




Article

Node Classification of Network Threats Leveraging Graph-Based Characterizations Using Memgraph

Sadaf Charkhabi ¹, Peyman Samimi ¹, Sikha S. Bagui ^{1,*} , Dustin Mink ²  and Subhash C. Bagui ³ 

¹ Department of Computer Science, University of West Florida, Pensacola, FL 32514, USA; sadaf.charkhabi@gmail.com (S.C.); pe.samimi@gmail.com (P.S.)

² Department of Cybersecurity, University of West Florida, Pensacola, FL 32514, USA; dmink@uwf.edu

³ Department of Mathematics and Statistics, University of West Florida, Pensacola, FL 32514, USA; sbagui@uwf.edu

* Correspondence: bagui@uwf.edu

Abstract: This research leverages Memgraph, an open-source graph database, to analyze graph-based network data and apply Graph Neural Networks (GNNs) for a detailed classification of cyberattack tactics categorized by the MITRE ATT&CK framework. As part of graph characterization, the page rank, degree centrality, betweenness centrality, and Katz centrality are presented. Node classification is utilized to categorize network entities based on their role in the traffic. Graph-theoretic features such as in-degree, out-degree, PageRank, and Katz centrality were used in node classification to ensure that the model captures the structure of the graph. The study utilizes the UWF-ZeekDataFall22 dataset, a newly created dataset which consists of labeled network logs from the University of West Florida's Cyber Range. The uniqueness of this study is that it uses the power of combining graph-based characterization or analysis with machine learning to enhance the understanding and visualization of cyber threats, thereby improving the network security measures.

Keywords: graph machine learning; graph neural networks; graph database; Memgraph; node classification; MITRE ATT&CK framework; network threats; PageRank; Katz centrality; betweenness centrality



Citation: Charkhabi, S.; Samimi, P.; Bagui, S.S.; Mink, D.; Bagui, S.C. Node Classification of Network Threats Leveraging Graph-Based Characterizations Using Memgraph. *Computers* **2024**, *13*, 171. <https://doi.org/10.3390/computers13070171>

Academic Editor: Paolo Bellavista

Received: 2 July 2024

Revised: 11 July 2024

Accepted: 12 July 2024

Published: 15 July 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In today's digital world, businesses of all sizes are increasingly vulnerable to cyber threats. The advent of sophisticated cyberattacks necessitates innovative defenses, especially for organizations constrained by limited resources. Addressing these vulnerabilities is not just about detecting threats, but it also hinges on the accuracy and effectiveness of the response strategies deployed. The concept of node classification offers a powerful yet straightforward solution. Node classification, at its core, is about sorting different nodes in a network into groups based on their connections or characteristics. This approach is especially relevant for analyzing network traffic, as it assists in classifying the various elements within the network, offering insights that can promptly be acted upon.

The relevance of node classification within the context of our dataset, which records 700,340 cybersecurity events through the Zeek Network Security Monitor, cannot be overstated. By employing node classification, we can examine through the binary distinction of attack versus non-attack events, parsing through details such as attack tactics and methodologies. This granular approach enables us to not only identify which nodes represent a security threat but also distinguish the nature of threats such as reconnaissance, privilege escalation, defense evasion, etc.

Our decision to leverage node classification is based on its ability to organize intricate data into understandable trends and forecasts. In our dataset, UWF-ZeekDataFall22 [1], every node represents a network activity, encapsulated by details such as IP addresses, ports, and protocols. By classifying these nodes, we transform raw data into a structured form, allowing us to pinpoint the potential vulnerabilities and preemptively boost our

defenses against the most critical threats. This method is crucial for organizations without extensive cybersecurity resources, offering a cost-effective and practical way to protect their online perimeters.

In this paper, we present a novel application of the node classification within Memgraph to our cybersecurity dataset, UWF-ZeekDataFall22 [1], leveraging the various graph-based features. So, before we delve into node classification, the first part of the paper presents a detailed characterization of the graph-based features as well as graph visualizations. Features that were analyzed were as follows: page rank, degree centrality, betweenness centrality, and Katz centrality. And graph visualizations allow for the intuitive recognition of patterns, clusters, and outliers that might go unnoticed in raw tabular data.

Memgraph is an open-source graph analytics platform that allows for the representation of graphs and the application of Graph Machine Learning methods via its assorted integrated libraries [2]. Comparable in many aspects to Neo4j, Memgraph has distinct differences; for instance, while Neo4j is developed in Java and stores data on a disk, Memgraph is built with C/C++ and utilizes in-memory data storage, which enhances the performance but also means that the volume of data that can be loaded is tied to the machine's available RAM. Both systems include specialized Data Science libraries—GDS for Neo4j and MAGE for Memgraph [3,4]. In this study, we leverage the algorithms from MAGE to generate Graph Neural Networks (GNNs) using frameworks like Torch and the Deep Graph Library (DGL), which aids in classifying network attack tactics, as well as in crafting visualizations of network connections and querying the graph for additional information. This approach is not only instrumental in identifying at-risk network resources but also in demonstrating the powerful combination of graph-based techniques and machine learning in enhancing cybersecurity defenses.

The rest of this paper is organized as follows. Section 2 presents the related literature; Section 3 presents the dataset; Section 4 explains data preprocessing; Section 5 introduces Memgraph; Section 6 presents graph visualizations using Memgraph; Section 7 presents graph characterizations using Memgraph; Section 8 presents node classification; Section 9 presents the conclusions; and finally, Section 10 presents future works.

2. Related Literature

The ever-evolving domain of cybersecurity demands strong and inventive protective measures as organizations confront increasingly complex and sophisticated cyber threats. The efficacy of response strategies and the ability to accurately detect and classify network threats are critical areas of research in the field.

Significant contributions to cybersecurity discourse include the introduction of Bayesian Privilege Attack Graphs, which provide a mission-centric decision support framework [5]. This approach uses these graphs to model causal relationships and assess the resilience of system configurations, thus informing strategic decision making. A case study focusing on a medical information system illustrates the practical implications of their model, especially when user demands conflict with the most resilient configuration.

In another work, Jacob et al. [6] discuss the challenges of detecting cybersecurity attacks in software applications that employ a microservices architecture. By utilizing graph convolutional networks, they provide a graph-based anomaly detection system that captures the spatial and temporal dynamics within an application's microservice traffic. This enables the identification of anomalous distributed traffic indicative of cyberattacks, contributing a novel perspective to the detection process. Further extending the framework of cyber threat analysis, recent works emphasize the importance of node behavior classification within network traffic analysis. Not only does this allow for the detection of individual malicious connections, but it also identifies the nodes generating such traffic, facilitating targeted actions to mitigate threats and enhance cybersecurity [7].

Machine learning classification techniques are critical in the field of cyber intrusion detection, offering an efficient method to identify the potential security threats. These techniques enable systems to learn from past data, effectively recognizing and categorizing

cybersecurity events, which can significantly enhance the speed and accuracy of threat detection and response [8]. Moreover, recent advancements in the clustering algorithms have opened new avenues for understanding the complex datasets in cybersecurity. Clustering techniques, such as k-means and hierarchical clustering, enable the identification of hidden structures and relationships within network data, aiding in the categorization of network entities based on their properties and behaviors [9–11].

Our contribution to this field lies in applying node classification within the Memgraph in-memory graph database to analyze a comprehensive dataset of cybersecurity events from the Zeek Network Security data. Memgraph was mainly used since it is an effective tool for identifying vulnerabilities and suspicious behavior. By focusing on both detection and classification, our approach not only identifies security threats but also describes the nature of these threats, helping organizations in strengthening their defenses against the most critical vulnerabilities.

3. The Dataset

The dataset used in this work is UWF-ZeekDataFall22, available at [1]. The Cyber Range at the University of West Florida produced these Zeek Conn Log datasets. They are categorized in accordance with the MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) Framework [12]. This framework is grounded in threat models that define adversarial tactics, and currently includes 14 main tactics alongside numerous techniques and sub-techniques. This dataset offers insights into the collection of 700,340 cybersecurity records. Within these data, each entry signifies either an attack or its absence, and the distribution is nearly even: 350,339 records indicate no attack, which are benign records, while 350,001 records confirm attacks. There is a range of attack tactics present in the dataset that are discussed below [13].

Among the predominant attack methods, “Resource Development” stands out with a vast number of instances. This tactic often involves an adversary trying to establish resources that they can use to maintain their foothold and further their attack, such as creating new accounts or obtaining more software or tools.

“Reconnaissance” and “Discovery” are also significantly represented. Reconnaissance is the act of gathering preliminary data or intelligence on a target. This can involve identifying IP addresses, domain names, and network services. On the other hand, “Discovery” relates to the post-compromise phase, where the adversary actively seeks information about the attacked system or network, trying to understand what they have infiltrated and how they can exploit it.

Tactics like “Privilege Escalation” and “Defense Evasion” highlight the adversary’s attempts to gain more access and avoid detection. Privilege escalation involves techniques that allow the attacker to obtain a higher level of permissions in a system or network. In contrast, defense evasion encompasses methods to avoid being detected, including disabling security software or clearing logs.

Examples of less frequent methods include “Execution”, where malicious code is run, and “Initial Access”, indicating the point of entry for the adversary. “Command and Control” refers to the communication between the compromised systems and the attacker, while “Lateral Movement” deals with the efforts of the adversary to navigate through the network. “Persistence” highlights the attempts to maintain their foothold, and “Collection” emphasizes the gathering of valuable data from the compromised systems.

The dataset starts with a timestamp, denoted as “ts”, which marks the time of the first packet in each connection. To make this timestamp more user-friendly, the “datetime” column provides a human-readable version of the “ts” column. Every connection is uniquely identified by the “uid” column.

The dataset captures the IP address and port number of both the packet sender (“src_ip_zeek” and “src_port_zeek”) and the packet receiver (“dest_ip_zeek” and “des_port_zeek”).

The “proto” column indicates the transport layer protocol used in the connection, such as UDP. Additionally, the “service” column identifies the application protocol transmitted

over the connection, for instance, NTP. The “duration” column provides insights into how long each connection lasted, and for connections that underwent a three-way or four-way tear-down, this duration excludes the final ACK.

To understand the volume of data exchanged, the dataset includes columns like “orig_bytes” and “resp_bytes”, which represent the number of payload bytes sent by the originator and responder, respectively. The state of each connection is captured in the “conn_state” column, with values like S0, S1, and SF, each signifying a specific state of the connection.

For a better understanding of the connection’s origin, the “local_orig” column indicates whether a connection originated locally. Similarly, the “local_resp” column provides information on whether a connection was responded to locally. The dataset also accounts for potential packet losses, as indicated by the “missed_bytes” column, which represents the number of bytes missed in content gaps.

Another feature of this dataset is the “history” column. It records the state history of connections as a string of letters, each having a specific meaning, such as “s” for a SYN without the ACK bit set, “h” for a SYN + ACK, and so on. This provides a chronological record of the connection’s state transitions.

Additionally, the dataset introduces three new columns that enhance its utility:

- “label_technique”: this column provides the label for the data using the MITRE ATT&CK technique as provided by student.
- “label_tactic”: this column represents the MITRE ATT&CK tactic mapped from the student-entered technique.
- “label_binary”: this is a binary (0/1) label indicating whether the record represents an attack or not.

4. Data Preprocessing

The dataset was in the form of several CSV files without headers which had to be handled prior to data preprocessing. We initially added a predefined header, loaded the content into a DataFrame, and appended this DataFrame to a list. After processing all the files, we concatenated these individual DataFrames into a single DataFrame. A temporary “temp.csv” was used during the process and was deleted at the end.

The function, “show_tactics_summary”, provides a quick overview of the tactics found within the dataset, breaking down the count of each unique tactic label. Using the DataFrame API, it groups the entries based on their tactics and displays them in a descending order of occurrence (Table 1).

Table 1. Tactics within the dataset and their counts.

Tactic	Count
None	350,339
Resource Development	275,471
Reconnaissance	51,492
Discovery	16,819
Privilege Escalation	3066
Defense Evasion	3064
Execution	30
Initial Access	19
Command and Control	17
Lateral Movement	11
Persistence	10
Credential Access	1
Collection	1

To create the graph network in Memgraph, the dataset underwent a preprocessing sequence as shown in Figure 1. This preliminary stage aimed at producing CSV files for nodes and edges, suitable for importing into Memgraph.

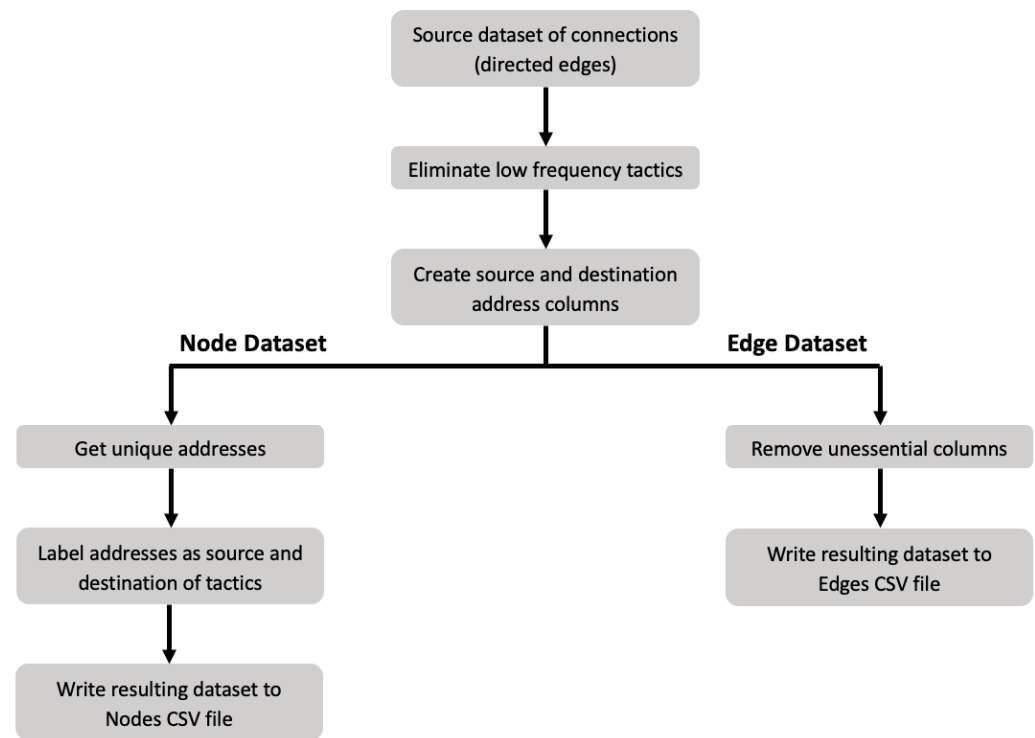


Figure 1. Preprocessing steps.

As shown in Figure 1, the “remove_tactics” function filters the dataset to retain only a specified set of tactics in the first step. This allows for a focused analysis of the more relevant tactics, which are “Resource Development”, “Reconnaissance”, “Discovery”, “Privilege Escalation”, and “Defense Evasion”, and eliminates the rest, as shown in Table 2.

Table 2. Relevant tactics and their counts.

Tactic	Count
None	350,339
Resource Development	275,471
Reconnaissance	51,492
Discovery	16,819
Privilege Escalation	3066
Defense Evasion	3064

To enhance the readability and streamline the data representation, the “add_merged_address_and_port_columns” function combines the source and destination IP addresses with their corresponding ports. This merged data provides a more holistic view of network connections.

“drop_columns” is a straightforward utility that drops unnecessary columns from the dataset. By removing these columns, we refined the dataset and made the subsequent operations faster and more memory-efficient. Figure 2 shows how we modified the dataset and reduced the number of columns.

Table 3 shows a sample of a dataset after removing the uncommon tactics and irrelevant features.

