

GRAPH-BASED NETWORK ANOMALY DETECTION

BY

Ahmed Mohammed El-Nazir

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

SECURITY & INFORMATION ASSURANCE

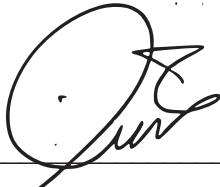
May, 2020

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

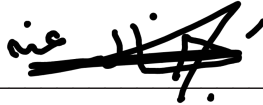
DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Ahmed El-Nazir** under the direction his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**.



Dr. HAMOUD ALJAMAAN
Department Chairman



Dr. Salam A. Zummo
Dean of Graduate Studies



Dr. Emad Ramadan
(Advisor)



Dr. Sami Zhioua
(Member)



Dr. Jameleddine Hassine
(Member)

7/5/2020

Date



© Ahmed El-Nazir

2020

Dedication

ACKNOWLEDGMENTS

First of all, I would like to express my sincere appreciation to my advisor Dr. Emad Ramdan for the constant support of my Master's study and thesis, for his patience, inspiration, interest, and sharing with me his valuable knowledge, his supervision in all the time of research and writing of this thesis.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Sami Zhioua, and Dr. Jameleddine Hassine, for their encouragement, insightful comments, and valuable assessment.

My sincere thanks also go to Prof. Syed Ahmed M. Said and Dr. Farag Azzedin, for all the respected advice and motivational support.

I thank my friends and my relatives for believing in me especially Eng. Hiba Ahmed for all the support and time that she provide to me. My family: my parents Mohammed El-Nazir and Saniya Saeed, for supporting me emotionally throughout my life. My sister and my two brothers Moez and Osman.

Last but not the least, I would like to give a special and emotional thanks to my wife Mayasir Al-Fadel for being the most supported person in my career and my life.

Table of Contents

ACKNOWLEDGMENTS.....	V
LIST OF TABLES	X
LIST OF FIGURES	XI
LIST OF ABBREVIATIONS.....	XII
ABSTRACT	XIII
ملخص الرسالة	XIV
CHAPTER 1 INTRODUCTION	1
1.1 Problem statement	3
1.2 Thesis Objectives	3
1.3 Methodology	4
1.4 Main Contributions	5
1.5 Thesis Organization.....	5
CHAPTER 2 BACKGROUND	6
2.1 Anomaly Detection	6
2.2 Network Attacks	6
2.3 Graphs	7
2.4 Graph Database	8
2.5 Graph metrics	9
2.6 Centrality Measures and Node Ranking	11
2.6.1 Degree Centrality	11
2.6.2 In-Degree Centrality	12

2.6.3	Out-Degree Centrality	12
2.6.4	Closeness Centrality	12
2.6.5	Harmonic Closeness Centrality	13
2.6.6	Eccentricity.....	13
2.6.7	Betweenness Centrality.....	13
2.6.8	Modularity Class.....	14
2.7	SMOTE: Synthetic Minority Over-sampling TEchnique	15
2.8	Feature Selection (Boruta)	15
2.9	Classifiers.....	16
2.9.1	Support Vector Machine (SVM)	16
2.9.2	Random Forest	16
2.9.3	Naïve Bayes	17
2.10	The Complexity of Graph-Based Analysis.....	17
CHAPTER 3 LITERATURE REVIEW		18
3.1	Anomaly Detection	18
3.2	Graph-Based Analysis	19
3.2.1	Node Analysis.....	19
3.2.2	Edge and Community Analysis.....	22
3.2.3	Graph Analysis Tools	24
3.3	Research Gap	26
CHAPTER 4 PROPOSED APPROACH.....		27
4.1	Datasets.....	29
4.1.1	Network Infrastructures	30
4.1.2	Network Flow	30

4.1.3	Generated Labelled Flows Datasets.....	31
4.1.4	Graph-Based Datasets	32
4.2	Graph Models	33
4.3	Build the Graph.....	33
4.4	Compute Graph Metrics.....	34
4.5	Analyzing Graph-Metrics.....	34
4.5.1	Baseline Measures.....	35
4.5.2	Diameter-Based Measures	36
4.5.3	Flow-Based Measures.....	37
4.5.4	Community Analysis	38
CHAPTER 5 EXPERIMENT & RESULTS		39
5.1	Benign Data	40
5.2	DDoS.....	43
5.2.1	Graph Model	43
5.2.2	Outcomes	44
5.3	PortScan.....	49
5.3.1	Graph Model	49
5.3.2	Outcomes	50
5.4	Botnet.....	53
5.4.1	Graph Model	53
5.4.2	Outcomes	55
5.5	Web Attack	57
5.5.1	Graph Model	57
5.5.2	Outcomes	58
5.6	FTP-SSH Patator	61

5.6.1	Graph Model	61
5.6.2	Outcomes	61
5.7	Infiltration – Dropbox download	62
CHAPTER 6 EVALUATION AND DISCUSSION		64
6.1	Evaluation	64
6.1.1	Preparing the Dataset	65
6.1.2	Effectiveness	66
6.1.3	Performance	66
6.2	Outcome Verification	68
6.3	Comparison with Existing NIDS (SNORT)	71
6.3.1	Snort	71
6.3.2	Prepare the Datasets	72
6.3.3	Performance	73
6.4	Discussion	73
6.4.1	Threat Identification	74
6.4.2	Performance & Scalability	75
6.4.3	Unknown attacks	76
6.4.4	Monitoring	77
6.5	Comparison with Existing Graph-Based Intrusion Detection Method	77
CHAPTER 7 CONCLUSIONS		79
REFERENCES		83
VITAE		89

LIST OF TABLES

Table 1: Samples of network flow data used to build a SOCKET_Based Graph.....	31
Table 2: Samples of the Graph_Based metrics from the GB dataset.....	32
Table 3: Statistics of Graph-Based Datasets.....	39
Table 4: <i>Degree & Modularity</i> Metrics for the top 9 nodes based on <i>Degree</i> value.....	41
Table 5: <i>Centrality</i> Metrics for the top 9 nodes based on <i>Degree</i> value	43
Table 6: <i>Degree's & Modularity</i> class metrics of the top 9 nodes after DDoS	45
Table 7: <i>Centrality</i> metrics of the top 9 nodes based on <i>Degree</i> values after DDoS occurred ...	47
Table 8: Samples of participant nodes in the DDoS attack	49
Table 9: Edges weight of Benign flow	51
Table 10: Edges weight of PortScan flow.....	52
Table 11: Degree & Modularity Metrics after the occurrence of BotNet attack	54
Table 12: Centrality Metrics for affected nodes after the occurrence of BotNet attack.....	57
Table 13: <i>Centralities & Modularity</i> Metrics after the occurrence of WebAttack attack	60
Table 14: Samples of affected nodes after the occurrence of WebAttack attack	60
Table 15: Samples of affected nodes after the occurrence of FTP attack.....	62
Table 16: The affected nodes after the occurrence of Infiltration attack.....	63
Table 17: AVERAGE CLASSIFICATION TIMES IN SECONDS.	67
Table 18: Graph-based Outcomes Verification, for different Graph-based tools.....	70
Table 19: Performance comparison based on the processing time	73
Table 20: The criteria of the conducted comparison	74
Table 21: Graph-Based the result of the comparison with other Graph-Base Method.....	78

