, pp. 10 984–11 003, Oct. 2021. [Online]. Available: https://doi.org/10.1007/s11227-021-03715-6

[16] J. Chen, D. Zhang, Z. Ming, K. Huang, W. Jiang, and C. Cui, "GraphAttacker: A general multi-task graph attack framework," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 2, pp. 577–595, Mar. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9613779/

[17] Y. A. Farrukh, I. Khan, S. Wali, D. Bierbrauer, J. A. Pavlik, and N. D. Bastian, "Payload-byte: A tool for extracting and labeling packet capture files of modern network intrusion detection datasets," in *2022 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*, Dec. 2022, pp. 58–67. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10062281

[18] B. Dodiya and U. Singh, "Malicious traffic analysis using wireshark by collection of indicators of compromise," *International Journal of Computer Applications*, vol. 183, pp. 975–8887, Feb. 2022.

[19] Q. Zheng, S. Tang, B. Chen, and Z. Zhu, "Highly-efficient and adaptive network monitoring: When int meets segment routing," *IEEE Trans. Netw. Serv. Manage.*, vol. 18, no. 3, pp. 2587–2597, Sep. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9387410/

[20] I. Sembiring, Suharyadi, A. Iriani, J. V. B. Ginting, and J. A. Ginting, "A novel approach to network forensic analysis: Combining packet capture data and social network analysis," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 3, 2023, place: West Yorkshire, United Kingdom Publisher: Science and Information (SAI) Organization Limited. [Online]. Available: https://www.proquest.com/docview/2807223019/abstract/7CD772056BFE4A45PQ/1

[21] M. Z. Nicanor, "A comparison between text, parquet, and pcap formats for use in distributed network flow analysis on hadoop," *JACN*, pp. 59–64, 2017. [Online]. Available: http://www.jacn.net/index.php?m=content&c=index&a=show&catid=51&id=299

[22] P. Boniol and T. Palpanas, "Series2Graph: Graph-based subsequence anomaly detection for time series," *Proc. VLDB Endow.*, vol. 13, no. 12, pp. 1821–1834, Aug. 2020, arXiv:2207.12208 [cs]. [Online]. Available: http://arxiv.org/abs/2207.12208

[23] Y. Dong, R. Wang, and J. He, "Real-time network intrusion detection system based on deep learning," in *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*. Beijing, China: IEEE, Oct. 2019, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/document/9040718/

[24] T. S. Rajput, K. Maheshwar, and T. Jain, "Research paper: Voip packet analyzer for detecting threats in sip network: International journal of advanced research in computer science," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 9, pp. 613–618, Dec. 2017, publisher: International Journal of Advanced Research in Computer Science. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=obo&AN=128762614&lang=tr&site=ehost-live

[25] H. Kim, B. S. Lee, W.-Y. Shin, and S. Lim, "Graph anomaly detection with graph neural networks: Current status and challenges," *IEEE Access*, vol. 10, pp. 111 820–111 829, 2022, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/document/9906987/?arnumber=9906987

[26] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A comprehensive survey on graph anomaly detection with deep learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12 012–12 038, Dec. 2023, conference Name: IEEE Transactions on Knowledge and Data Engineering. [Online]. Available: https://ieeexplore.ieee.org/document/9565320/?arnumber=9565320

[27] J. Scott-Brown and B. Bach, "Netpanorama: A declarative grammar for network construction, transformation, and visualization," Oct. 2023, arXiv:2310.18902 [cs]. [Online]. Available: http://arxiv.org/abs/2310.18902

[28] Y. Wu, H.-N. Dai, and H. Tang, "Graph neural networks for anomaly detection in Industrial internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9214–9231, Jun. 2022, conference Name: IEEE Internet of Things Journal. [Online]. Available: https://ieeexplore.ieee.org/document/9471816/?arnumber=9471816

[29] M. Sülü and R. Daş, "Graph visualization of cyber threat intelligence data for analysis of cyber attacks," *Balkan Journal of Electrical and Computer Engineering*, vol. 10, no. 3, pp. 300–306, Jul. 2022. [Online]. Available: http://dergipark.org.tr/en/doi/10.17694/bajece.1090145

[30] M. Baykara and R. Das, "A novel honeypot based security approach for real-time intrusion detection and prevention systems," *Journal of Information Security and Applications*, vol. 41, pp. 103–116, Aug. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212616303295

[31] L. Cai, Z. Chen, C. Luo, J. Gui, J. Ni, D. Li, and H. Chen, "Structural temporal graph neural networks for anomaly detection in dynamic graphs," May 2020, arXiv:2005.07427 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2005.07427

[32] T. Pourhabibi, K.-L. Ong, B. H. Kam, and Y. L. Boo, "Fraud detection: A systematic literature review of graph-based anomaly detection approaches," *Decision Support Systems*, vol. 133, p. 113303, Jun. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923620300580