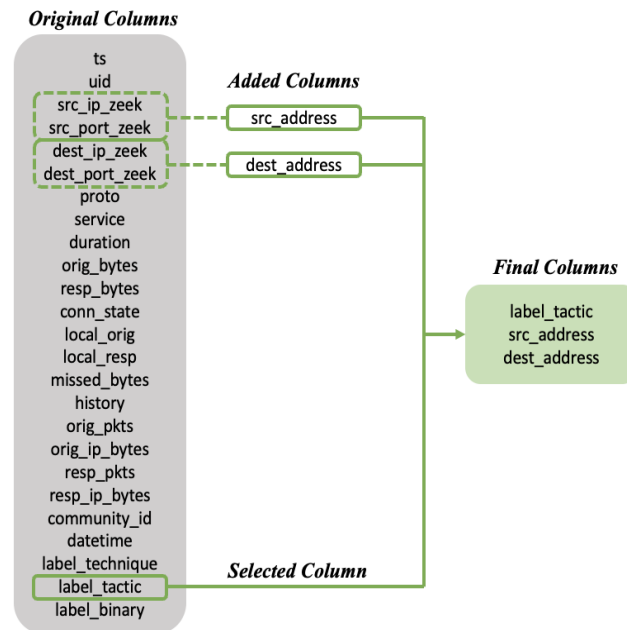


Figure 2. Modification of the dataset and reduction in the number of columns.

Table 3. Sample of the dataset after removing the uncommon tactics and irrelevant features.

Label_Tactic	Src_Address	Dest_Address
Privilege Escalation	143.88.10.11:42296	143.88.10.13:9999
Resource Development	143.88.5.11:54413	143.88.5.12:31266
Resource Development	143.88.5.11:54412	143.88.5.12:17727
Resource Development	143.88.5.11:44183	143.88.5.12:40597
Resource Development	143.88.5.11:44184	143.88.5.12:20219
Resource Development	143.88.5.11:44183	143.88.5.12:61318
Resource Development	143.88.5.11:44184	143.88.5.12:6772
Resource Development	143.88.5.11:54412	143.88.5.12:49357
Resource Development	143.88.5.11:54412	143.88.5.12:21018
Resource Development	143.88.5.11:54413	143.88.5.12:34151

The “create_nodes_df” and “create_edges_df” functions are especially important for graph-based analysis. They prepare the data for node and edge representations respectively. Within “create_nodes_df”, we process both source and destination data separately, remove duplicates, group them by unique addresses, and finally merge them. This helps in creating a comprehensive node dataset, which is crucial for the network graph visualization and analysis.

“add_ids_to_edges_df” appends unique integer identifiers to each node (source and destination). This is invaluable for graph databases and tools that require distinct identifiers for nodes. Samples of node and edge data are demonstrated in Tables 4 and 5, respectively.

Table 4. Sample of node data.

Tactics_Src	Tactics_Dest	Address	id
none	null	0.0.0.0:68	1
none	null	143.88.0.2:10048	2
none	null	143.88.0.2:10170	3
none	null	143.88.0.2:1030	4
none	null	143.88.0.2:10347	5
none	null	143.88.0.2:10428	6
none	null	143.88.0.2:10510	7
none	null	143.88.0.2:10653	8
none	null	143.88.0.2:10772	9
none	null	143.88.0.2:10877	10

Table 5. Sample of edge data.

Label_Tactic	Src_Address	Dest_Address	Src_Address_id	Dest_Address_id
Privilege Escalation	143.88.10.11:42296	143.88.10.13:9999	8589939149	17179874476
Resource Development	143.88.5.11:44183	143.88.5.12:4057	8589951182	34359770286
Resource Development	143.88.5.11:44183	143.88.5.12:6138	8589951182	17179896322
Resource Development	143.88.5.11:44183	143.88.5.12:56299	8589951182	25769829829
Resource Development	143.88.5.11:44183	143.88.5.12:21280	8589951182	25769822528
Resource Development	143.88.5.11:44183	143.88.5.12:34068	8589951182	8589956151
Resource Development	143.88.5.11:44183	143.88.5.12:5053	8589951182	8589959575
Resource Development	143.88.5.11:44183	143.88.5.12:20780	8589951182	17179888107
Resource Development	143.88.5.11:44183	143.88.5.12:15936	8589951182	17179887106
Resource Development	143.88.5.11:44183	143.88.5.12:29517	8589951182	8589955195

Lastly, the “write_csv” function is a generic utility that takes in any DataFrame and writes it to a specified CSV file path. This ensures that the output is a single CSV file, rather than multiple partitions, which is a common occurrence when dealing with large-scale data in distributed systems.

5. Memgraph

Memgraph stands out as a high-performance, in-memory graph database engineered to handle the ingestion, querying, and visualization of extensive graph datasets leveraging the power of the Cypher query language [14]. We used Memgraph to facilitate the ingestion, storage, and analysis of graph-based cybersecurity data for node classification. This process began with the transfer of CSV files into the Memgraph container using Docker commands, ensuring that the data were accessible within the database.

As shown in Figure 3, within Memgraph, the “LOAD CSV” command was employed to read the CSV files and create nodes in the database. Each node’s properties were derived from the CSV columns, including address, tactics_src, and tactics_dest. To enhance the query performance when searching for nodes based on their address, an index was created based on the address property of NetworkNode.

```
LOAD CSV FROM "/fall_2022_nodes.csv" WITH HEADER AS row
CREATE (n:NetworkNode {
  address: row.address,
  tactics_src: row.tactics_src,
  tactics_dest: row.tactics_dest})
SET n.tactics_src = row.tactics_src
SET n.tactics_dest = row.tactics_dest;
CREATE INDEX ON :NetworkNode(address);

LOAD CSV FROM "/fall_2022_edges.csv" WITH HEADER AS row
MATCH (src:NetworkNode {address: row.src_address})
MATCH (dest:NetworkNode {address: row.dest_address})
CREATE (src)-[e:COMMUNICATES_WITH]->(dest)
SET e.tactic = row.label_tactic;
```

Figure 3. Cypher query: creating nodes and edges.

The second “LOAD CSV” command established relationships between nodes, effectively modeling the connections in the graph. Each relationship was labeled with the corresponding tactic type, indicating the nature of the communication between nodes.

To confirm the successful execution of the query and loading of the data, we extracted and display a limited selection of edges using the Reconnaissance tactic, as shown in Figure 4. Figure 5 presents a graphical display of the 10,000 edges of the Reconnaissance tactic. It shows two main bigger clusters with two very distinct centroids, and some outliers from these centroids, and a few much smaller clusters with very different centroids. There appears to be at least one clear outlier in between one of the bigger clusters and one of the smaller clusters. Figure 6 shows a zoomed-in view of 13 edges.

```

MATCH path = ()-[e]->()
WHERE e.tactic = 'Reconnaissance'
RETURN *
LIMIT 10000;

```

Figure 4. Cypher query: displaying limited section of the graph.

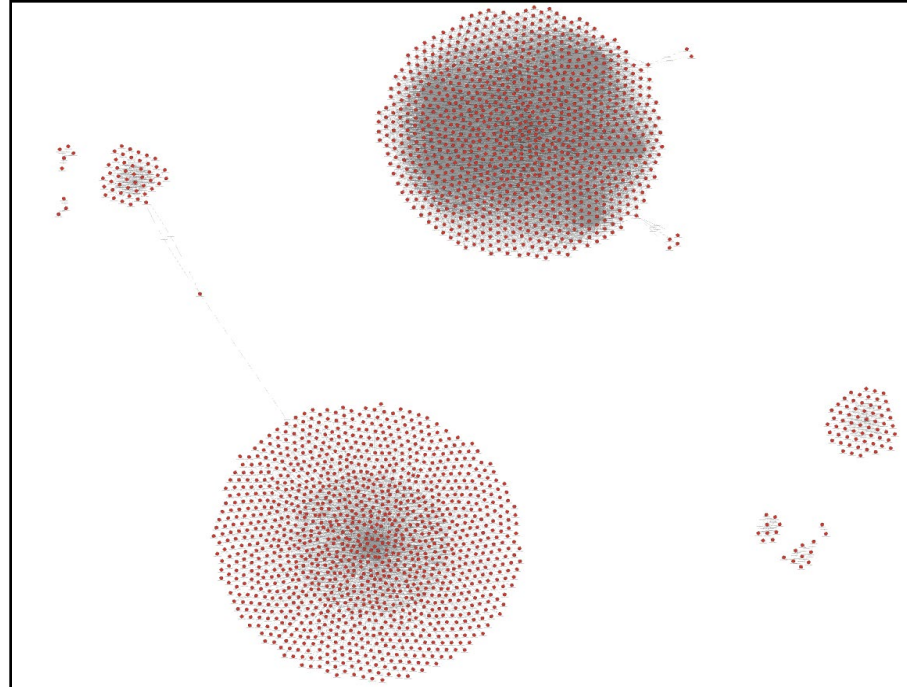


Figure 5. Ten thousand edges of the reconnaissance tactic.

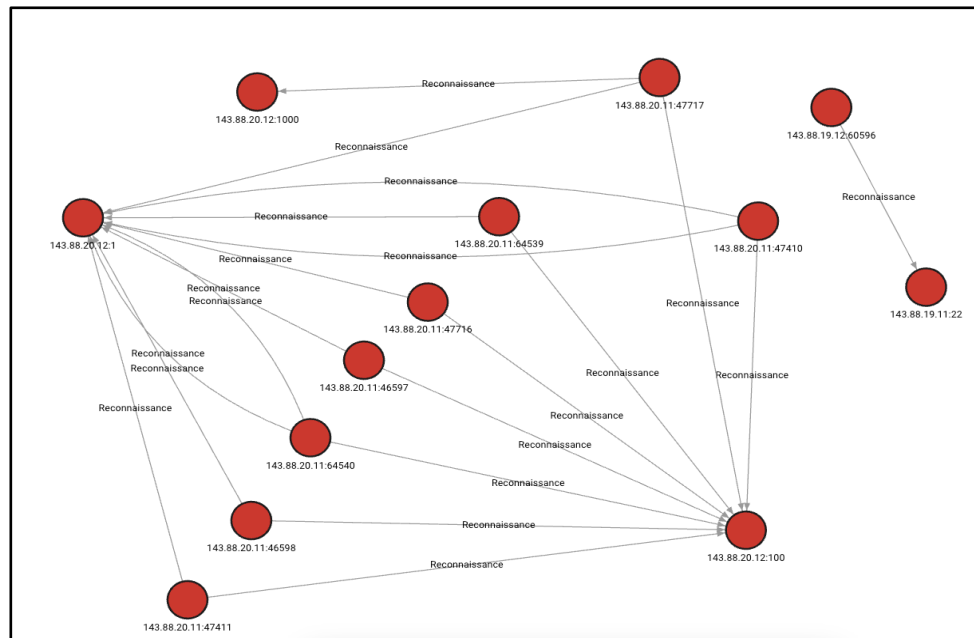


Figure 6. Ten thousand edges of the reconnaissance tactic and a zoomed-in view of 13 of those edges.

6. Graph Visualizations

Graph visualization is important in the analysis and interpretation of complex datasets, particularly in the field of cybersecurity, where understanding the web of connections can be vital in identifying vulnerabilities and threats. It allows for the intuitive recognition

of patterns, clusters, and outliers that might go unnoticed in raw tabular data. Graph visualizations can also be useful in the selection of proper algorithms like PageRank and betweenness centrality, which can highlight the nodes of particular importance. Figures 7–11 show graph representations of the important attack tactics in our dataset as well as the benign connections. A limit of 10,000 was applied where the number of edges exceeded that number. This limit, set only for visualization purposes, was selected because Memgraph was struggling with extremely large graphs. By limiting the edges to 10,000, the computational feasibility and visualization clarity were balanced.

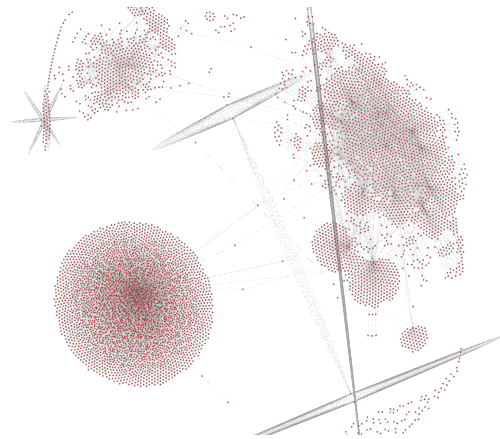


Figure 7. Ten thousand edges with benign connections.

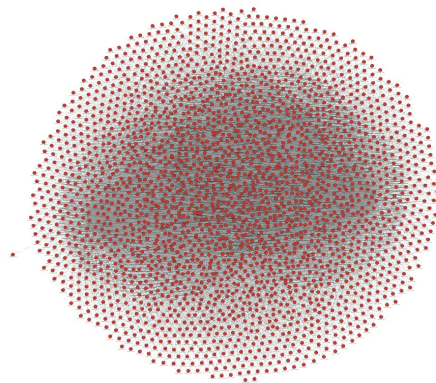


Figure 8. Ten thousand edges of the with Resource Development Tactic.

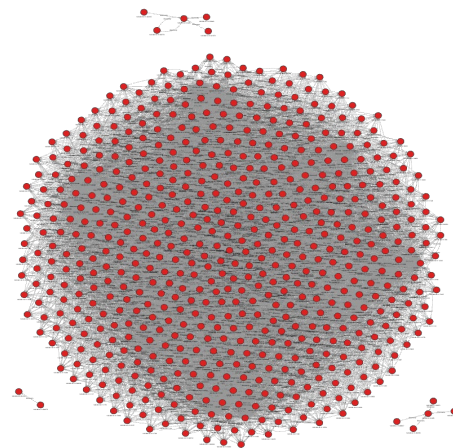


Figure 9. All edges of the discovery tactic.

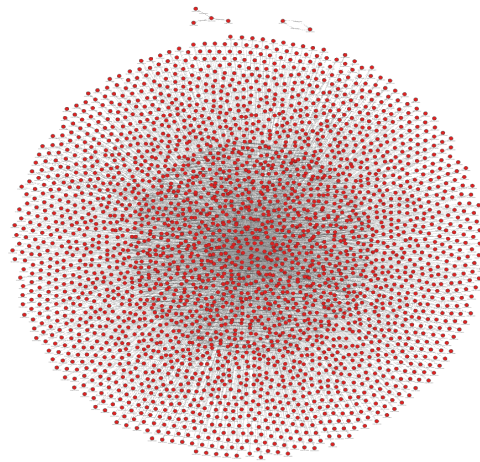


Figure 10. All edges of the privilege escalation tactic.

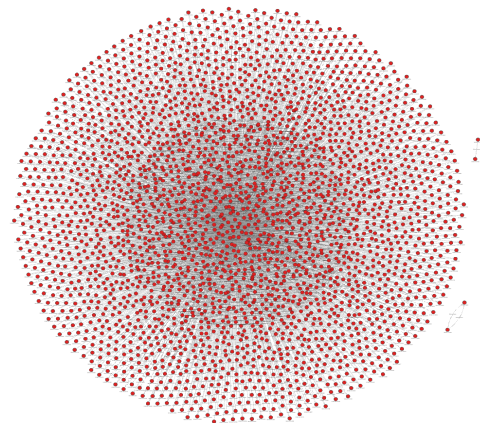


Figure 11. All edges of the defense evasion tactic.

The benign graph (Figure 7) serves as a point of reference where the connections are typically less structured and more random compared to the intentional and strategic arrangements seen in attack tactics. The Resource Development graph (Figure 8) shows a concentrated buildup of connections around certain nodes. These nodes could represent the strategic compilation of resources that are to be used in the later stages of an attack. The Reconnaissance graph (Figure 5) depicts a high level of interconnectedness, with some nodes acting as central hubs. This reflects the tactic's focus on extensively mapping the network and identifying the potential targets for exploitation. Similarly, the Discovery graph (Figure 9) features central nodes with numerous direct connections, symbolizing the tactic of exploring the compromised environment to uncover the vulnerabilities that can be used for future attacks. In the privilege escalation graph (Figure 10), certain nodes have a high concentration of connections, denoting the critical assets targeted for gaining higher-level access. The defense evasion graph (Figure 11) displays a complex web of connections as adversaries implement various techniques to avoid detection.