LIST OF FIGURES

Figure 1: Our proposed Graph-Based approach	3
Figure 2: IP_Based Model	8
Figure 3: SOCKET_Based Model	8
Figure 4: Edge Directions, (a) shows the edge as an arrow represent the flow direction, where (b) represent an undirected edge	10
Figure 5: Graph-Based Components.....	28
Figure 6: Testbed Architecture	29
Figure 7: Degree Centralities, shows how Degree values are measures.....	35
Figure 8: Diameter & Flow Centralities, shows how Diameter and Flow centralities are measures.	36
Figure 9: Benign Graph, shows how the benign traffic transformed into a graph.....	42
Figure 10: DDoS Graph, shows how the DDoS affect the graph, where the red arrow point to the nodes that participate in the attack.....	46
Figure 11: DDoS <i>Modularity</i> Graph I, shows the graph colored based on <i>Modularity</i> class	48
Figure 12: DDoS <i>Modularity</i> Graph II, shows how all nodes that participate in DDoS attack within the same, at the right site there is Close look of DDoS Community	48
Figure 13: Shows the graph upon portscan attack	50
Figure 14: PortScan Graph, shows how the weight of the RED edge that targeting the victim machine's can be visible within the graph.....	53
Figure 15: Botnet Graph, shows Botnet affect over the graph.....	55
Figure 16: Botnet Graph, Shows the impact of CNC server traffic over the graph.....	56
Figure 17: Web Attack Graph, shows graph components after web attack occurred.....	59
Figure 18: Web Attack Graph, shows how all nodes that participate in Webattack within the same Community, at the right site a close look to WebAttack Community.....	59
Figure 19: Evaluation of GLF and GB datasets, Average accuracy (%) per validation method for each dataset. Random Forest, Naive Based, SVM.....	65

LIST OF ABBREVIATIONS

IDS:	Intrusion Detection System
CAD:	Contextual Anomaly Detection
WAD:	Watchmen Anomaly Detection
FP:	False positive
IP:	Internet Protocol
MDL:	Minimum Description Length
DNS:	Domain Name Server
AD:	Active Directory
SMOTE:	Synthetic Minority Over-sampling TEchnique
GLF:	Generated Labelled Flows
GB:	Graph-Based
SVM:	Support Vector Machine
RF:	Random Forest
NB:	Naive Bayes
PC:	Personal Computer
DDoS:	Distributed Denial of Service
SQL:	Structured Query Language
XSS:	Cross-site scripting
FTP:	File Transfer Protocol
CNC:	Command and Control server
NIDS:	Network Intrusion Detection System

ABSTRACT

Full Name : Ahmed Mohammed El-Nazir
Thesis Title : Graph-Based Network Anomaly Detection
Major Field : Security & Information Assurance
Date of Degree : MAY/2020

The ability to analyze data formed as a graph has become essential in many areas for detecting unusual patterns. One of the most critical areas in detecting cyber-attacks is anomaly detection, yet there is no much work that has been performed for discovering anomalies in Graph-Based data. In this thesis, we implemented a Graph-Based approach to reveal anomalies within network flow by observing deviations from normal flow using graph metrics. We utilize eight graph metrics to measure the behavior of each node based on its position and type of connections. Each network flow is transformed into either IP-Based or SOCKET-Based graph model according to the impact of the targeted attack within the flow. We evaluate the effectiveness and performance of our algorithm for each anomaly type by using a benchmark CICIDS2017 dataset. Also, we conducted comparisons between our proposed approach and two existing Anomaly detection methods, first, with Network Anomaly Detection System (SNORT), and against a Graph-Based Anomaly Detection method. Our Proposed approach demonstrated the efficiency of exploring a graph-based structure of network flow to detect network cyber-attacks.

ملخص الرسالة

الاسم الكامل: أحمد محمد صالح النذير

عنوان الرسالة: كشف شذوذ الشبكة باستخدام الهيكلية البيانية

التخصص: السرية وأمن المعلومات

تاريخ الدرجة العلمية: مارس، ٢٠٢٠م

القدرة على تحليل البيانات التي تمت هيكلتها كرسوم بياني، أصبحت ضرورية في العديد من المجالات للكشف عن الأنماط غير العادية. يعد الكشف عن الشذوذ من أهم المجالات في الكشف عن الهجمات السيبرانية، وومع ذلك لا يوجد العديد من الأعمال التي تمت للقيام باكتشاف الشذوذ عن طريق تحليل البيانات المهيكلية بطريقة الرسم البياني. في هذه الأطروحة ، قمنا بتنفيذ نهج جديد قائم على الرسم البياني للكشف عن الشذوذ الموجود داخل بيانات الشبكة من خلال ملاحظة الانحرافات عن التدفق الطبيعي باستخدام مقاييس الرسم البياني. ونحن هنا نقوم باستخدام ثمانية مقاييس بيانية لقياس سلوك كل عقدة بناءً على موقعها ونوع اتصالاتها. يتم تحويل كل بيانات الشبكة إلى نموذج بياني مستند إلى اما البناء عن طريق عناوين العقد أو عن طريق دمج العناوين بالمنافذ، وذلك وفقاً لتأثير الهجوم المستهدف داخل التدفق. وكما قمنا بتقييم فعالية وأداء الخوارزمية الخاصة بنا لكل نوع شاذ باستخدام مجموعة بيانات CICIDS2017 المعيارية. أيضاً ، أجرينا مقارنات بين نهجنا المقترح وطريقتين موجودتين للكشف عن الشذوذ ، أولاً ، مع نظام الكشف عن الشذوذ الشبكي (SNORT) ، وثانياً، طريقة مختلفة للكشف عن الشذوذ القائمة على الرسم البياني. وقد أظهر نهجنا المقترح كفاءة في استكشاف الهجمات الالكترونية على الشبكة من خلال التحقق من سلوكها في الهيكلية البيانية.

CHAPTER 1

INTRODUCTION

Detecting anomalies within network flow or system log files while considering the vast amount of user events or network traffic can be a hard task to accomplish. For instance, one of the most critical aspects of centralized information technology environments is to ensure system security. Having a reliable user authentication system is not an easy goal to achieve. Several methods need to be applied to accomplish that level of identity verification. For instance, username and password, second-factor authentication, biometrics, and identity cards. One of the most widely used network authentication protocol is Kerberos [1], developed by Microsoft. For such a system the massive amount of data gathered from the authentication events will be saved in log files and analyzed to understand users' behaviors. After that, we could apply anomaly detection methods over this save events to detect an abnormality, spot vulnerabilities, and design best protection techniques to achieve centralized environment security.

The massive size of data generated by user events overwhelms the traditional data storage and negatively affects its performance. One of the solutions is to use a new database model called NoSQL databases [14], which has been successfully utilized by Facebook, Google, and many Big-data systems. One of the NoSQL implementations depends on the

Graph-Based database to manage and access the massive flow of relational data without a noticeable impact on the performance [46][47].

Cyber-attacks can be detected by comparing events with a predefined list of attacks (Signature Based) or detecting the deviation from normal behaviors (Anomaly Based) [9]. The approach of the Signature-Based Intrusion Detection System is one of the most common techniques that implement a Signature-Based approach in detecting known threats. However, they don't scale well due to the wide variety of existing and new attacks [13]. By contrast, Anomaly-Detection depends on the ability to discriminate normal from abnormal behavior, to overcome the need for storing all predefined signatures. This will help in detecting threats by observing the unusual patterns of cyber-attacks.

Implementing anomaly detection to analyze data that represented in a graph form (Nodes/Edges), can significantly enhance both the performance and efficacy of the detection process. Graph Database is one of the graph implementations that use to overcome performance degradation when dealing with huge relational databases[48]. It aims to restructure the traditional way of data storage in a much effective way, by enhancing the search process, gaining more adaptability, and decreasing the response time. Therefore, dealing with a massive amount of network flow that formed in a Graph-Based structure can improve both the speed and scalability of IDS. Our graphs represent only the essential features that each connection has. Those features are src_IP, dst_IP, src_PORT, and dst_port, which are used by most IDS and firewall systems to detect threats. Figure 1 shows the main approach phases.

1.1 Problem statement

Considering the previous facts, there is a real need to find a new approach that can efficiently detect attacks behaviors with respect to both accuracy and performance. Many of the existence anomaly-based IDS suffer from performance degradation, false alarm rates, and limitation in handling large expandable data such as network flows or authentication log files. Therefore, in this work, we will explore the problem of detecting network attacks behaviors considering only graph-metrics for both nodes and edges.



Figure 1: Our proposed Graph-Based approach

1.2 Thesis Objectives

The goal of this thesis is to utilize the Graph-Based analysis to detect abnormality in network flow. This can be achieved through observing deviations in graph metrics that disclose the behaviors of network attacks. The proposed method leads toward faster and more efficient abnormality detection. In this thesis, we aim to achieve the following objectives:

- 1- Store network traffic in a Graph-Based database structure.
- 2- Calculate Graph-Based measurements and study their characteristics.
- 3- Analyze Graph-Based measurements for both benign and malicious traffics.

- 4- Evaluate the results by comparing the detection using Graph-Based metrics (GB datasets) against the detection of network flow features (GLF dataset).
- 5- Evaluate the performance by comparing the proposed approach with existing Network IDS (SNORT).
- 6- Evaluate the proposed approach with an existing Graph-Based method

1.3 Methodology

In this thesis, we are going to focus on detecting network attacks, by observing its deviation in term of graph-metrics, which affected by its behaviors and activity that these attacks need to perform in order to be applied, to implement our approach we need to perform the following:

- Built the graph from extracted edge-like values of network traffic, using graph-analyzes tools.
- We Computed centralities and modularity metrics for all nodes and edge_weight for all edges within the graph.
- We Analyzed graph metrics to study its changes over both benign and malicious traffic.
- Data mining has been performed over a dataset created from those metrics and existence labeled datasets with network features metrics.
- We Conducted a comparison with an existence Network Anomaly Detection System (Snort) to measure the performance.
- Finally, another comparison has been conducted with an existence Graph-Based analysis method.

1.4 Main Contributions

By conducting this thesis the main contributions that we are trying to achieve are the followings:

- Detect network attacks activities, by monitoring the deviation in graph metrics, by study and observe a unique pattern for network attacks.
- Verify the quality of the result in both accuracy and performance.
- Prove the efficiency of our approach against existing anomaly detection methods.

1.5 Thesis Organization

The rest of the thesis organized as follows. Chapter 2, will go through some background and definitions that related to the conducted research. Chapter 3 shows a review of the previous related work to illustrate both anomaly detection and Graph-Based analyses approach. Chapter 4 presents the proposed approach in detail. Chapters 5 focused on experiment implementation and results. Subsequently, Chapter 6 presents the evaluation of the proposed approach using three different machine learning classifiers and the discussion. Finally, Chapter 7 concludes the study

CHAPTER 2

BACKGROUND

2.1 Anomaly Detection

A zero-day attack is an unknown vulnerability exploited by an attacker before being discovered by the systems developers. Zero-day attacks can create a serious impact on system security; due to the lack of available information, discovering these types of attacks is not an easy task to achieve. Moreover, the identification of those attacks by Signature-Based detection methods is unlikely. Thus, utilizing Anomaly Detection is deemed more suitable for detecting those attacks [13].

Network Anomaly Detection is considering flow behaviors to identify abnormalities and spot the deviation that appears within the flow. To apply that, we need to define clearly what normal behavior is, and how far the features should deviate to be considered as an anomaly. For instance, cyber-attack traffic is acting differently than benign traffic. That deviation can be considered as an anomaly by the Anomaly Detection System.

2.2 Network Attacks

At first glance, monitoring the network traffics can face several difficulties without having a prior understanding of the main network components behaviors (servers,

firewalls, PC's...etc.). However, when we transform this wide mess of network traffic into Graph-Based structures, we can determine where we should focus based on the position and influence of each node. This will help to gain a better understanding of the network and therefore ease the process of observing abnormal activities. In this thesis, we focused on analyzing Six network attacks: DDoS, Botnet, Port Scan, FTP-SSH Patator, Infiltration – Dropbox download, and a collection of Web-Attacks. Each attack requires a suitable setup for building its graph, this setup will help to explore its impact in order to achieve a high detection rate. Chapter 4 shows which model will be the best for each attack.

2.3 Graphs

Graphs are composed of nodes connected by edges. A node can refer to an object (human, computer, user privilege, etc.), and edges describe the relationships between nodes. This structure simplifies the visualization of data based on its unique characteristics. There are many metrics used to measure graphs, such as distance between nodes, graph density, neighborhood relations, etc. In our work, we are going to focus on: shortest paths, graph communities, node centrality measures, and modularity. Those metrics will be used to better understand the components of the graph.

In our work, a graph is composed of machine identifiers (IP or SOCKET). Let $G = (M, C)$ where M is the set of all machine identifiers in our network flow, and C represents the set of all connections. Figure 2 shows the IP-Based model where nodes

defined as a machine IP, and Figure 3 shows a SOCKET_Based Model, where the PORT and IP are combined to identify nodes.