These visualizations illuminate the stages and strategies of cyberattacks versus benign connections. While benign activity displays a less purposeful structure, each attack tactic reveals the intent and focus: Resource Development and Discovery for setting the stage, Reconnaissance for information gathering, privilege escalation for gaining deeper access, and defense evasion for sustaining the attack without detection.

7. Graph Characterizations

A graph consists of points known as nodes, interconnected by links called edges. In this framework, each node corresponds to a distinct IP address, and each edge describes the

linkage between a source and a destination IP address, representing the tactic of the attack deployed. Thus, the graph demonstrates the trajectory of an attack from the attacker's computer to that of the target.

In the analysis of a graph, the key metrics of interest include PageRank, Degree, betweenness centrality, and Katz centrality. It is important to note that in the context of cybersecurity attacks, the graph is directed—this means that the connections between nodes (representing cyberattacks) have a specific direction, much like an arrow pointing from the attacker to the target. Consequently, the degree of each node is split into two distinct types: in-degree, which counts the number of inbound attack vectors to a node, and out-degree, which tallies the number of outbound attack vectors from a node.

7.1. PageRank

PageRank in Memgraph is an algorithm that ranks the nodes in a graph based on the structure of incoming links. It is similar to Google's PageRank algorithm, which was originally used to rank websites in search results [15]. In Memgraph, nodes are analogous to web pages, and directed edges (which indicate the direction from one node to another) are like the hyperlinks between them. The PageRank algorithm assigns a probability distribution representing the likelihood that a person randomly clicking on a link will land on any particular node [16]. In a directed graph, PageRank considers both the quantity and quality of these directed edges. A higher PageRank indicates that a node is more "important" or "influential" within the graph, based on how many connections it has and how significant those connections are. In this work, the PageRank score is used to evaluate the probability of an attack using previous attack records.

The calculation of PageRank involves an iterative process where each node's rank is updated based on the ranks of the nodes linking to it, adjusted by a damping factor that accounts for the probability of random jumps to any node. This iterative process continues until the ranks' convergence is complete. The Cypher query used to generate the PageRank scores is presented in Figure 12, and Table 6 presents the PageRank execution times for the most common tactics in the dataset.

```
MATCH n = ()-[e]->()
WHERE e.tactic = "[name of tactic]"
WITH project(n) AS proj
CALL pagerank.get(proj) YIELD node, rank
SET node.rank = rank
RETURN node.address AS address, rank
ORDER BY rank DESC;
```

Figure 12. PageRank score cypher query.

Table 6. PageRank execution time for the most common tactics.

Tactic	Row Count	Execution Time (ms)
None	80,133	1540
Resource Development	65,829	1450
Reconnaissance	18,303	529
Discovery	1060	348
Privilege Escalation	2873	357
Defense Evasion	2871	349

Tables 7–11, showing the IP addresses and their corresponding PageRank scores, categorized by different attack tactics, serve as a strategic tool in cybersecurity threat analysis and network defense planning. A high PageRank score for an IP address within a specific attack tactic category suggests that this address plays a significant and influential role within the context of that attack tactic. This information is critical as it allows for prioritization of resources and defensive measures more effectively. Cybersecurity professionals can allocate enhanced monitoring, deploy additional security controls, or take preemptive action to protect these high-risk IP addresses.

Table 7. Top 10 PageRank scores for Resource Development.

Resource Development	
Address	PageRank Score
143.88.5.12:80	0.000941348
143.88.5.12:8181	0.000491105
143.88.5.12:3500	0.000478241
143.88.5.12:445	0.000465377
143.88.5.12:21	0.000439649
143.88.5.12:8080	0.00022096
143.88.5.12:631	0.00022096
143.88.9.15:9999	0.000195231
143.88.5.15:9999	0.000105183
143.88.5.12:6697	6.65911×10^{-05}

Table 8. Top 10 PageRank scores for Reconnaissance.

Reconnaissance	
Address	PageRank Score
143.88.255.10:53	0.441815467
143.88.5.12:8080	0.001601374
143.88.10.12:6565	0.001159242
143.88.2.11:3000	0.000236343
143.88.20.12:445	0.000133542
143.88.6.12:445	0.000133322
143.88.6.12:3000	0.000120444
143.88.6.12:8181	0.000120444
143.88.20.12:8181	8.19574×10^{-05}
143.88.255.56:161	8.1811×10^{-05}

Table 9. Top 10 PageRank scores for Discovery.

Discovery	
Address	PageRank Score
143.88.18.11:3000	0.017779142
143.88.18.11:8181	0.008576373
143.88.6.12:445	0.003969507
143.88.10.12:445	0.003969507
143.88.255.56:161	0.001668997
143.88.18.11:8011	0.000917536
143.88.18.11:9100	0.000916797
143.88.18.11:1099	0.000916791
143.88.18.11:1075	0.000916791
143.88.18.11:70	0.000916789

Table 10. Top 10 PageRank scores for Privilege Escalation.

Privilege Escalation	
Address	PageRank Score
143.88.255.10:53	0.458748235
143.88.7.14:4444	0.000668235
143.88.10.13:9999	0.000348235
143.88.18.12:51372	0.000188235
143.88.18.12:40658	0.000188235
143.88.18.12:51394	0.000188235
143.88.18.12:53354	0.000188235
143.88.18.12:48608	0.000188235
143.88.18.12:49693	0.000188235
143.88.18.12:45109	0.000188235

Table 11. Top 10 PageRank scores for Defense Evasion.

Defense Evasion	
Address	PageRank Score
143.88.255.10:53	0.459067963
143.88.10.13:9999	0.000348478
143.88.6.13:4444	0.000348478
143.88.18.12:43444	0.000188366
143.88.18.12:55429	0.000188366
143.88.18.12:48298	0.000188366
143.88.18.12:56590	0.000188366
143.88.18.12:33656	0.000188366
143.88.18.12:55797	0.000188366
143.88.18.12:48346	0.000188366

7.2. Degree Centrality

Degree centrality in network analysis measures a node's direct connections within a graph, serving as an indicator of the node's activity and potential influence. In Memgraph, degree centrality can distinguish important nodes based on their interaction level. Specifically, in directed graphs, there are two sides to consider: in-degree centrality, which counts the number of incoming edges to a node, highlighting those nodes that attract the most connections, and out-degree centrality, which considers outgoing edges [17].

Unlike the PageRank algorithm, which accounts for both the quantity and quality of links, degree centrality focuses solely on the number of direct connections, thus providing a quantitative measure of a node's centrality. It is a simpler metric that does not involve iterative calculations, making it efficient for analyzing large networks quickly. In Memgraph's context, understanding nodes with high in-degree or out-degree centrality can be critical, particularly in applications like cybersecurity, where they might signify the potential points for the spread of information or vulnerabilities for network attacks [18]. The Cypher query used to obtain in-degree centrality is presented in Figure 13.

```
MATCH n = ()-[e]->()
WHERE e.tactic = "[name of tactic]"
WITH project(n) AS proj
CALL degree_centrality.get(proj, 'in') YIELD node, degree
SET node.in_degree = degree
RETURN node.address AS address, degree
ORDER BY degree DESC;
```

Figure 13. In-degree centrality cypher query.

Tables 12–17 present the top 10 IP addresses ranked by their in-degree centrality scores. The in-degree centrality scores reveal the number of direct connections to an IP address, reflecting its potential role as a target in cyberattacks. For example, the IP address "143.88.255.10:53" consistently appears with high in-degree centrality across several tactics, suggesting that it is an essential piece of network infrastructure that is frequently targeted.

Table 12. Top 10 in-degree centrality scores for benign connections.

None	
Address	In-Degree
10.0.10.1:53	3.441771078
143.88.1.1:53	0.276543703
143.88.11.1:53	0.108807967
8.8.8.8:53	0.093033994
8.8.4.4:53	0.091860929
143.88.255.10:53	0.036152848
ff02::1:2:547	0.034642839
143.88.0.41:53	0.017084311
172.28.128.255:138	0.010582539
172.28.128.255:137	0.008922777

Table 13. Top 10 in-degree centrality scores for Resource Development.

Resource Development	
Address	In-Degree
143.88.5.12:80	0.001200097
143.88.5.12:3500	0.000622835
143.88.5.12:8181	0.000607644
143.88.5.12:445	0.000607644
143.88.5.12:21	0.00054688
143.88.5.12:8080	0.000288631
143.88.5.12:631	0.000288631
143.88.9.15:9999	0.000227867
143.88.5.12:6667	0.000151911
143.88.5.12:4676	0.000121529

Table 14. Top 10 in-degree centrality scores for Reconnaissance.

Reconnaissance	
Address	In-Degree
143.88.255.10:53	2.342203038
143.88.10.12:6565	0.005682439
143.88.5.12:8080	0.003387608
143.88.2.11:3000	0.000874221
143.88.20.12:445	0.000710305
143.88.20.12:1556	0.000655666
143.88.20.12:5822	0.000655666
143.88.20.12:5405	0.000655666
143.88.20.12:6667	0.000601027
143.88.20.12:18040	0.000601027

Table 15. Top 10 in-degree centrality scores for Discovery.

Discovery	
Address	In-Degree
143.88.18.11:3000	0.029272899
143.88.18.11:8011	0.019830028
143.88.18.11:646	0.018885741
143.88.18.11:9100	0.018885741
143.88.18.11:70	0.018885741
143.88.18.11:1075	0.018885741
143.88.18.11:992	0.018885741
143.88.18.11:7496	0.018885741
143.88.18.11:1099	0.018885741
143.88.18.11:50002	0.017941454

Table 16. Top 10 in-degree centrality scores for Privilege Escalation.

Privilege Escalation	
Address	In-Degree
143.88.255.10:53	1.065807799
143.88.7.14:4444	0.001044568
143.88.10.13:9999	0.000696379
143.88.18.12:51372	0
143.88.18.12:40658	0
143.88.18.12:51394	0
143.88.18.12:53354	0
143.88.18.12:48608	0
143.88.18.12:49693	0
143.88.18.12:45109	0

Table 17. Top 10 in-degree centrality scores for Defense Evasion.

Defense Evasion	
Address	In-Degree
143.88.255.10:53	1.066550523
143.88.6.13:4444	0.000696864
143.88.10.13:9999	0.000348432
143.88.18.12:43444	0
143.88.18.12:55429	0
143.88.18.12:48298	0
143.88.18.12:56590	0
143.88.18.12:33656	0
143.88.18.12:55797	0
143.88.18.12:48346	0

From Tables 16 and 17 it can be noted that only three nodes are at the receiving end of these tactics.

The Cypher query used to obtain the out-degree centrality is presented in Figure 14.

```
MATCH n = ()-[e]->()
WHERE e.tactic = "[name of tactic]"
WITH project(n) AS proj
CALL degree centrality.get(proj, 'out') YIELD node, degree
SET node.out_degree = degree
RETURN node.address AS address, degree
ORDER BY degree DESC;
```

Figure 14. Out-degree centrality cypher query.

Tables 18–23 present the top 10 IP addresses ranked by their out-degree centrality scores. The out-degree centrality scores reveal the number of direct connections from an IP address.

Table 18. Top 10 out-degree centrality scores for benign connections.

None	
Address	Out-Degree
fe80::250:56ff:fe9e:5457:546	0.021576898
172.28.128.3:138	0.010582539
172.28.128.3:137	0.008922777
143.88.1.50:138	0.008548395
143.88.1.50:137	0.007350372
143.88.11.10:35104	0.007025907
143.88.11.10:61612	0.006988469
143.88.11.10:50888	0.006813757
143.88.11.10:53887	0.006701443
143.88.11.10:42982	0.006639046

Table 19. Top 10 out-degree centrality scores for Resource Development.

Resource Development	
Address	Out-Degree
143.88.5.11:54413	1.045284681
143.88.5.11:54412	1.045254299
143.88.5.11:44183	1.044874521
143.88.5.11:44184	1.044646655
143.88.5.11:53748	4.55733×10^{-05}
143.88.5.11:52264	3.03822×10^{-05}
143.88.5.11:44634	3.03822×10^{-05}
143.88.5.11:44216	3.03822×10^{-05}
143.88.5.11:37695	3.03822×10^{-05}
143.88.5.11:54466	3.03822×10^{-05}

Table 20. Top 10 out-degree centrality scores for Reconnaissance.

Reconnaissance	
Address	Out-Degree
143.88.20.11:47716	0.058245001
143.88.20.11:64539	0.058081084
143.88.20.11:46598	0.057480057
143.88.20.11:47717	0.057425418
143.88.20.11:46597	0.05731614
143.88.20.11:64540	0.057206863
143.88.20.11:47410	0.057152224
143.88.20.11:47411	0.057042946
143.88.7.11:42979	0.000655666
143.88.7.11:53007	0.000546388

Table 21. Top 10 out-degree centrality scores for Discovery.

Discovery	
Address	Out-Degree
143.88.18.12:59247	1.003777148
143.88.18.12:38565	1.000944287
143.88.18.12:51277	0.997167139
143.88.18.12:39128	0.996222852
143.88.18.12:51278	0.993389991
143.88.18.12:52794	0.992445703
143.88.18.12:52795	0.992445703
143.88.18.12:50198	0.992445703
143.88.18.12:59246	0.991501416
143.88.18.12:59017	0.987724268

Table 22. Top 10 out-degree centrality scores for Privilege Escalation.

Privilege Escalation	
Address	Out-Degree
143.88.18.12:52620	0.001044568
143.88.18.12:34388	0.000696379
143.88.18.12:60712	0.000696379
143.88.18.12:36691	0.000696379
143.88.18.12:56708	0.000696379
143.88.18.12:36876	0.000696379
143.88.18.12:41039	0.000696379
143.88.18.12:33548	0.000696379
143.88.18.12:51245	0.000696379
143.88.18.12:48856	0.000696379

Table 23. Top 10 out-degree centrality scores for Defense Evasion.

Defense Evasion	
Address	Out-Degree
143.88.18.12:52620	0.001045296
143.88.18.12:37628	0.000696864
143.88.18.12:59767	0.000696864
143.88.18.12:55832	0.000696864
143.88.18.12:43722	0.000696864
143.88.18.12:53705	0.000696864
143.88.18.12:53096	0.000696864
143.88.18.12:53147	0.000696864
143.88.18.12:36559	0.000696864
143.88.18.12:38070	0.000696864

By incorporating both in-degree and out-degree centrality in an analysis, it is possible to optimize the distribution of security resources and implement strategic defenses to

protect the nodes that are either pivotal in network operations or are potential targets for malicious activities.

7.3. Betweenness Centrality

Betweenness centrality is a measure used in network analysis to determine the importance of nodes in a graph. It quantifies the number of times that a node acts as a bridge along the shortest path between two other nodes. In essence, the nodes with high betweenness centrality have a significant influence on the flow of information (or any other resource) through the network because more shortest paths pass through them [19].

In Memgraph, betweenness centrality can be used to identify the key nodes within a graph that are critical for maintaining the network's connectivity. For example, in cybersecurity, nodes with high betweenness centrality might be those through which the majority of network traffic flows, making them the potential targets for attacks. By identifying these nodes, the network administrators can prioritize them for additional security measures [20].

The calculation of betweenness centrality for a node involves counting how many of the shortest paths from all nodes to all others pass through that node. The centrality score is usually normalized by dividing by the number of pairs of nodes, excluding the node of interest, to account for the size of the network.