Figure 2: IP_Based Model

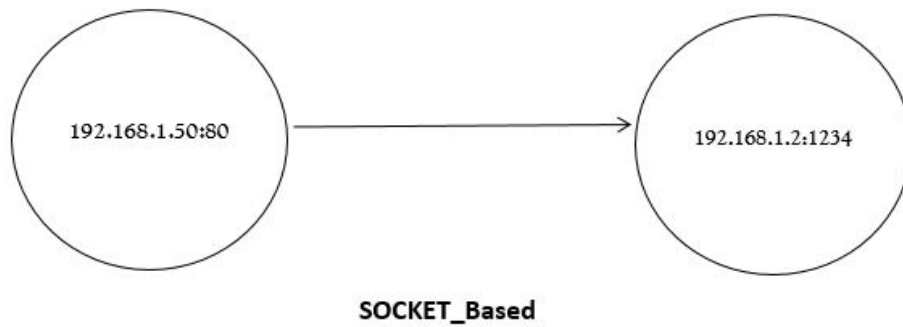


Figure 3: SOCKET_Based Model

2.4 Graph Database

Graph Database is a NoSQL database model constructed in Graph-Based structure, where all data is saved in a node/edge form. Nodes represent entities and edges show the relations between them. Graph Databases are utilized in enhancing responses when

dealing with massive relational databases. The way the Graph Database is designed improved both speed and scalability compared to relational databases where many expensive join operations are required [49].

In this thesis, We convert network flow into a Graph-Based data structure. Then we analyze graph metrics values to detect any deviation that cyberattacks can make to the graph components. To do so, we utilized many graph-based tools such as Gephi [20], igraph package in R language [54], Cytoscape [52], and Nodexl[53].

2.5 Graph metrics

Graph metrics can vary based on edge direction. it can be either directed or undirected. An undirected graph is a set of connected nodes where all the edges are bidirectional, which means all the edges are considered as a path in both directions. On the other hand, a directed graph is a set of connected nodes, where all the edges are directed from one node to another. The edges are typically drawn as arrows to indicate the direction Figure 4.

Nodes are important for different reasons, based on their positions, or on the identity of whos occupies these positions. The value of a node can be decided using Graph-Based metrics, which use to detect abnormal activities. Graph-Based metrics are divided into the following categories:

- Basic metrics, such as nodes count and edges weight; the more connections are there, the more weight the edge will have.

- *Degree* measurements represent the actual number of unique edges that are directly attached to the node. Considering edge direction *Degree* measurements are divided into three main values: 1) *In-Degree*: the number of incoming edges, 2) *Out-Degree*: the number of outgoing edges, and 3) *Degree*: the total number of edges that directly connect to the node.
- Diameter-Based measurements, 1) *Closeness Centrality* [24] where each node will have its *Closeness* values based on how close the node is to all other connected nodes, 2) *Harmony Closeness Centrality* [26] is a harmony approach that measures *Closeness* while dealing with unconnected nodes, 3) the *Eccentricity* value which measures the influence that a node has in a network by calculating the distance of the furthest connected node. Finally, the Flow-Based measurements, and here we will consider the most common metric of this group which is *Betweenness Centrality* [25], it concentrates on how a node participates in connecting graph parts.

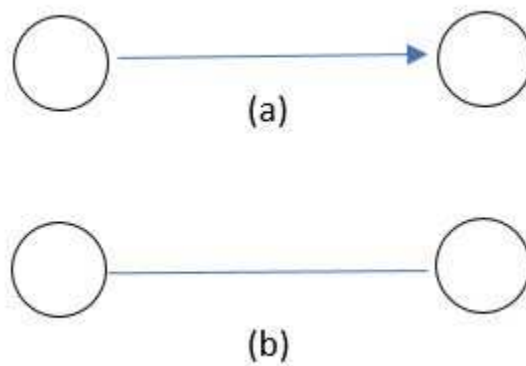


Figure 4: Edge Directions, (a) shows the edge as an arrow represent the flow direction, where (b) represent an undirected edge

2.6 Centrality Measures and Node Ranking

Centrality, in general, reflects how important a node is based on specific criteria. Node influence can be measured differently according to the chosen features. Those features include the number of connections with another node, the position of the node in connecting graph parties, the distance between nodes, number of hops, or the type of neighbors. Considering these features affect the node influence and thus affect the centrality measurements.

2.6.1 Degree Centrality

Degree centrality considers the node direct connections. the more connections a node has, the more *Degree* value it earns within the graph. For graph $G = (N, E)$ where N is the set of all nodes, and E represents the set of all edges, the *Degree* value of a particular node n is the summation of all edges that have direct contact to it.

Degree centrality can be used to measure how active a node is, regardless of the rank of neighbors who connected to it. In an undirected graph, all the edges are used in both directions. Differently, a *Degree* in a directed graph has two sub-measures based on the direction of the connection, *In-Degree*, and *Out-Degree*.

2.6.2 In-Degree Centrality

In-Degree measure denotes all the incoming edges that are targeting the node from its direct neighbors. In a network flow, the high value of *In-Degree* could be an indicator of a highly visited site or a Domain Name System (DNS) server, where most other nodes are frequently in need to reach it.

2.6.3 Out-Degree Centrality

Out-Degree is reflecting the sociability of the node. Active Directory (AD) servers usually have a high number of outgoing connections in order to manage the privileges and distribute the rules among connected users and machines.

2.6.4 Closeness Centrality

Closeness Centrality represents the average length of short paths between the node and all nodes that connected to it. Therefore, the closer node can get to other nodes, the more central position it occupies. For a node $(s) \in N$ in Graph $G = (N, E)$ the *Closeness Centrality* of s is $(C_{cc}(s))$ is defined as:

$$C_{cc}(s) = \frac{1}{\sum_d dist(s, d)}$$

Where $dist(s, d)$ is the shortest path distance between the source node (s) and the destination node (d) .

2.6.5 Harmonic Closeness Centrality

A harmonious approach of measuring *Closeness Centrality*, where the nodes within the graph are not necessarily connected. This is achieved by replacing the distance for an isolated node with 0, instead of infinity in the regular closeness centrality measure.

2.6.6 Eccentricity

Eccentricity is a Diameter-Based Centrality measure that simply shows the distance between the node and the furthest connected node to it. The *Eccentricity* of node n is calculated by computing the shortest path between node n and all other nodes in the graph, then chooses the longest path among those shortest paths and assign it to the node as an *Eccentricity* value. Let n and u be nodes, and $dist(n, u)$ is the shortest distance path between them, then the *Eccentricity* ($C_E(n)$) equals:

$$C_E(n) = \frac{1}{\max\{dist(n, u)\}}$$

2.6.7 Betweenness Centrality

Betweenness can be used in measuring both nodes and edges. However, it is usually linked with the node position. Considering all the short paths between graph nodes, the *Betweenness* value of a node is the number of short paths that this node is part of. The more *Betweenness* value the node has, the more influence it has in connecting graph parties.

For a node $n \in N$ in Graph $G = (N, E)$, *Betweenness* Centrality of n ($C_{bc}(n)$) is defined as:

$$C_{bc}(n) = \sum_{s \neq d \neq v \in V} \frac{\sigma_{sd}(n)}{\sigma_{sd}}$$

Where σ_{sd} is the number of all short paths that pass through n . In a directed graph, *Betweenness* is correlated with the type of edges (*In-Degree*\ *Out-Degree*), so the node with a zero *in* or *Out-Degree* will have (0) *Betweenness* value. Web servers, for instance, usually have high *Betweenness* value due to the request\response mechanism that is implemented with its clients. However, a network printer within the LAN infrastructure may have only *In-Degree* edges, thus have a very low or zero *Betweenness* value.

2.6.8 Modularity Class

In a network environment, modularity considers a community of nodes based on their internal connections and the density of those connections. Nodes within the same community are sharing the same modularity value. Graph-Based modularity is used in detecting communities and analyzing their internal connections and their relations with the other communities within the graph.

2.7 SMOTE: Synthetic Minority Over-sampling TEchnique

Due to the highly imbalanced datasets used in this study, there is a need for an approach to overcome this issue. Therefore, we used a sampling technique called SMOTE, which has been proposed by Chawla et al. [19]. We applied SMOTE over our datasets to enrich the minority class by creating synthetic samples. To do this, samples have been collected from the minority class to create artificial samples to enrich the minority group. Neighbors from the nearest k relatives are randomly nominated, based on the amount of needed over-sampling.

2.8 Feature Selection (Boruta)

To manage high-dimensional datasets, in terms of the number of features and metrics in a much efficient way, and since Generated Labelled Flows (GLF) data occupied a very large set of features (80 features per each file) there is a need to implement a features selection statistical technique that can help to extract useful information from this huge volume set.

Boruta is a feature selection algorithm that captures the important features from the features set, by going through all the features in a predefined number of iterations. After that, recording the hit times for each feature to define the importance of each one. At last, labeling all the features as either confirmed, rejected, or tentative, based on their scores in mean, median, min, max, and the number of hits.

2.9 Classifiers

To evaluate the possibility of detecting intrusions within the network by utilizing only graph metrics, we implemented three well-known machine learning algorithms over the generated Graph-Based datasets and the benchmark dataset CICIDS2017 [8]. Firstly, we explored the datasets for each attack, clean all the missing cases which we found in both Botnet and web Attack files. Then we performed feature selection using Boruta [31], which estimates the importance of each feature within the datasets and fits it into one of three ranks (confirmed, rejected or tentative). For this study, we only considered the confirmed features. Finally, we implemented the classifiers and reported the results.

2.9.1 Support Vector Machine (SVM)

Support Vector Machine is a practical algorithm for high-dimensional spaces; mainly, it utilizes supervised learning models for both classification and regression analysis. SVM is challenged to partition a feature space into two or more groups by finding an optimal means of separating based on their known class labels.

2.9.2 Random Forest

Random forest is mainly considered as a method for classification and regression [44]. It works by building many decision trees at the time of training and extracting classes, which is layered based on the classification prediction of the mean of specific trees. The random forest algorithm works by compiling predictions made by multiple

decision trees of varying depth, where each decision tree in the forest is trained on a subdivision set of the dataset, which is called the bootstrapped dataset.

2.9.3 Naïve Bayes

Naïve Bayes is a simple and powerful logic-based technique that is often known to outperform complex Naïve Bayes algorithms for very large datasets, Naïve Bayes classifiers are highly scalable. They consider that the feature is independent of the value of any other feature. Moreover, one of the Naïve Bayes advantages is the fact that it requires a small number of training data to guess the necessary parameters for classification.

2.10 The Complexity of Graph-Based Analysis

To measure the time complexity of creating a graph, we need to count precisely how many nodes and edges the graph has. In addition to the type of links between nodes. For instance, both *Closeness* and *Betweenness* centralities are relying on calculating the shortest path between nodes. To illustrate, let us assume the total number of nodes is N , and the total number of edges is E , the time complexity for an unweighted network is $O(NE)$, where $O(NE + N^2 \log N)$ is the time on weighted networks [37], storage, on the other hand, will require $O(N + E)$ space.

CHAPTER 3

LITERATURE REVIEW

3.1 Anomaly Detection

Many approaches have been proposed to detect unknown attacks. For instance, data mining [12] and unsupervised machine learning [15][16] are common approaches to solve this issue. However, a robust trend in detecting Zero-Day Attacks is by considering the context of the events. An attempt to follow this trend was shown by Dimopoulos et al. [17] by using Contextual Anomaly Detection (CAD). The authors have focused on the contextual information of instances to detect anomalies while decreasing the FP rate. In addition, they utilized FastDTW to compute the distances then cluster similar instances according to their distances from the context set. Next, they perform classifications based on the anomaly score within a specific time window. The results showed improvements in detection accuracy.

While Anomaly Detection techniques are costly in terms of time and computation, production environments require a more scalable and adaptable solution. To overcome that, Mdini et al. [18] developed a lightweight application called Watchmen Anomaly Detection (WAD), which efficiently identifies anomalies from the network data flow. Their algorithm has been proposed based on unsupervised learning that relies on pattern recognition to ease the detection process for mobile operation in real-time. Furthermore, Ammann, Wijesekera, and Kaushik [6] have argued that the attack path may provide more

information than the analyst needs. This would affect the quality of penetration in a scalable environment. Their assumptions reduced the complexity of the analysis problem from exponential to polynomial. Even very large networks could be analyzed in an acceptable timeframe using the right features only.

3.2 Graph-Based Analysis

3.2.1 Node Analysis

Previous work that was addressed graph-based analysis to detect network security threats was too humble compared to the promising potential findings that could be achieved in terms of accuracy and performance. A graph can be used to deal with heavy relational databases, like representing network communications. Grandjean in his work [10], showed how nodes and directed edges could be used to simulate the user activity on Twitter. They also showed how it could be possible to analyze digital humanities by visualizing "who is following who," and they were able to detect the relationships and find a user who may be involved in a particular group. Another approach was the work of Grana et al. in [4], where they studied computer network attacks. They aim to reduce the FP rate, which considered one of the major factors that many anomaly detection techniques fail to achieve due to the variety of normal activity over time. They pointed out two ways to address the issue of the behavioral method: 1) Improve the specification of the network under normal conditions and 2) Enhance the comparison between the hypothetical and normal behavior of a network that has exposed to risk and that of a normal one. Another example of applying graph analysis is the work of Kent and Alexander [5]. The authors by using graphs to represent

user authentication activities have proved that it is possible to find differences between non-privileged and privileged users considering the size and complexity of node connections.

Akoglu, Tong, and Koutra conducted a survey addressing how graph analysis can be the new approach in detecting anomalies [21]. They studied how nodes could be grouped into a few clusters based on their relations, weights, connections, and its similarity with other nodes. Those measures have helped to spot and analyze normal typologies, therefore, identifying any patterns that deviate from normal as an outlier. They confirmed the importance of understanding how anomaly could exist. For instance, to detect Portscan attacks, we need to break down the components of that attacks to the following: source node which sends many packets in a short duration of time to a particular destination node creating a unique pattern that can be detected using the right metrics (IP's for both source and destination, duration of transactions and the number of edges).