For example, in the context of the cybersecurity within Memgraph, betweenness centrality could be used to analyze the network traffic flow to detect the anomalies or potential security breaches. By examining the nodes that frequently occur on the shortest paths of network traffic, it is possible to monitor and secure the most critical components of a network infrastructure, possibly predicting and preventing cyberattacks before they happen. The Cypher query used to obtain betweenness centrality is presented in Figure 15.

```
MATCH n = ()-[e]->()
WHERE e.tactic = "[name of tactic]"
WITH project(n) AS proj
CALL betweenness_centrality.get(proj)
YIELD node, betweenness_centrality
Set node.betweenness_centrality = betweenness_centrality
RETURN node.address AS address, betweenness_centrality
ORDER BY betweenness_centrality DESC;
```

Figure 15. Betweenness Centrality cypher query.

In the application of betweenness centrality to our network graph, a contrast was observed between the results for “attack tactics” and “benign connections”. The zero centrality scores for all nodes under attack tactics indicate that these nodes are not situated on the paths most traveled during these events, suggesting that such tactics might be non-centralized and spread out. In contrast, the nodes associated with benign connections show non-zero centrality (Table 24), underscoring their role in regular traffic flow and implying an interconnected network during normal operations. This differentiation in centrality metrics could reflect the underlying structure and behavior patterns unique to both benign and malicious network activities.

Table 24. Top 10 betweenness centrality scores for benign connections.

Address	Betweenness Centrality
143.88.11.10:61612	1.61×10^7
143.88.11.10:42982	1.53×10^7
143.88.11.10:35104	8.81×10^8
143.88.11.14:2030	7.37×10^8
143.88.11.14:16992	6.98×10^8
143.88.11.11:3	9.34×10^{10}
143.88.11.13:57294	3.89×10^{10}
143.88.11.11:5353	3.11×10^{10}
143.88.11.11:51493	2.98×10^{10}
143.88.11.14:49158	2.86×10^{10}

7.4. Katz Centrality

Katz centrality, as implemented in Memgraph, is a measure of centrality that extends beyond immediate direct connections to consider the total number of walks through a node, thus capturing a wider range of influences in a graph. Unlike other centrality measures that only consider the shortest path, Katz centrality accounts for an infinite series of walks, where each successive path length is penalized by a factor α , which is less than one. This ensures that shorter paths contribute more to the centrality score than longer paths, with the attenuation factor diminishing the influence of each additional step in the path [16]. Katz centrality is particularly useful in networks where pathways of influence extend beyond immediate neighbors, such as citation networks or the World Wide Web. It is also applicable in contexts where traditional centrality measures might not be effective, such as in directed acyclic graphs.

In Memgraph, Katz centrality can be calculated based on the work of Alexander van der Grinten and others, which focuses on the scalable computation of Katz centrality in both static and dynamic graphs. The method involves iterative approximations to refine the upper and lower bounds of centrality scores, ensuring that the centrality rankings are accurate, even if the exact centrality values are not guaranteed to be precise [21]. The Cypher query used to obtain Katz centrality is presented in Figure 16.

```
MATCH n = ()-[e]->()
WHERE e.tactic = "Defense Evasion"
WITH project(n) AS proj
CALL katz_centrality.get(proj)
YIELD node, rank
Set node.katz_centrality = rank
RETURN node.address AS address, rank
ORDER BY rank DESC;
```

Figure 16. Katz centrality cypher query.

Tables 25–30 present the top 10 IP addresses ranked by their Katz centrality scores for the common attack tactics as well as the benign connections reflecting the relative importance of different nodes in the network. For benign connections (None), high Katz centrality scores (e.g., for addresses like 10.0.10.1:53) indicate nodes that are central in the network communication during regular activities. These are likely to be key infrastructure components such as DNS servers or gateways that handle a lot of traffic and have many connections.

In the context of specific attack tactics like Resource Development, Reconnaissance, and others, the Katz centrality scores are lower compared to benign traffic. This suggests that the nodes involved in these tactics are less central to the network's overall structure. They may represent specialized or occasional communications rather than regular traffic, reflecting the more undercover nature of these activities.

Table 25. Top 10 Katz centrality scores for benign connections.

None	
Address	Katz Centrality
10.0.10.1:53	55,159.2
143.88.1.1:53	4432
143.88.11.1:53	1743.8
8.8.8.8:53	1491
8.8.4.4:53	1472.2
143.88.255.10:53	579.4
ff02::1:2:547	555.2
143.88.0.41:53	273.8
172.28.128.255:138	169.6
172.28.128.255:137	143

Table 26. Top 10 Katz centrality scores for Resource Development.

Resource Development	
Address	Katz Centrality
143.88.5.12:80	15.8
143.88.5.12:3500	8.2
143.88.5.12:8181	8
143.88.5.12:445	8
143.88.5.12:21	7.2
143.88.5.12:631	3.8
143.88.5.12:8080	3.8
143.88.9.15:9999	3
143.88.5.12:6667	2
143.88.5.12:40060	1.6

Table 27. Top 10 Katz centrality scores for Reconnaissance.

Reconnaissance	
Address	Katz Centrality
143.88.255.10:53	8573.4
143.88.10.12:6565	20.8
143.88.5.12:8080	12.4
143.88.2.11:3000	3.2
143.88.20.12:445	2.6
143.88.20.12:1556	2.4
143.88.20.12:5405	2.4
143.88.20.12:5822	2.4
143.88.20.12:6667	2.2
143.88.20.12:119	2.2

Table 28. Top 10 Katz centrality scores for Discovery.

Discovery	
Address	Katz Centrality
143.88.18.11:3000	6.2
143.88.18.11:8011	4.2
143.88.18.11:1075	4
143.88.18.11:7496	4
143.88.18.11:1099	4
143.88.18.11:70	4
143.88.18.11:9100	4
143.88.18.11:646	4
143.88.18.11:992	4
143.88.18.11:1093	3.8

Table 29. Top 10 Katz centrality scores for Privilege Escalation.

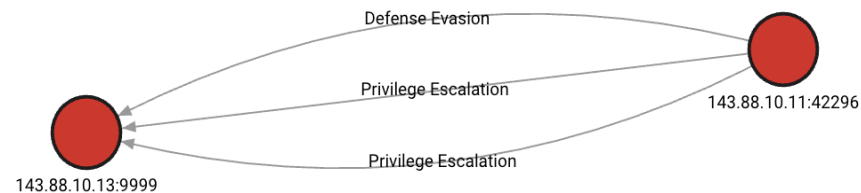
Privilege Escalation	
Address	Katz Centrality
143.88.255.10:53	612.2
143.88.7.14:4444	0.6
143.88.10.13:9999	0.4
143.88.18.12:52537	0
143.88.18.12:49994	0
143.88.7.11:53302	0
143.88.18.12:52575	0
143.88.18.12:52556	0
143.88.7.11:53300	0
143.88.18.12:41414	0

Table 30. Top 10 Katz centrality scores for Defense Evasion.

Defense Evasion	
Address	Katz Centrality
143.88.255.10:53	612.2
143.88.6.13:4444	0.4
143.88.10.13:9999	0.2
143.88.18.12:52975	0
143.88.18.12:35501	0
143.88.18.12:58899	0
143.88.18.12:34066	0
143.88.18.12:56178	0
143.88.18.12:52941	0
143.88.18.12:44901	0

Addresses that repeatedly appear across different attack tactics with non-zero centrality scores (like 143.88.255.10:53) may indicate nodes that are potential control points or valuable assets targeted across various stages of cyberattacks. These insights can be used to identify the critical nodes for further security considerations, as well as to understand the behavior and propagation patterns of potential threats within the network infrastructure.

Figure 17 is a visualization of the node with address 143.88.10.13:9999 that has both privilege escalation and defense evasion Katz centrality.

**Figure 17.** Katz centrality for the node with 143.88.10.13:9999 address.

8. Node Classification

Graph Neural Networks (GNNs) were used for node classification to determine whether IP address and port combinations within datasets are indicative of an attack or benign activity. Node classification is pivotal in determining if a node acts as the origin or target of an attack tactic. Within the Memgraph environment, node classification is executed utilizing the torch open-source machine learning library.

8.1. Parameters Used for Node Classification

To train the models, a range of parameters were configured, including the architecture of the hidden feature layers, the specific layer type utilized, the learning rate for model optimization, and the total number of training cycles or epochs.

8.1.1. Hidden Feature Layers

Hidden features refer to the dimensions of the hidden layers within GNNs that process and transform the input features into an embedding space. The size of the hidden features is a key parameter in model configuration, determining the complexity and capacity of the GNN to capture the patterns in the data. For our experiments, we employed hidden feature sizes of [8, 8] and [16, 16], which represent the dimensions of the node embeddings in the respective hidden layers. The feature size of [8, 8] was chosen to facilitate quicker convergence. By limiting the dimensionality of the hidden layers, the computational complexity was reduced, allowing the model to process and learn from the graph data more efficiently. In contrast, the [16, 16] feature size was selected to enhance the model's capacity to capture the more intricate relationships and nuanced patterns within the graph.

8.1.2. Layer Type

Layer types in Memgraph define the architecture of the GNNs and are crucial for the model's ability to learn from the graph-structured data. We utilized various layer types to best capture the relational patterns within our dataset.

1. Graph Attention Networks with Jumping Knowledge (GATJK): This layer type combines the strengths of graph attention networks (GAT) with jumping knowledge (JK) networks. GAT layers allow nodes to attend over their neighborhoods' features, assigning more weight to the more important nodes during feature aggregation. The JK extension enables the network to leverage neighborhood information from different neighborhood ranges, improving the model's performance on graphs with complex structures. We used GATJK for its capacity to capture both the local and more distant relational information within the graph. GATJK is particularly adept at handling intricate and hierarchical relationships within our network data.
2. Graph Attention Networks (GAT and GATv2): These layers use attention mechanisms to weigh the influence of neighboring nodes. This is particularly useful in differentiating the importance of the various connections that a node might have. GATv2 is an iteration of GAT with improved attention mechanisms that allow for more nuanced weighting of neighbor contributions. These layers were chosen for their effectiveness in capturing local graph structures and highlighting the critical nodes that play pivotal roles in network behaviors.
3. GraphSAGE: Short for Graph Sample and Aggregate, this approach generalizes to unseen nodes by learning a function that generates embeddings by sampling and aggregating the features from a node's local neighborhood. We selected GraphSAGE for its inductive capabilities, which are particularly effective when working with graphs that evolve over time, such as in network security.

The choice to use GATJK, SAGE, and GATv2 was motivated by the different properties of our network data and the need to be experimented with both transductive and inductive learning paradigms to determine which yields the best performance for our node classification tasks.

8.1.3. Learning Rate

The learning rate is a critical hyperparameter that influences the rate at which the model learns from the training data. It determines the step size at each iteration while moving toward a minimum of the loss function. In our work, we experimented with different learning rates to find the optimal value that minimizes the loss while preventing the model from overshooting the minimum. We specifically used the learning rates of 0.1, 0.001, and 0.0005 in various training scenarios. Initially a higher learning rate of 0.1 was selected to accelerate the model's training. This expedited the exploration of the parameter space, allowing the model to quickly descend towards a local minimum of the loss function. A medium learning rate of 0.001 was then employed to strike a balance between the convergence speed and stability. Towards the later stages of training, a lower learning rate of 0.0005 was tested to fine-tune the model parameters. We systematically evaluated each learning rate across multiple training scenarios, assessing its impact using the metrics such as classification accuracy and loss curve.

8.1.4. Number of Epochs

The number of epochs is essential in determining how well the model learns from the data without being underfit or overfit. We experimented with three numbers of epochs, starting with five to allow the model to grasp the basics of the patterns in the data swiftly. We then tested higher numbers, such as 10 and 100 epochs, to observe the long-term learning trends and to determine the point of diminishing returns where additional epochs do not equate to better generalization on unseen data. The selection of the number of epochs in our node classification study was driven by a combination of theoretical principles and

empirical observations. Starting with a low number of epochs allowed us to establish a baseline, while moderate and high epochs facilitated a deeper learning process.

8.2. Feature Selection and Preprocessing for Node Classification

In the initial phase of the node classification, the built-in modules described in the previous section were applied to calculate the centrality metrics for each node, as shown in the Cypher query presented in Figure 18.

```
CALL degree_centrality.get("in") YIELD node, degree SET node.in_degree = degree;
CALL degree_centrality.get("out") YIELD node, degree SET node.out_degree = degree;
CALL pagerank.get() YIELD node, rank SET node.rank = rank;
CALL betweenness_centrality.get() YIELD node, betweenness_centrality SET node.betweenness_centrality = betweenness_centrality;
CALL katz_centrality.get() YIELD node, rank SET node.katz_centrality = rank;
```

Figure 18. Calculating the centrality metrics for each node.

The nodes were updated with feature arrays, namely Features1 and Features2, which contained the calculated centrality metrics (Figure 19). These feature arrays were essential for the machine learning tasks and contained the following metrics:

```
MATCH (node) SET node.features1 = [node.in_degree, node.out_degree, node.rank];
MATCH (node) SET node.features2 = [node.in_degree, node.out_degree, node.rank, node.katz_centrality, node.betweenness_centrality];
```

Figure 19. Features arrays.

Features1 Array:

- In-Degree: this reflects the number of incoming connections to a node, indicating its popularity or influence within the network.
- Out-Degree: this represents the number of outgoing connections from a node, highlighting its role in disseminating information.
- PageRank: This quantifies a node's importance by considering both its incoming and outgoing links. Nodes with a high PageRank are influential in the network.

Features2 Array:

Feature1 Array plus Katz centrality and betweenness centrality

- Katz centrality: this measures a node's centrality by considering paths of different lengths, providing a more comprehensive view of influence.
- Betweenness centrality: this evaluates a node's significance in terms of facilitating the communication between other nodes, identifying critical intermediaries.

We initially queried the graph to find the tactic combinations present within the dataset. Two separate queries were executed to extract and document the unique values of tactics_dest (Figure 20) and tactics_src (Figure 21), representing the destination and source types in the network data, respectively. This information was later used when we were assigning numerical classes to the nodes based on their tactics_src and tactics_dest properties.

```
MATCH (n) WITH DISTINCT n.tactics_dest AS DestinationType
RETURN DestinationType;
```

Figure 20. Extracting unique values of tactics_dest.

```
MATCH (n) WITH DISTINCT n.tactics_src AS SourceType
RETURN SourceType;
```

Figure 21. Extracting unique values of tactics_src.

The destination types are:

1. None
2. Resource Development
3. Defense evasion, privilege escalation
4. Discovery
5. Reconnaissance
6. Discovery, Reconnaissance
7. Reconnaissance, Resource Development
8. Privilege escalation
9. Defense evasion, privilege escalation, Reconnaissance, none
10. Defense evasion

The source types are:

1. None
2. Resource Development
3. Reconnaissance
4. Discovery
5. Defense evasion, privilege escalation
6. Discovery, Reconnaissance
7. Defense evasion, Discovery, privilege escalation
8. Discovery, Reconnaissance
9. Privilege escalation
10. Defense evasion

Multiple MATCH statements were employed to assign numerical classes to the nodes based on their `tactics_src` and `tactics_dest` properties (Figure 22). This classification categorized the nodes according to the tactics that they represented, enabling the subsequent machine learning tasks to differentiate between them.

```
MATCH (n) WHERE n.tactics_dest = "none" SET n.dest_class = 1;
MATCH (n) WHERE n.tactics_dest = "Reconnaissance" SET n.dest_class = 2;
MATCH (n) WHERE n.tactics_dest = "Discovery" SET n.dest_class = 3;
MATCH (n) WHERE n.tactics_dest = "Resource Development" SET n.dest_class = 4;
MATCH (n) WHERE n.tactics_dest = "Privilege Escalation" SET n.dest_class = 5;
MATCH (n) WHERE n.tactics_dest = "Defense Evasion" SET n.dest_class = 6;
MATCH (n) WHERE n.tactics_dest = "" SET n.dest_class = 7;
MATCH (n) WHERE n.tactics_dest = "Defense Evasion, Privilege Escalation" SET n.dest_class = 8;
MATCH (n) WHERE n.tactics_dest = "Discovery, Reconnaissance" SET n.dest_class = 9;
MATCH (n) WHERE n.tactics_dest = "Reconnaissance, Resource Development" SET n.dest_class = 10;
MATCH (n) WHERE n.tactics_dest = "Defense Evasion, Privilege Escalation, Reconnaissance, none" SET n.dest_class = 11;

MATCH (n) WHERE n.tactics_src = "none" SET n.src_class = 1;
MATCH (n) WHERE n.tactics_src = "Reconnaissance" SET n.src_class = 2;
MATCH (n) WHERE n.tactics_src = "Discovery" SET n.src_class = 3;
MATCH (n) WHERE n.tactics_src = "Resource Development" SET n.src_class = 4;
MATCH (n) WHERE n.tactics_src = "Privilege Escalation" SET n.src_class = 5;
MATCH (n) WHERE n.tactics_src = "Defense Evasion" SET n.src_class = 6;
MATCH (n) WHERE n.tactics_src = "" SET n.src_class = 7;
MATCH (n) WHERE n.tactics_src = "Defense Evasion, Privilege Escalation" SET n.src_class = 8;
MATCH (n) WHERE n.tactics_src = "Discovery, Reconnaissance" SET n.src_class = 9;
MATCH (n) WHERE n.tactics_src = "Defense Evasion, Discovery, Privilege Escalation" SET n.src_class = 10;
```

Figure 22. Assigning classes.