Some studies have implemented a Graph-Based approach to convert the log file to a graph-based structure. They used it to understand log files better [2][3]. By applying that, authors became able to detect and study anomaly patterns gaining a more observable view of the entire users' events. Hagberg et al. [2] managed to structure the authentication graph as a bipartite graph. They found that limiting the number of stored credentials in computer caches can reduce the chance of a "credential hopping" attack within the enterprise network.

Neil et al. [3] introduced the concept of a new edge, which aims to determine all the edges that have never been used to communicate between two nodes within a subgroup. This concept illustrates how a random set of edges (time-series of communications between computers) can be used to identify local anomalous areas of the network by monitoring attacker paths through the nodes.

Many Graph-Based approaches have been applied to detect fraud and abnormal behaviors. Each approach has focused either on the node values, edges values or both to understand the nature of normal metrics and subsequently find any anomalies within the graph. Eberle and Holder, in their work [35], mentioned that Graph-Based approaches were used to detect anomalies that presented in areas where anomalies involved in an entity/relationship are very similar to non-anomalous behavior. They introduced three new algorithms (addition, modification, and deletions) to resolve the issue of detecting anomalies in all its possible variations in the graph. In other words, Graph-Based methods were designed to detect anomalies in labeled graphs, where anomalies contained unanticipated deviations that could help to resemble abnormal behavior. So, any substructure could be considered as abnormal if it diverges from the “normative” substructures based on the Minimum Description Length (MDL) principle that they create. The work of Noble and Cook [36] presented techniques for detecting Graph-Based Anomalies by calculating the consistency of the graph. They provided a measure to calculate the regularity of the graph by utilizing the concept of conditional entropy.

Another usage of Graph-Based analysis was carried to observe internal threats. Those kinds of threats can significantly affect the stability of critical systems like national security, financial sections, and the privacy of all the existing users. Authors of [39] proposed a hybrid framework-based on anomaly detection and graphical analysis methods to stand against those types of threats. Their approach aims to analyze mixed data to isolate potential malicious users who might be hiding as a legitimate user. After that, distinguish the users who exhibit behaviors that are considered suspicious. Due to the critical impact of internal threats over the assets, authors raised the sensitivity level of their approach to include suspicious users even if only one suspicious parameter found. In addition, Akoglu, Tong, and Koutra delivered a comprehensive structured summary of the most recent methods used in detecting anomalies in databases [37]. Those Databases were formed in a Graph-Based structure as well. They showed how the graphs could capture real-world events.

3.2.2 Edge and Community Analysis

On the other hand, many Graph-Based analysis researchers focused on analyzing connections between nodes (Edges metrics). For instance, the authors of [38] focused their work on graph edges to distinguish between suspicious and typical node behavior. This is accomplished by leveraging edge metrics to introduce “EDGECENTRIC” which is a simplified comparison-based approach that uses pressure to detect anomalies of an edge-attributed graph. In addition, they managed to successfully detect numerous anomalies in

several large real-world charts with over one million connections, and over 545k users with 0.87 accuracies.

Grana et al. [4] tested the effect of setting new controlling rules on the network environment in order to minimize the impact of having a large connected chain of computers after visualizing the data flow as a graph. Subsequently, they applied cache rules which allowed them to spot the decrease in the big chain size, reducing the possibility of attack separation.

A different area of focus was covered by Chen, Hendrix, and Samatova [40] where they conducted community analysis within the graph. Their approach utilized community-based analysis for understanding Graph internal interactions to observe abnormal behaviors. To do so, they presented a method to detect anomalies within the community components interactions, then categorize each community accordingly. Furthermore, to discover the community-based anomaly, they developed a scalable, free of parameter algorithm. Finally, they conduct a comparison between their proposed algorithm and some available algorithms in terms of performance measures over synthetic flows and real-world baseline datasets. Their approach achieved a noticeable speedup in the runtime compering with the other approaches.

3.2.3 Graph Analysis Tools

Gephi and Neo4j are examples of many tools that can be used to perform a Graph-Based analysis. In a case study for utilizing graph techniques, Van Erven et al. [7] found Neo4j very effective in detecting bank fraud perpetrated by users who have shared information to confuse and pass the security checks. Thus, allowing them to steal from the banks without been detected.

Gephi is open-source software designed for network analysis [20], it combines data mining algorithms with interactive visualizations. As a graph analysis tool, it provides data filtering, nodes clustering, and graph manipulation. Additionally, it helps the analysis using graph measurement tools. For instance, Gephi utilizes Betweenness, Closeness, and Degrees for computing centrality. As for clustering, Modularity function can be used for detecting graph communities. To analyze network data, it was tested against Social Network Analysis, Geographical Network Analysis, and Time-Varying Network Analysis [22]. HUSSAIN, Muhammad, and YAKUBU [23] used Gephi and R language to conduct a study for three different huge relational datasets, Facebook, Twitter, and Dbpedia. They efficiently explore the data and visualize it under different graph measurements. Then they acknowledge that Gephi was capable of handling all tested datasets with an efficient and high-performance rate.

Neo4j [43], as one of the available Graph-Based database tools, stored the data in a node and edge structure. Moreover, nodes and edges can have multiple properties depending on the relationships between the components of the database. Nodes could be labeled, colored, and resized to have better visualization for the data. Neo4j enhances performance and extends the scalability of databases. Run Neo4j in a powerful machine could handle graphs with a million nodes and edges. However, this number can be scale up by implementing Neo4j in multiple machines, plus it could be implemented as a standalone or embedded server. Traditional databases have their weakness when dealing with many relational and heavy connected data, which requires many joins for large tables. Vicknair et al. [42] tested SQL and No-SQL graph databases for performance. The results of the experiment showed that the graph database has much more to provide in terms of structural queries. It performed faster for both text search and relational tables.

Mathew and Kumar [41], Explored the Neo4j graph database to test and analyze the NoSQL graph database, they achieved better performance comparing with Multidimensional indexing. Both search and update functions were analyzed and the response time was faster in both cases.

Another open-source network analyzer tool is Cytoscape [52]. Cytoscape is designed to visualize complex networks. It was originally designed for biological data (e.g. genes, cells, proteins, and patients)[55], and now it has been generalized for any type of complex

network analysis. In addition to visualization, it can compute many graph-metric for both nodes and edges, which can asset efficiently in observing network components activities.

3.3 Research Gap

Most of the available Anomaly-Based IDS are suffering from performance degradation, rising in false alarm rates, in addition to the limitation in handling big scalable data as network flows or authentication log files. On the other hand, Signature-Based IDS may overcome the performance issue, although it is still limited to known attacks. In the presence of the rapid increase in the transferred data, there is a real need for new approaches that can handle this huge amount of data and utilize the flow behavior to gain accurate indicators of network threats. Although many works have addressed graph-based analysis in detecting network attacks, to the best of our knowledge, none of them has addressed the deviation that occurred to Graph centralities measurements and how it can be a valuable indicator of revealing attacks impact over the network.

CHAPTER 4

PROPOSED APPROACH

Cyber-attacks are divided into two main classes based on their impact over the victim environment: Network attacks and Host attacks. Each class has its unique touch over the targeted victim. In this experiment, we focused on network attacks, where we managed to distinguish its effect over the network flow by redesigning network traffic into a Graph-Based structure (nodes and edges). Many available tools can be used to apply Graph-Based analysis in order to build and explore graph networks, in our experiment, we will implement Gephi tool. This chapter will focus more on our purposed approach, and it will illustrate each phase as shown in Figure 5, the algorithm that we designed for this work. Algorithm 1 shows the structures of our proposed approach.

Algorithm 1: Graph-Based Analysis

```
Import network flow
Build the Graph
if Graph Model = "SOCKET_Based"
    for each node do                \\ compute graph metrics

        Compute Baseline centralities
        Compute Diameter-Based centralities
        Compute Flow-based centrality
    end
    apply Visualization do          \\ visualize the graph
        Resize the nodes based on Degree value
        Color the nodes Based on community_class
        Apply visualization algorithm (OpenOrd + ForceAtlas 2)
    end
    Perform Graph Analysis do
        analyze the communities
        spot the high centrality nodes within the community
        analyze node centralities
        analyze both In\Out degree for the high centrality nodes
    end
else                \\IF Graph Model is "IP_Based"
    find the abnormal edge_weight
```

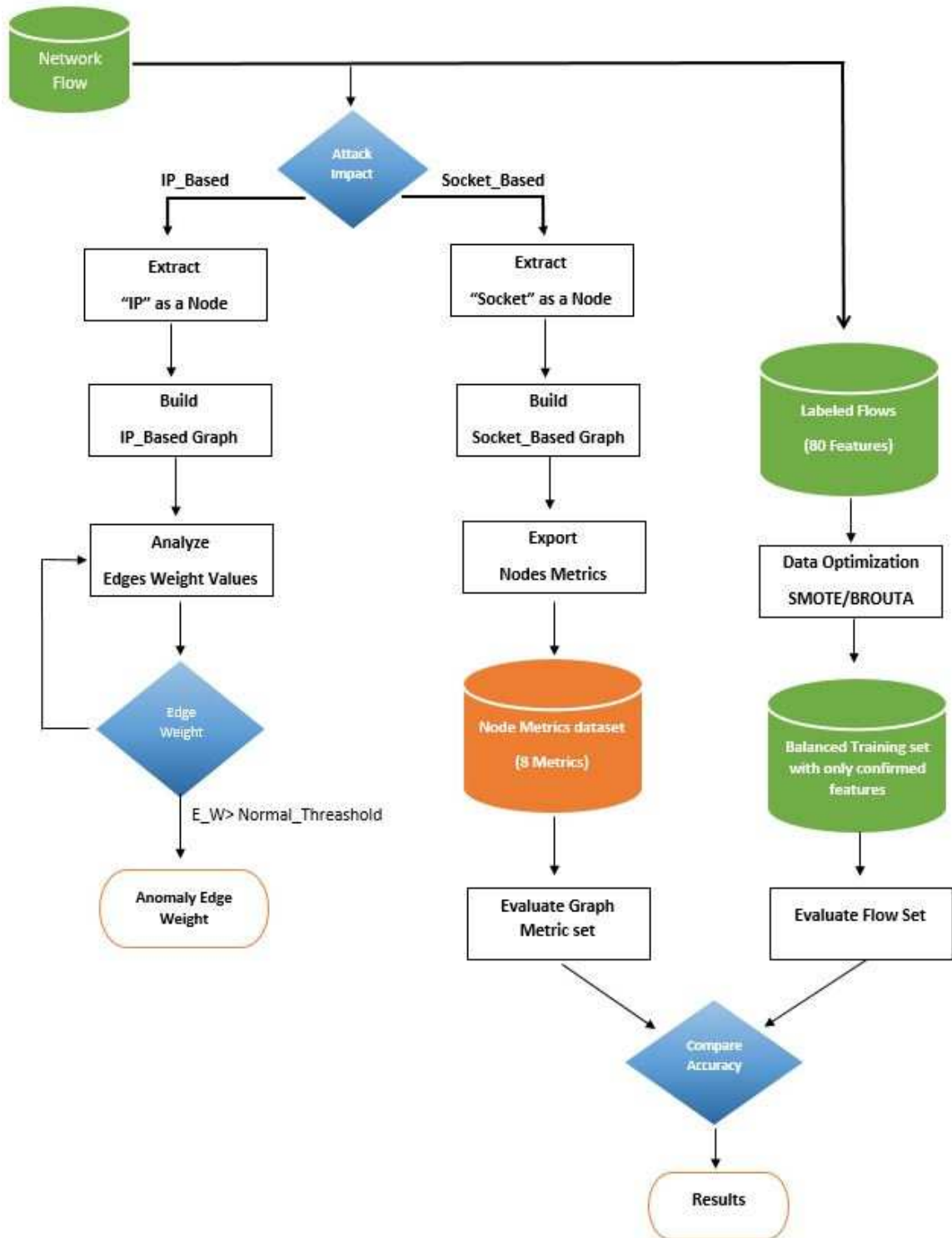


Figure 5: Graph-Based Components

4.1 Datasets

Evaluation of our approach in detecting abnormal network behaviors requires an up-to-date labeled dataset that contains a complete network flow with a recent diversity of network attacks. Therefore, we implemented our approach over the CICIDS2017 IDS dataset [8], which has been released in 2017 by the Canadian Institute for Cybersecurity, with the most common attacks based on the 2016 McAfee report. First, we describe the infrastructure of the network that generated this dataset. Thereafter, we will extract the connections from the dataset to build our graph and calculate graph-metrics for each node.

We have three different sets of data in our experiment, the first one is the original “Network Flow”, which is a .pcap file that contains all the traffics that have been captured, then we have a set of network features, and statistical data named “Generated Labelled Flows Datasets (GLF)” dataset, and finally computed graph metrics under “Graph-Based dataset (GB)”.