To prepare the data for the node classification, the `node_classification.set_model_parameters` procedure was invoked. This procedure allowed for the configuration of the GNN model,

specifying the parameters such as layer type, learning rate, hidden feature size, and others (as shown in Figure 23). Once the model was configured, the `node_classification.train` statement was used to train the GNN model on the dataset, with the defined features and classes (Figure 24).

```
CALL node_classification.set_model_parameters({
  layer_type: "GATJK",
  learning_rate: 0.1,
  hidden_features_size: [8, 8],
  class_name: "dest_class",
  features_name: "features1",
  num_epochs: 5,
  console_log_freq: 1,
  checkpoint_freq: 1,
  device_type: "cuda",
  path_to_model: "/tmp/torch_models/model_GATJK_",
  weight_decay: 0.0005,
  split_ratio: 0.8,
  metrics: ["loss", "accuracy", "f1_score", "precision", "recall",
    "num_wrong_examples"],
  aggregator: "mean"
}) YIELD * RETURN *;
```

Figure 23. Specifying parameters for node classification.

```
CALL node_classification.train(5)
PROCEDURE MEMORY UNLIMITED
YIELD * RETURN *;
```

Figure 24. Node classification.

8.3. Node Classification Results and Discussion

In this study, we assessed the impact of varying configurations on the performance of graph neural network (GNN) models for node classification. To ensure the robustness and effectiveness of our node classification approach, we conducted a comprehensive exploration of various parameter sets, as summarized in Table 31. We systematically adjusted the various aspects of our Graph Neural Network (GNN) model to assess their impact on the node classification performance. The key model parameters adjusted included the feature set (features1 and features2), layer type (GATJK, SAGE and, GATv2), hidden feature size ([8-8] and [16-16]), learning rate (0.1, 0.001, and 0.0005), and epochs (5, 10, and 100). By varying these parameters, we aimed to identify the most effective configuration that would yield the optimal classification results for our dataset.

Figure 25a–r show the plots that were generated by running the GNN model with each of the configurations in Table 31, recording the performance metrics (accuracy, precision, recall, F1 score) for both the source and the destination data, as well as the loss metrics (training loss validation loss) after each epoch. These metrics were then plotted over the number of epochs to visualize the training process and the outcomes for both the source and destination, allowing for a comparative analysis of the model's performance across the different datasets. By comparing the plots presented in Figure 25a–r, the following observations were made.

Table 31. Parameter sets for node classification.

Features 1/2	Source/Dest	Hidden Features	Layer Type	Learning Rate	Epochs
1	Dest	[8, 8]	"GATJK"	0.1	100
1	Source	[8, 8]	"GATJK"	0.1	100
2	Dest	[8, 8]	"GATJK"	0.1	100
2	Source	[8, 8]	"GATJK"	0.1	100
1	Dest	[8, 8]	"GATJK"	0.1	5
1	Source	[8, 8]	"GATJK"	0.1	5
2	Dest	[8, 8]	"GATJK"	0.1	5
2	Source	[8, 8]	"GATJK"	0.1	5
1	Dest	[16, 16]	"GATJK"	0.1	100
1	Source	[16, 16]	"GATJK"	0.1	100
2	Dest	[16, 16]	"GATJK"	0.1	100
2	Source	[16, 16]	"GATJK"	0.1	100
1	Dest	[8, 8]	"GATJK"	0.001	100

Table 31. Cont.

Features 1/2	Source/Dest	Hidden Features	Layer Type	Learning Rate	Epochs
1	Source	[8, 8]	"GATJK"	0.001	100
2	Dest	[8, 8]	"GATJK"	0.001	100
2	Source	[8, 8]	"GATJK"	0.001	100
1	Dest	[8, 8]	"GATJK"	0.001	5
1	Source	[8, 8]	"GATJK"	0.001	5
2	Dest	[8, 8]	"GATJK"	0.001	5
2	Source	[8, 8]	"GATJK"	0.001	5
1	Dest	[8, 8]	"SAGE"	0.0005	10
1	Source	[8, 8]	"SAGE"	0.0005	10
2	Dest	[8, 8]	"SAGE"	0.0005	10
2	Source	[8, 8]	"SAGE"	0.0005	10
1	Dest	[8, 8]	"GATv2"	0.0005	100
1	Source	[8, 8]	"GATv2"	0.0005	100
2	Dest	[8, 8]	"GATv2"	0.0005	100
2	Source	[8, 8]	"GATv2"	0.0005	100
1	Dest	[8, 8]	"GATv2"	0.0005	10
1	Source	[8, 8]	"GATv2"	0.0005	10
2	Dest	[8, 8]	"GATv2"	0.0005	10
2	Source	[8, 8]	"GATv2"	0.0005	10
1	Dest	[8, 8]	"GATv2"	0.0005	5
1	Source	[8, 8]	"GATv2"	0.0005	5
2	Dest	[8, 8]	"GATv2"	0.0005	5
2	Source	[8, 8]	"GATv2"	0.0005	5

(a)

Features Name: features1, Hidden Feature Size: [8-8], Layer Type: GATJK, Learning Rate: 0.1, Epochs: 5

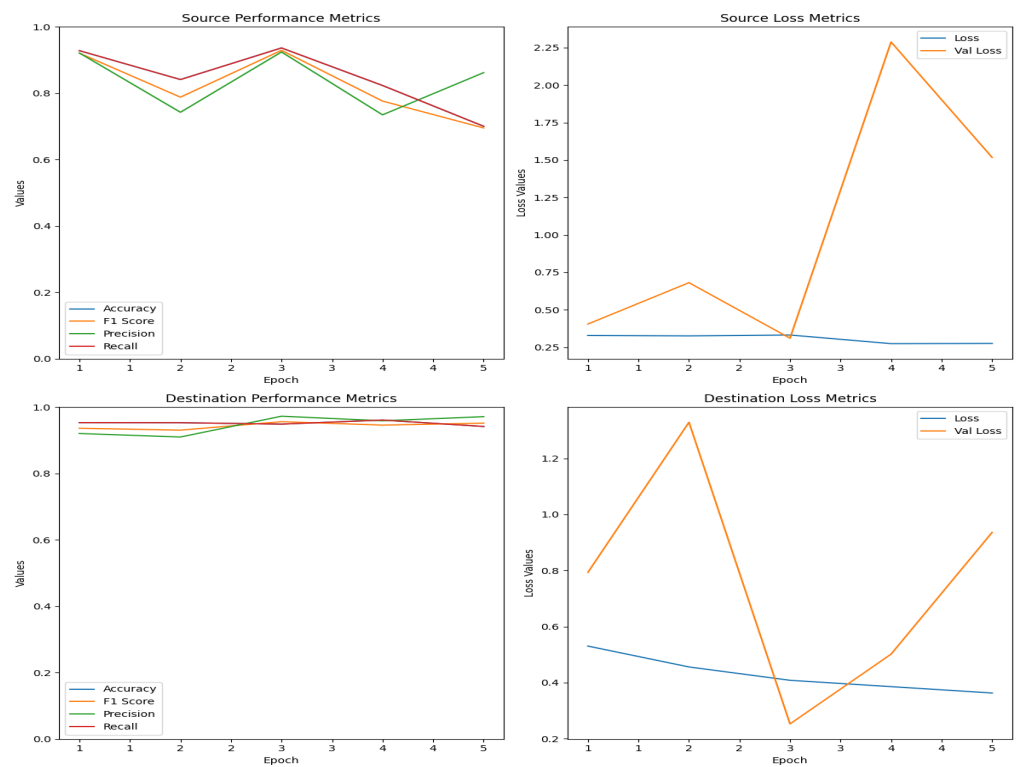
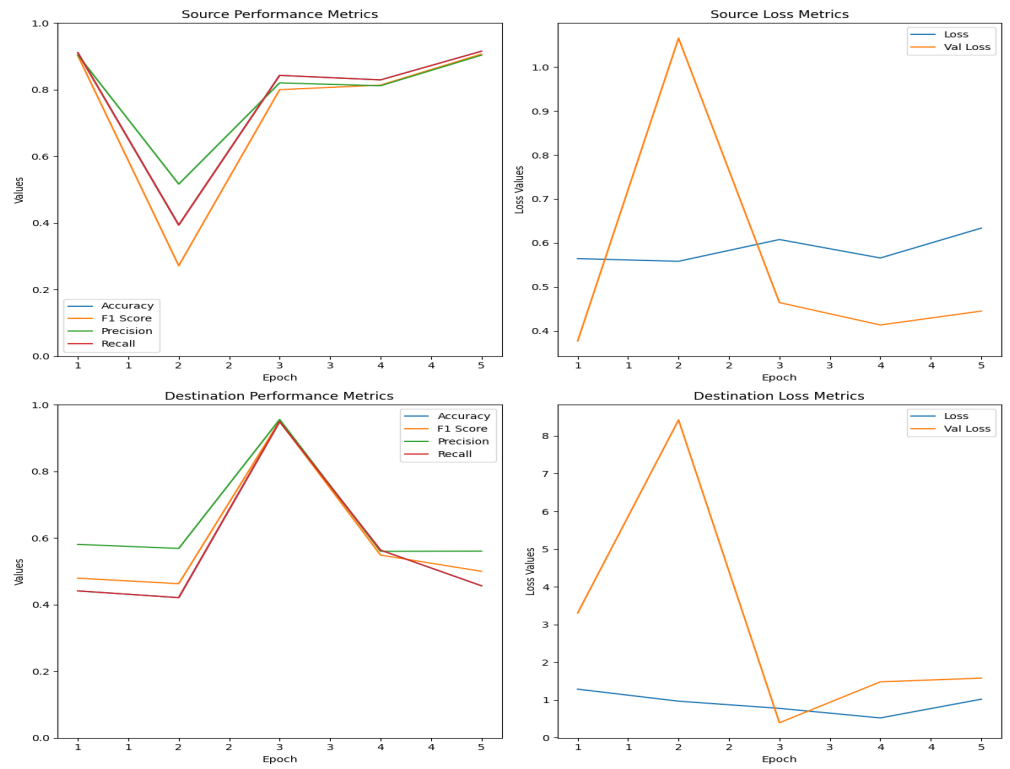


Figure 25. Cont.

(b) Features Name: features2, Hidden Feature Size: [8-8], Layer Type: GATJK, Learning Rate: 0.1, Epochs: 5



(c) Features Name: features1, Hidden Feature Size: [8-8], Layer Type: GATJK, Learning Rate: 0.001, Epochs: 5

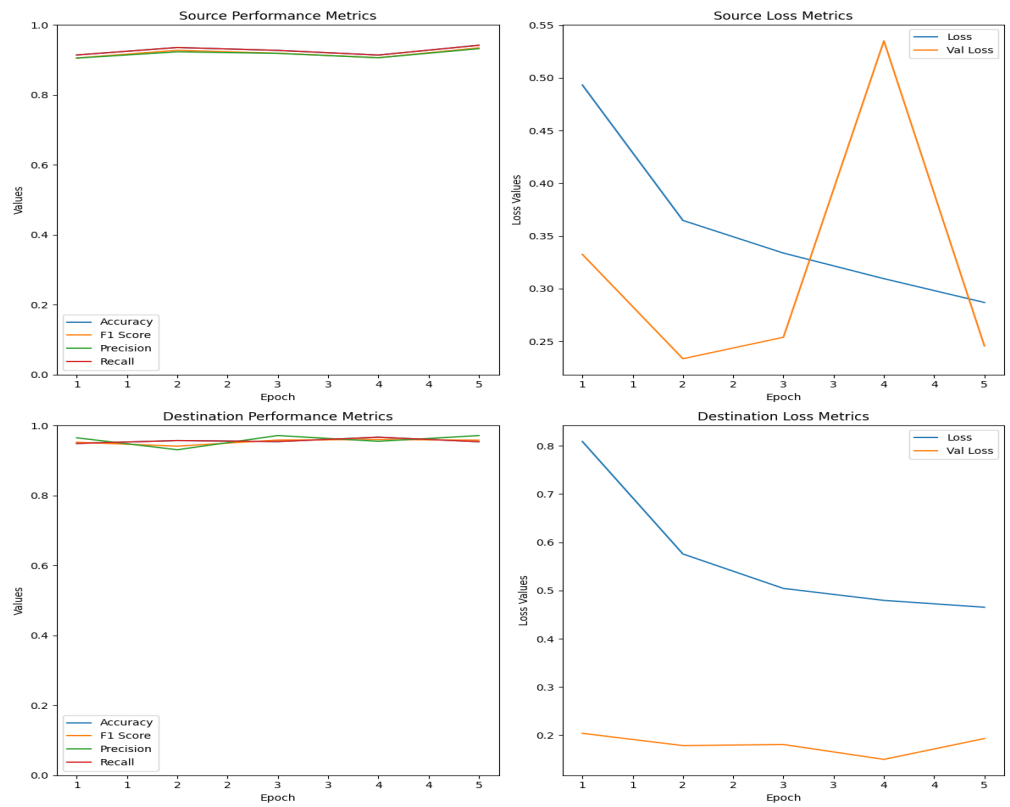
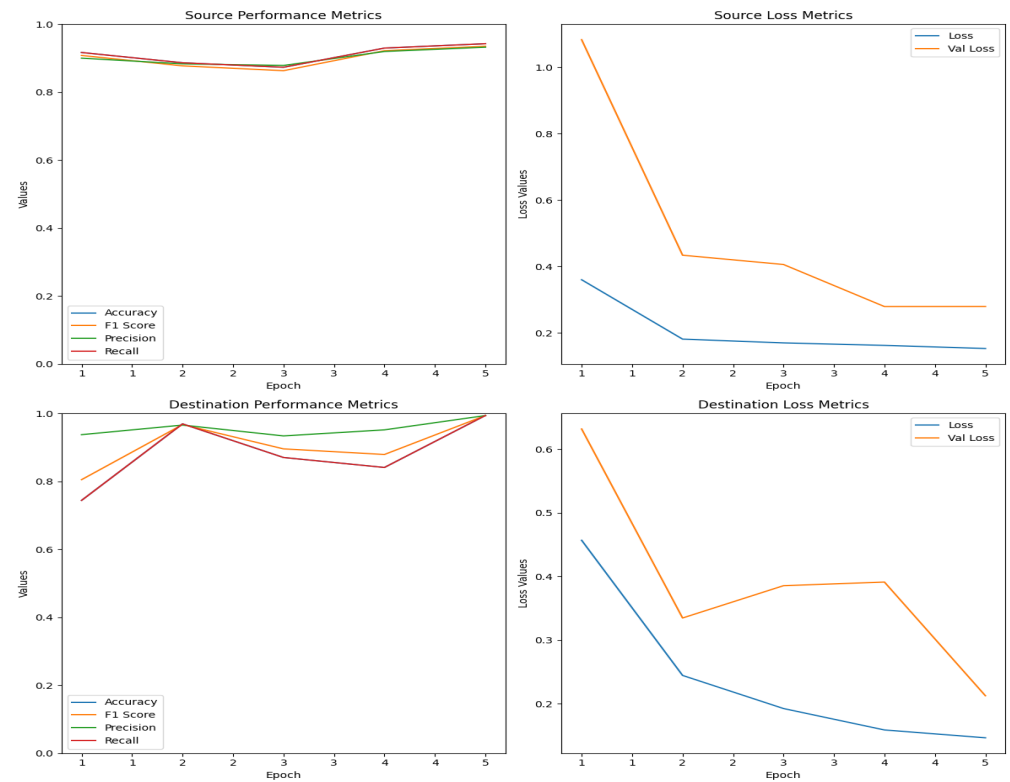


Figure 25. Cont.

(d)

Features Name: features2, Hidden Feature Size: [8-8], Layer Type: GATJK, Learning Rate: 0.001, Epochs: 5



(e)

Features Name: features1, Hidden Feature Size: [8-8], Layer Type: SAGE, Learning Rate: 0.0005, Epochs: 10

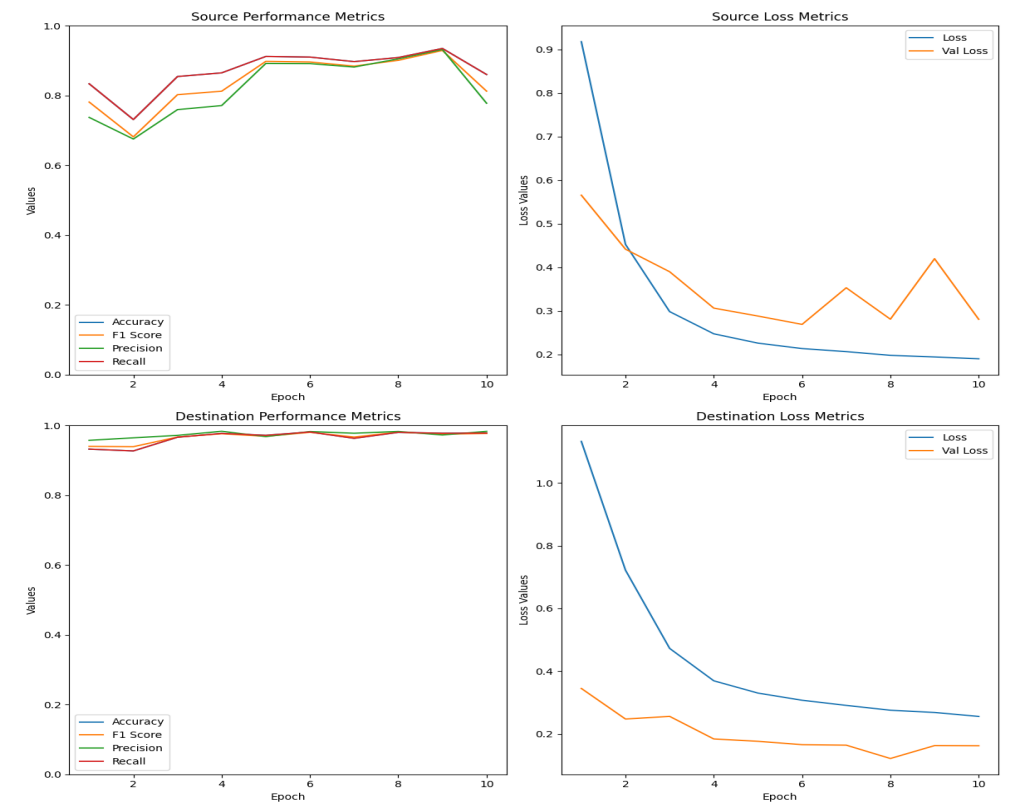
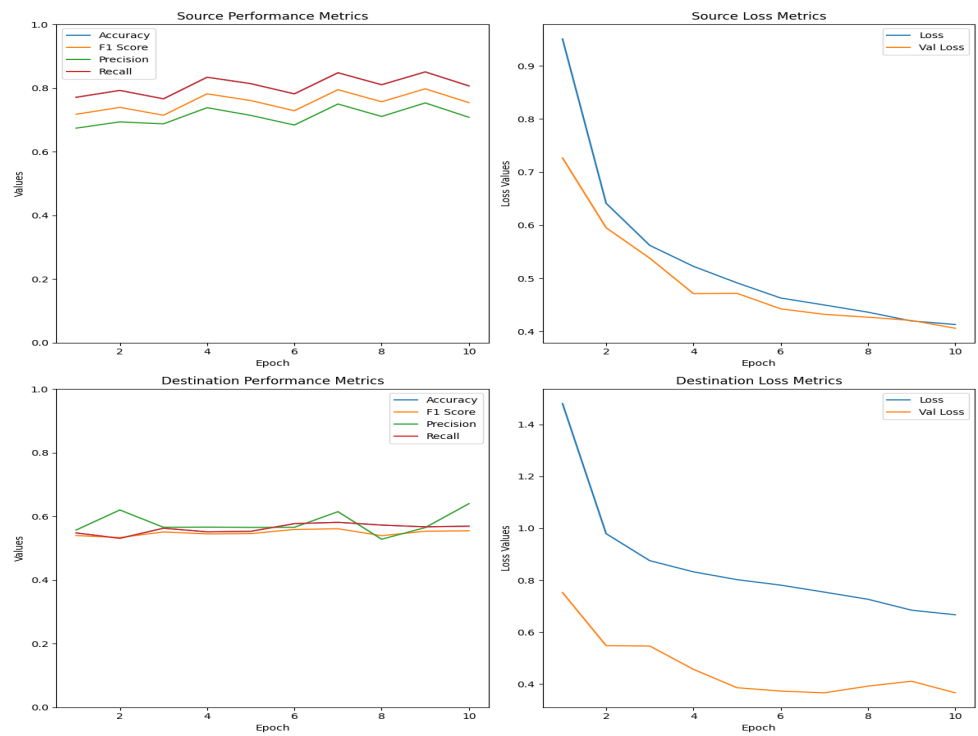


Figure 25. Cont.

(f) Features Name: features2, Hidden Feature Size: [8-8], Layer Type: SAGE, Learning Rate: 0.0005, Epochs: 10



(g) Features Name: features1, Hidden Feature Size: [8-8], Layer Type: GATJK, Learning Rate: 0.1, Epochs: 100

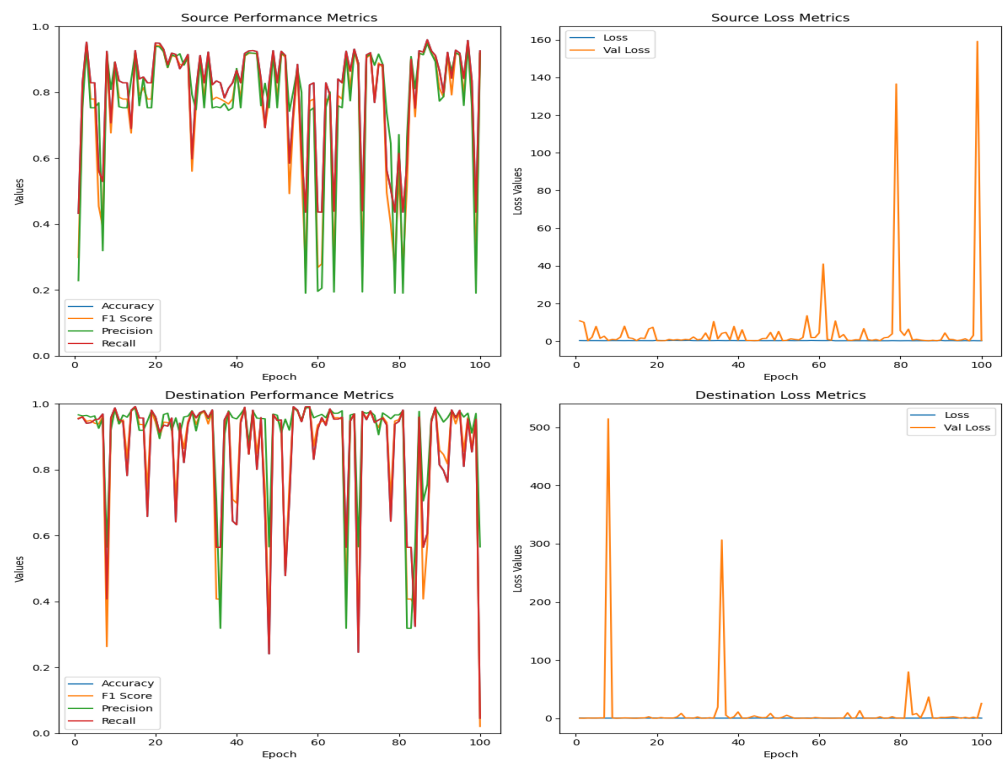
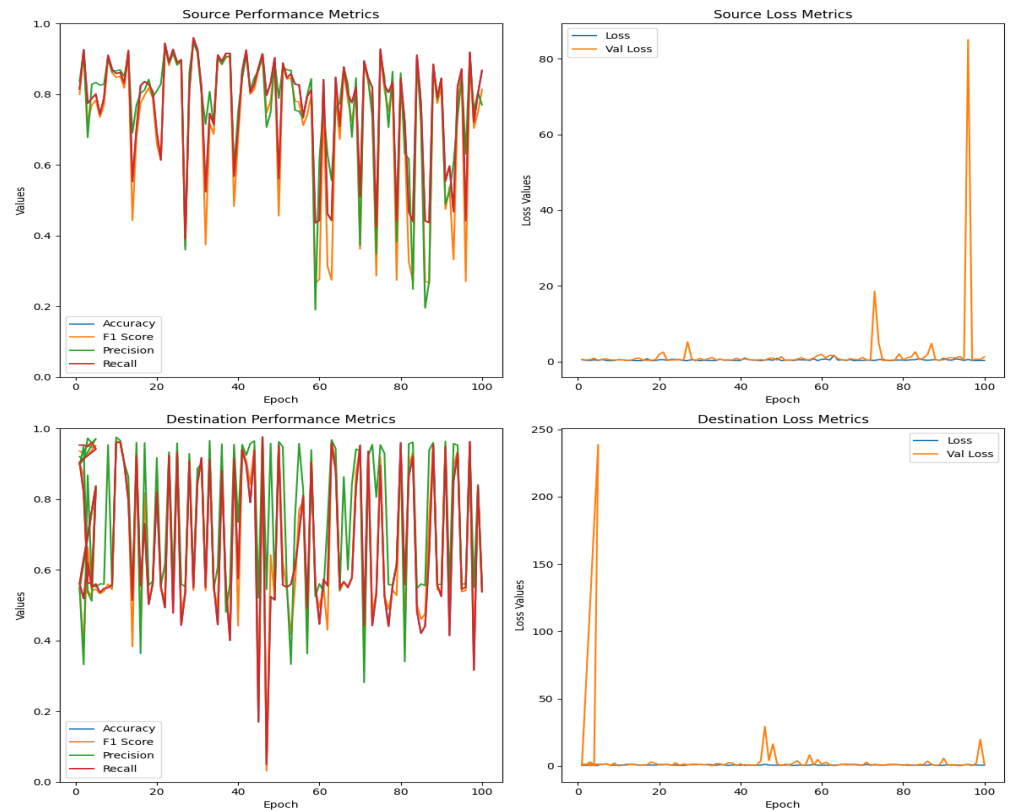


Figure 25. Cont.

(h)

Features Name: features2, Hidden Feature Size: [8-8], Layer Type: GATJK, Learning Rate: 0.1, Epochs: 100



(i)

Features Name: features1, Hidden Feature Size: [8-8], Layer Type: GATJK, Learning Rate: 0.001, Epochs: 100

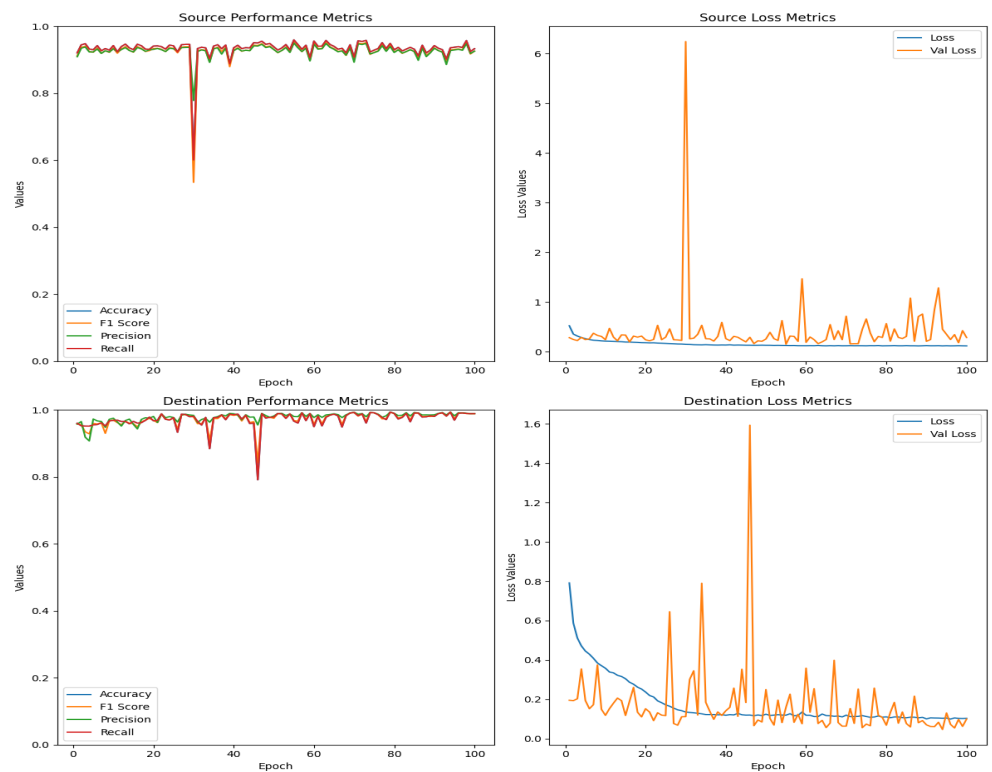
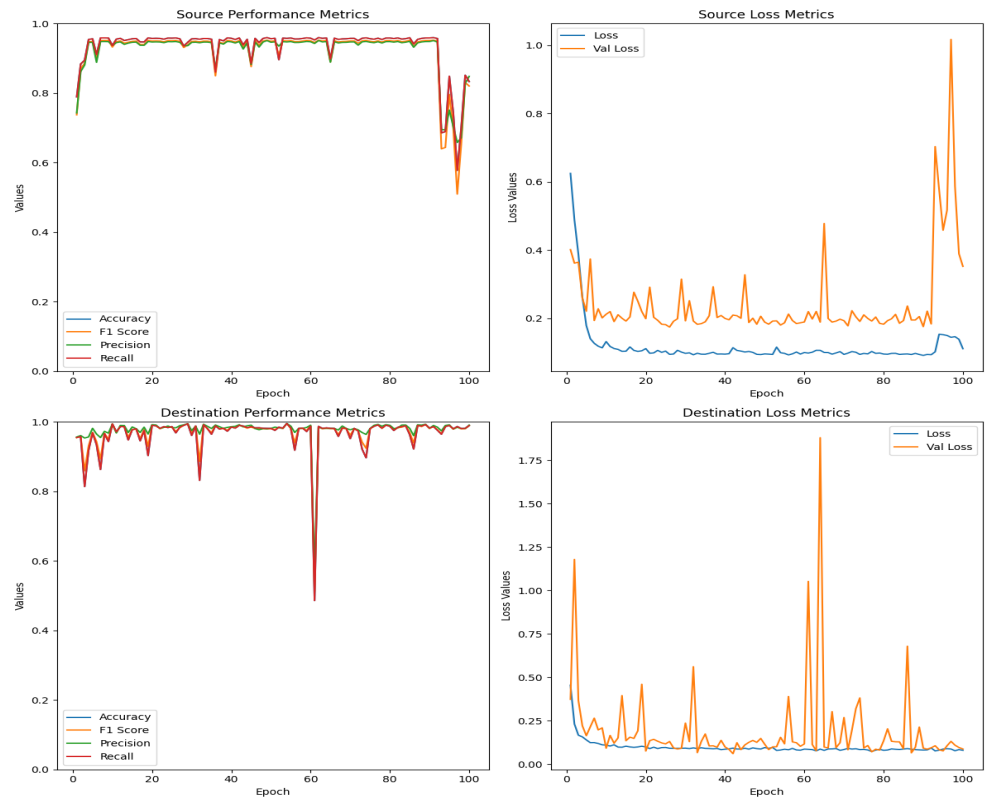


Figure 25. Cont.

(j)

Features Name: features2, Hidden Feature Size: [8-8], Layer Type: GATJK, Learning Rate: 0.001, Epochs: 100



(k)

Features Name: features1, Hidden Feature Size: [16-16], Layer Type: GATJK, Learning Rate: 0.1, Epochs: 100

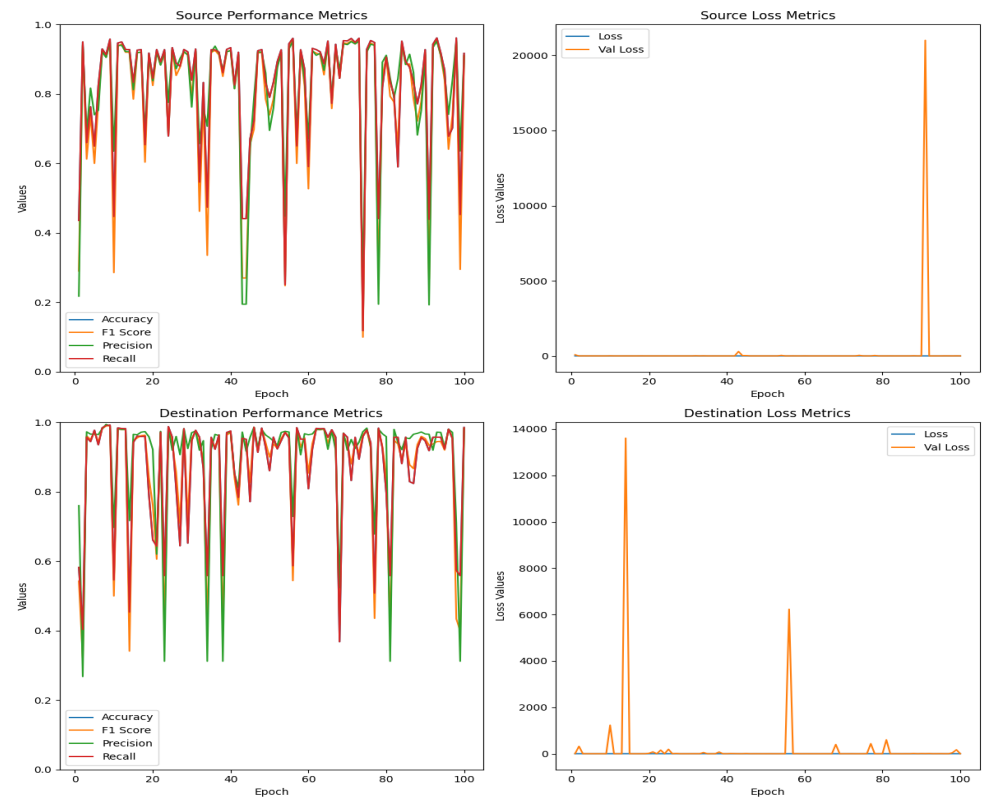
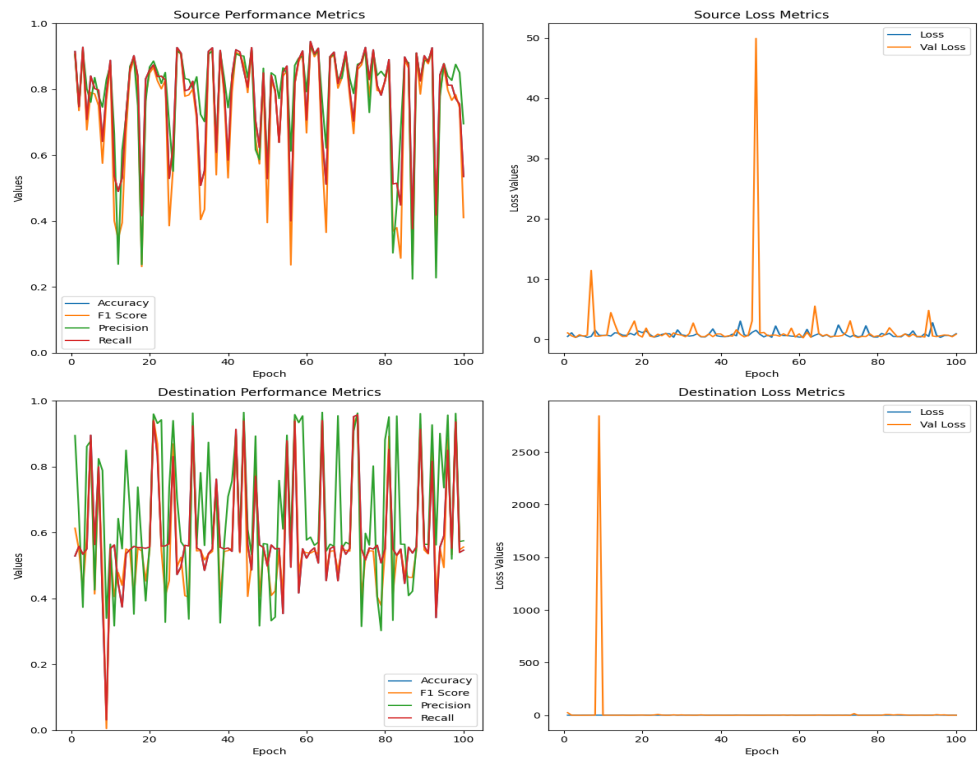


Figure 25. Cont.

(l)

Features Name: features2, Hidden Feature Size: [16-16], Layer Type: GATJK, Learning Rate: 0.1, Epochs: 100



(m)

Features Name: features1, Hidden Feature Size: [8-8], Layer Type: GATv2, Learning Rate: 0.0005, Epochs: 100

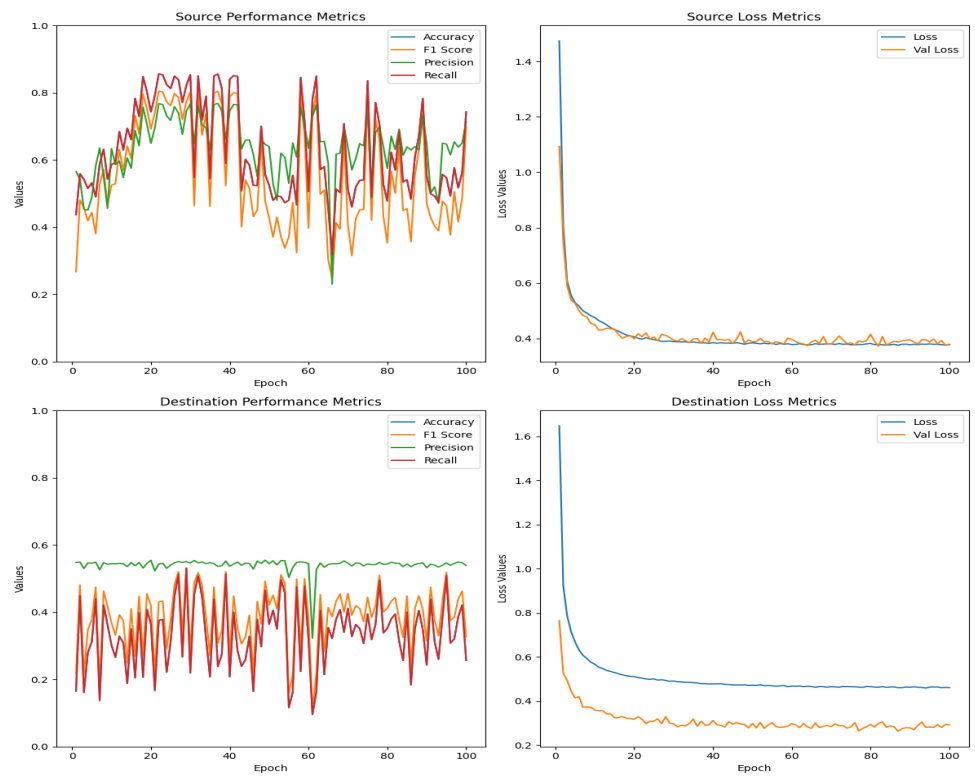
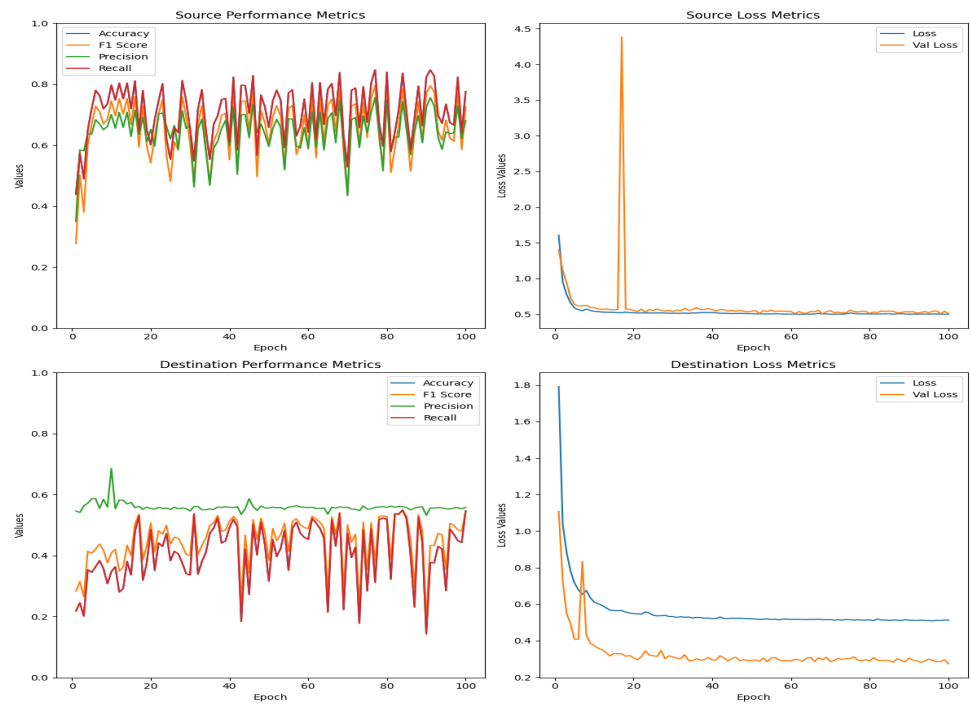


Figure 25. Cont.

(n)

Features Name: features2, Hidden Feature Size: [8-8], Layer Type: GATv2, Learning Rate: 0.0005, Epochs: 100



(o)

Features Name: features1, Hidden Feature Size: [8-8], Layer Type: GATv2, Learning Rate: 0.0005, Epochs: 10

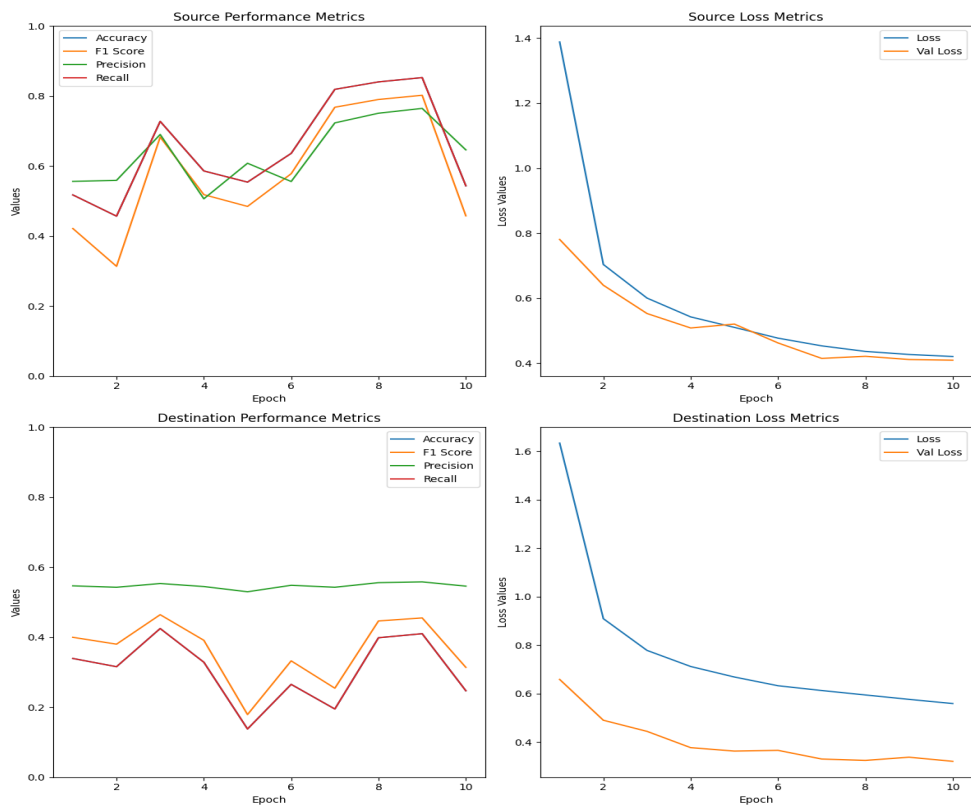
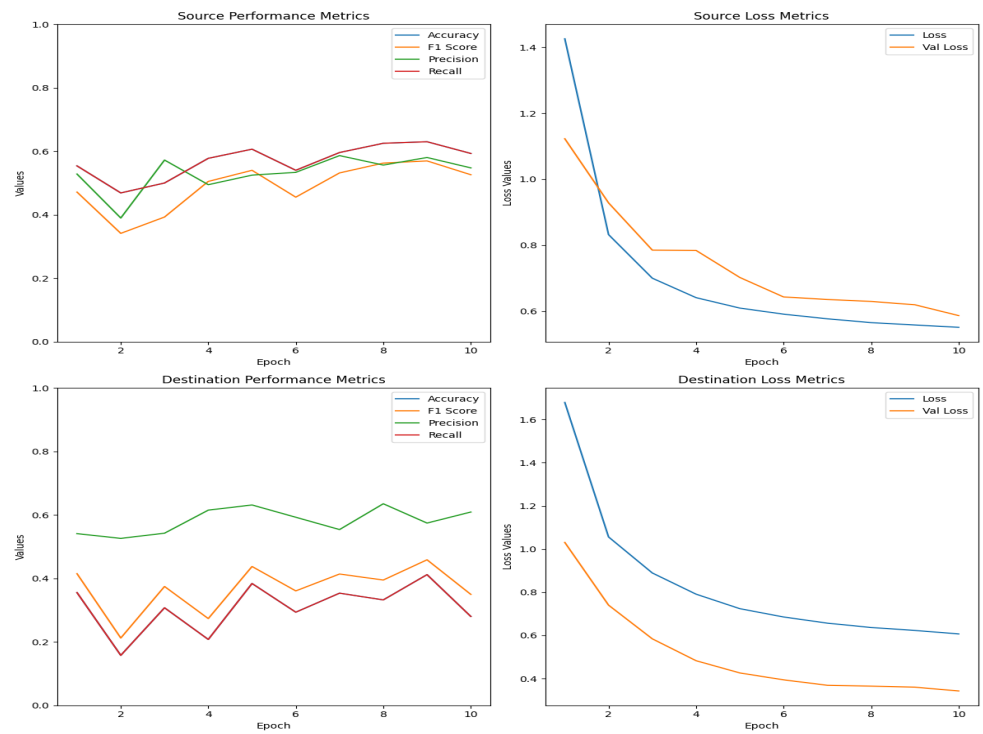


Figure 25. Cont.

(p)

Features Name: features2, Hidden Feature Size: [8-8], Layer Type: GATv2, Learning Rate: 0.0005, Epochs: 10



(q)

Features Name: features1, Hidden Feature Size: [8-8], Layer Type: GATv2, Learning Rate: 0.0005, Epochs: 5

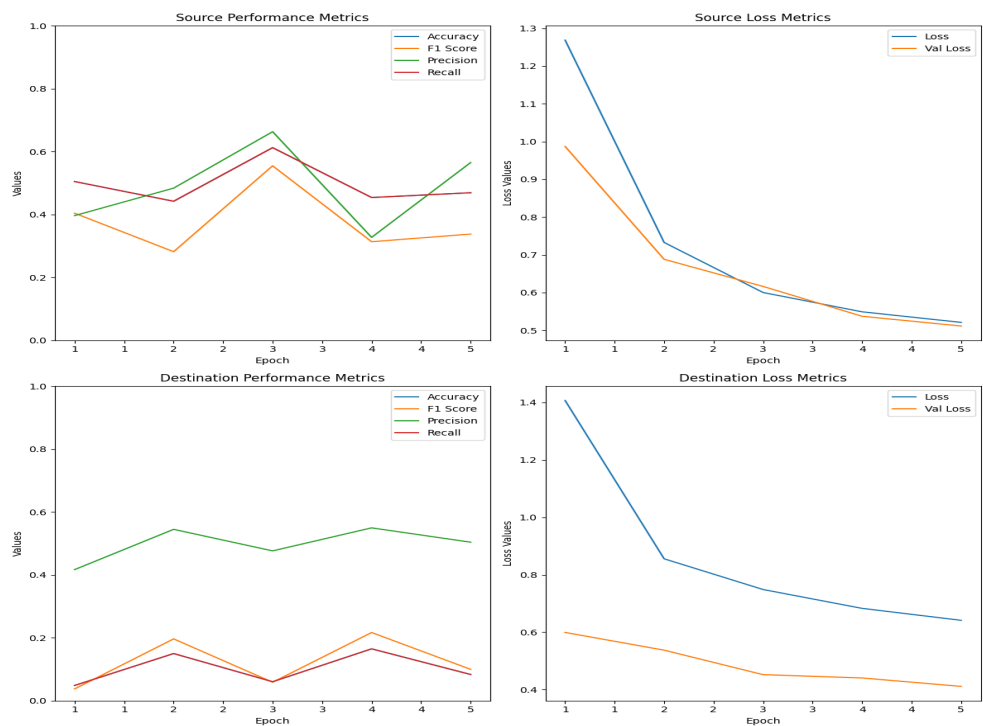


Figure 25. Cont.

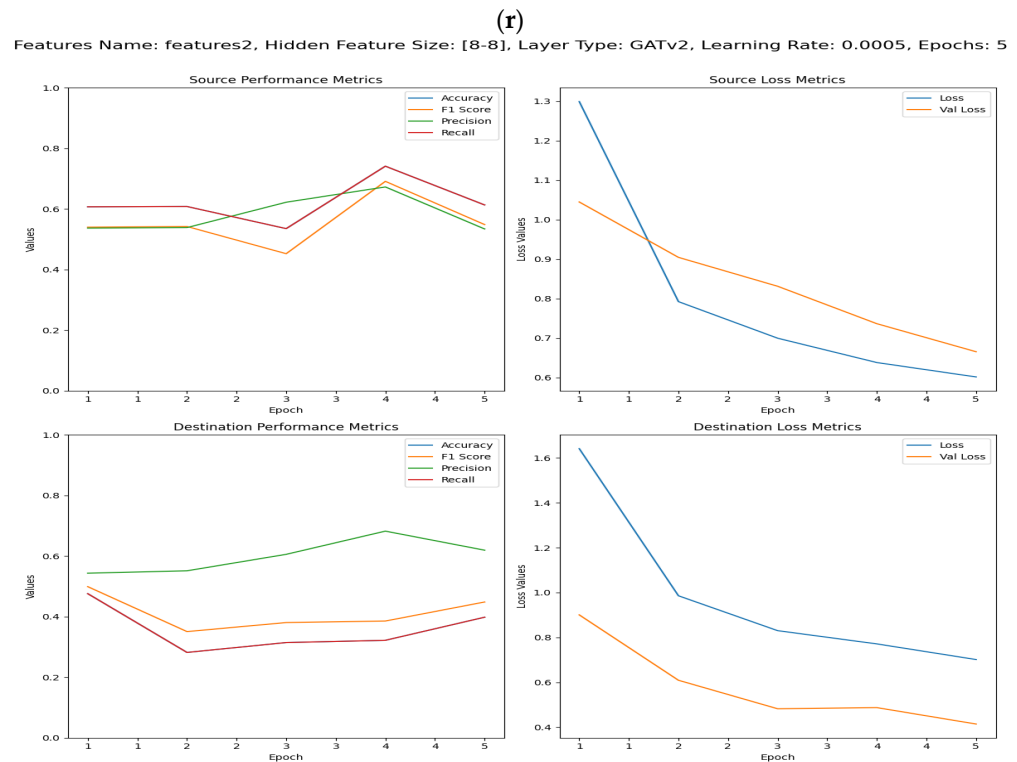


Figure 25. Node classification results: varying configurations of GNN models.

The inclusion of centrality measures (*katz_centrality*, *betweenness_centrality*) in *features2* compared to the basic nodal degrees and rank in *features1* showed a noticeable enhancement in the model's performance. Specifically, models using *features2* with SAGE layers and a learning rate of 0.0005 over 10 epochs (Figure 25f) exhibited an average F1 score improvement of 5%, a precision increase by 6%, and a recall rise by 4%, signaling a more accurate and consistent classification capability.

A hidden feature size of [8-8] struck an optimal balance between the model's complexity and generalizability. Configurations with this size displayed less variance in loss values and maintained the performance over a range of learning rates and epochs. In contrast, a size of [16-16] was associated with inconsistent loss patterns, particularly under the GATJK layer type. The SAGE layer type has proven to be more effective than GATJK and GATv2 when working with smaller hidden feature sizes, as demonstrated by the superior performance metrics in configurations utilizing SAGE (Figure 25f). Conversely, GATJK layers paired with larger hidden features ([16-16]) exhibited high variability in loss, with some models demonstrating training loss to validation loss ratios exceeding 20:1, indicative of potential overfitting. GATv2 layers, while showing improved stability over GATJK in some configurations, still fell short of the performance achieved by SAGE layers.

The models trained with a high learning rate of 0.1 were prone to an erratic performance, across all the tested configurations. This was evident in the performance metrics, where the loss values fluctuated by up to 200% from one epoch to the next. In contrast, a lower learning rate of 0.0005 consistently produced a smooth reduction in loss across successive epochs, suggesting a more stable and reliable training process.

The number of epochs played a critical role in the model training. While a higher number of epochs, such as 100, could potentially lead to better learning, it also increased the risk of overfitting, as shown by the variability in the loss metrics. A moderate number of epochs, specifically 10, was often enough to achieve a high performance without overfitting, as seen in the stable loss decline and high F1 score in the optimal configurations.

Considering all the factors examined, the most effective model configuration for the node classification was found to be features2 with a [8-8] hidden feature size, using the SAGE layer type, a learning rate of 0.0005, and over 10 epochs, as highlighted in Table 31.

This setup not only provided the highest performance metrics, with an F1 score plateauing at around 85%, but it also showed a consistent training behavior, with average loss decreases of about 0.03 per epoch. This configuration excels due to its ability to effectively capture and classify the node information without overfitting, ensuring the model's robustness and adaptability to new data, and making it the preferred choice for deployment in the cybersecurity node classification tasks.

9. Conclusions

In this study, we leveraged Memgraph, an open-source graph database, to perform a graph-based analysis of network data and applied Graph Neural Networks (GNNs) for classifying cybersecurity attack tactics as categorized by the MITRE ATT&CK framework. Our approach incorporated various graph characterization metrics such as PageRank, degree centrality, betweenness centrality, and Katz centrality to enhance the model's ability to capture the structural features of the graph. Using the UWF-ZeekDataFall22 dataset, we demonstrated that combining graph-based techniques with machine learning significantly improves the classification of network entities, identifying both the presence and nature of cyber threats.

The results from our node classification experiments indicated that certain model configurations, particularly those using SAGE layers with a lower learning rate and moderate epoch numbers, achieved high performance metrics. This underscores the potential of GNNs in the domain of cybersecurity for effective threat detection and network defense. Our study highlights the importance of integrating graph analytics and machine learning to address the complex cybersecurity challenges. By transforming the raw network data into structured graph representations and employing advanced classification techniques, we can gain deeper insights into the network threats, enabling proactive and informed security measures.

10. Future Works

While our research presents promising results, several areas demand further exploration. Future work can focus on the following aspects:

1. **Scalability and Performance Optimization:** As the volume of network data continues to grow, it is crucial to optimize the performance and scalability of our graph-based analysis framework. This can involve exploring more efficient algorithms for graph characterization and enhancing the computational efficiency of the GNN models.
2. **Advanced Feature Engineering:** Investigating additional graph-theoretic features can improve the accuracy and robustness of threat classification. This includes exploring the dynamic graph neural networks that can adapt to the changes in the network structure over time.
3. **sCross-domain Applications:** extending the application of our methodology to other domains such as financial fraud detection, social network analysis, and biological network analysis can demonstrate its versatility and effectiveness in different contexts.

By addressing these areas, future research can further advance the field of cybersecurity, providing more robust and adaptive defense mechanisms against the evolving cyber threats.

Author Contributions: Conceptualization, S.S.B.; methodology, S.C. and P.S.; software, S.C. and P.S.; validation, S.S.B., S.C., P.S., D.M. and S.C.B.; formal analysis, S.S.B., S.C., P.S., D.M. and S.C.B.; investigation, S.S.B., S.C., P.S., D.M. and S.C.B.; resources, S.S.B., D.M. and S.C.B.; data curation, D.M.; writing—original draft preparation, S.C. and P.S.; writing—review and editing, S.S.B., S.C., P.S., D.M. and S.C.B.; visualization, S.C. and P.S.; supervision, S.C.B., S.S.B. and D.M.; project administration,

S.C.B., S.S.B. and D.M.; funding acquisition, S.C.B., S.S.B. and D.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Centers of Academic Excellence in Cybersecurity, NCAE-C-002: Cyber Research Innovation Grant Program, Grant Number: H98230-21-1-0170. We would also like to thank the Askew Institute at University of West Florida for partially supporting this grant.

Data Availability Statement: The datasets are available at <https://datasets.uwf.edu/> (accessed on 20 August 2023).

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Datasets.uwf.edu. Available online: <https://datasets.uwf.edu/> (accessed on 17 April 2024).
2. Memgraph. Getting Started with Memgraph. Available online: <https://memgraph.com/docs/getting-started> (accessed on 6 January 2024).
3. Welcome to Neo4j. Available online: <https://neo4j.com/docs/getting-started/> (accessed on 6 January 2024).
4. Neo4j vs. Memgraph—How to Choose a Graph Database? Available online: <https://memgraph.com/blog/neo4j-vs-memgraph> (accessed on 6 January 2024).
5. Javorník, M.; Husák, M. Mission-centric decision support in cybersecurity via Bayesian Privilege Attack Graph. *Eng. Rep.* **2022**, *4*, e12538. [[CrossRef](#)]
6. Jacob, S.; Qiao, Y.; Ye, Y.; Lee, B. Anomalous distributed traffic: Detecting cyber security attacks amongst microservices using graph convolutional networks. *Comput. Secur.* **2022**, *118*, 102728. [[CrossRef](#)]
7. Zola, F.; Seguro-la-Gil, L.; Bruse, J.L.; Galar, M.; Orduna-Urrutia, R. Network traffic analysis through node behaviour classification: A graph-based approach with temporal dissection and data-level preprocessing. *Comput. Secur.* **2022**, *115*, 102632. [[CrossRef](#)]
8. Alqahtani, H.; Sarker, I.H.; Kalim, A.; Hossain, S.M.M.; Ikhtlaq, S.; Hossain, S. Cyber intrusion detection using machine learning classification techniques. In *Computing Science, Communication and Security: First International Conference, COMS2 2020, Gujarat, India, 26–27 March 2020*; Revised Selected Papers 1; Springer: Berlin/Heidelberg, Germany, 2020; pp. 121–131.
9. Ding, Z.; Cao, D.; Liu, L.; Yu, D.; Ma, H.; Wang, F. A Method for Discovering Hidden Patterns of Cybersecurity Knowledge Based on Hierarchical Clustering. In *Proceedings of the 2021 IEEE Sixth International Conference on Data Science in Cyberspace (DSC)*, Shenzhen, China, 9–11 October 2021; pp. 334–338. [[CrossRef](#)]
10. Kumari, R.; Sheetanshu; Singh, M.K.; Jha, R.; Singh, N.K. Anomaly detection in network traffic using K-mean clustering. In *Proceedings of the 2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, Dhanbad, India, 3–5 March 2016; pp. 387–393. [[CrossRef](#)]
11. Sagdatullin, A. Cybersecurity System with State Observer and K-Means Clustering Machine Learning Model. In *Distributed Computer and Communication Networks*; Vishnevskiy, V.M., Samouylov, K.E., Kozyrev, D.V., Eds.; Springer Nature: Cham, Switzerland, 2023; pp. 183–195.
12. What Is the MITRE ATT&CK Framework? Available online: <https://www.trellix.com/security-awareness/cybersecurity/what-is-mitre-attack-framework/> (accessed on 17 April 2024).
13. Bagui, S.S.; Mink, D.; Bagui, S.C.; Ghosh, T.; Plenkers, R.; McElroy, T.; Dulaney, S.; Shabanali, S. Introducing UWF-ZeekData22: A Comprehensive Network Traffic Dataset Based on the MITRE ATT&CK Framework. *Data* **2023**, *8*, 18. [[CrossRef](#)]
14. Kanakaris, N.; Michail, D.; Varlamis, I. A Comparative Survey of Graph Databases and Software for Social Network Analytics: The Link Prediction Perspective. In *Graph Databases*; CRC Press: Boca Raton, FL, USA, 2023; pp. 36–55.
15. Gleich, D.F. PageRank Beyond the Web. *SIAM Rev.* **2015**, *57*, 321–363. [[CrossRef](#)]
16. Memgraph. Pagerank. Available online: <https://memgraph.com/docs/advanced-algorithms/available-algorithms/pagerank> (accessed on 15 January 2024).
17. Memgraph. Degree_Centrality. Available online: <https://memgraph.com/docs/advanced-algorithms/available-algorithms/degree Centrality> (accessed on 22 January 2024).
18. Golbeck, J. *Introduction to Social Media Investigation*; Elsevier: Amsterdam, The Netherlands, 2015.
19. Xiang, N.; Wang, Q.; You, M. Estimation and update of betweenness centrality with progressive algorithm and shortest paths approximation. *Sci. Rep.* **2023**, *13*, 17110. [[CrossRef](#)] [[PubMed](#)]

20. Memgraph. Betweenness_Centrality. Available online: <https://memgraph.com/docs/advanced-algorithms/available-algorithms/betweenness centrality> (accessed on 29 January 2024).
21. van der, A.G.; Bergamini, E.; Green, O.; Bader, D.A.; Meyer-henke, H. Scalable Katz Ranking Computation in Large Static and Dynamic Graphs. *ACM J. Exp. Algorithmics (JEA)* **2022**, *27*, 1–16. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.