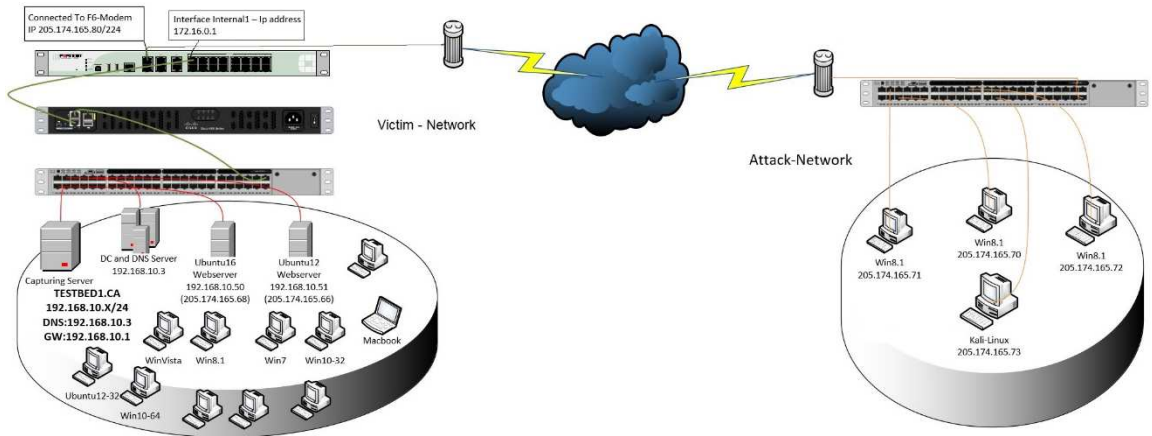


Figure 6: Testbed Architecture

4.1.1 Network Infrastructures

The CICIDS2017 dataset has been generated based on the traffic of two separate networks Figure 6. The first one is the victim-network, which contained servers, PCs, and network devices. Servers include an AD and a DNS server with IP address 192.168.10.3, plus two ubuntu web servers 192.168.10.50 and 192.168.10.51. Where the PCs represent the client machines over the victim network. Finally, router, couple switches, and a firewall as network devices. The second network represents the attack-network which consists of a set of PCs with public IPs, a router, and a switch, to perform the network attacks.

4.1.2 Network Flow

To generate the network flow for the normal activity B-Profile method has been used [8]. This method analyzes the abstract human interactions to generate realistic benign background traffic. The captured data has been collected by configuring a mirror port in one of the switch ports for five working days; on the other hand, the attacks have been launched separately each at a different period. The traffic of the above infrastructure has been used to generate two datasets, the first one is the GLF dataset, which is pre-calculated [8], and the second one is GB (Graph-Based) dataset, which has been calculated using Gephi tool to have the graph metrics for each node.

4.1.3 Generated Labelled Flows Datasets

For each attack, a .csv file has been created with 80 extracted features which have been calculated by CICFlowMeter [45]. Those features are representing a combination of network flows characteristics, flow statistical values, and network flags. Each attack family has a separate file with all the 80 features.

Firstly, GLF_DDoS dataset contained 225,746 connection records, the attack has been launched for 20 minutes by three different attackers from the attack-network, all the attack connections appeared as it came from the firewall IP, with different source ports each time. Subsequently, the GLF_BotNet dataset has 191,033 connection records. The attack was performed between 10:02 am to 11:02 am, by a single attacker and a group of 5 victim machines, the attack flow is showing the connection between the attacker and victims in both directions. Finally, the GLF_WebAttack dataset is combined with three web attacks, SQL injection, XSS, and network password cracking attack, with 170,366 connection records. In our experiment, we gathered all the attacks under the “WebAttack” label due to the similarity of the connection’s [8].

Table 1: Samples of network flow data used to build a SOCKET_Based Graph

Source	Target
192.168.10.50:33898	192.168.10.3:389
192.168.10.50:33904	192.168.10.3:389
192.168.10.14:59135	65.55.44.109:443
192.168.10.14:59555	192.168.10.3:53

4.1.4 Graph-Based Datasets

To generate the Graph-Based dataset, we extracted all the source and destination information from the Network Flow (Table 1), then we imported this data to Gephi tool to create a graph network. Each node in the graph is representing a machine socket (or only IP like in portscan). Later we calculate the graph-metrics for each node based on its connections and position to end up with eight features that represent the graph metrics for each node (Table 2). Those metrics represent the centrality measurements and modularity class. Each attack will have its .csv file that contains those metrics for each node. Firstly, GB_DDoS dataset contains 87,458 *SOCKET_Based* nodes. Secondly, the GB_BotNet dataset with 98,270 *SOCKET_Based* node, next GB_WebAttack dataset contains 87,632 *SOCKET_Based* node. Finally, the GB_Portscan file built differently as an *IP_Based* model with 248,848 records, where each record represents an edge from node to node, plus the weight of each edge. The weight value increases as the number of connections between nodes rises.

Table 2: Samples of the Graph_Based metrics from the GB dataset

NODE	Indegree	Outdegree	Degree	Modularity	Eccentricity	Closeness	Harmonic	Betweenness
192.168.10.3:53	55098	6	55104	1	1	1	1	482682
192.168.10.1:53	2500	0	2500	5	0	0	0	0
162.208.20.178:443	1270	1024	2294	1356	22	0.204093	0.284627	161622152.6
205.174.165.73:8080	1226	705	1931	1	26	0.118575	0.166515	30649288.07
72.21.91.29:80	754	652	1406	54	22	0.188818	0.252888	74915896.47
23.61.187.27:80	612	344	956	1181	22	0.177794	0.226553	39958260.82

4.2 Graph Models

To get the most accurate result that can assist in highlighting the presence of an attack, we need to build our graph while considering the impact of each attack on the network flow. In our experiment, we proposed two different models: *SOCKET_Based* and *IP_Based* graphs.

In *SOCKET_Based* graph, each node is considered as a socket, where the IP and PORT are combined to identify the node Figure 3, whereas, in *IP_Based* graph, each IP address is considered as a separate node, the focus here is on a machine to machine connection regardless of the targeted port as shown in Figure 2.

4.3 Build the Graph

The goal of building the graph is to create a set of graph measures that enriches our understanding of the flow to build a traceable structure and analyze the data flow in a more efficient way. To do this, we need to illustrate some points that we need to consider while dealing with graphs.

To build a graph we need to transfer all the connections that we have to an edges-like form, where each edge should represent a map from node to node, we utilize the edge weight to represent the density of the connections, therefore, reduce the total number of edges without affecting the information integrity. Nodes, on the other hand, represent all machines that exist in a captured flow. Consequently, as the number of machine sockets increases the number of nodes in the graph will rise as well.

4.4 Compute Graph Metrics

By calculating network diameter [27], we measure the distance between nodes, the connection type, and the number of connections for each node. We calculated different centrality metrics for each node, the *Betweenness* Centrality which addresses the number of shortest paths the node is part of. The *Closeness* Centrality, which addresses how close the node is to all other nodes. *Harmony Closeness* Centrality, which is a harmony approach to measure *Closeness* with respect to unconnected nodes. Lastly, *Eccentricity*, which simply shows the distance between the node and the farthest node from it (considering only the shortest paths).

lastly, the Modularity Class, which aim to group the similar nodes into internal communities, based on how dense the connections among them and how related those nodes are, it can be used in detecting relations between nodes within the same community.

4.5 Analyzing Graph-Metrics

There are different aspects to estimate the importance of a node. Here we will show some ways to measure node influence over the graph. Centrality measures proposed to identify the importance of a node. In other words, if we considered the *Betweenness* centrality as a measure of importance, the AD server will be considered as an important node due to its direct connection with most of the network entity. Here we will illustrate the variety of the graph metrics and the knowledge that can be revealed by understanding those metrics.

Centrality Measures on graphs are divided into three categories: baseline measures, diameter-based measures, and flow-based measures. Each one of these groups has a set of centrality measures. In this experiment, we will use the most common ones from each group.

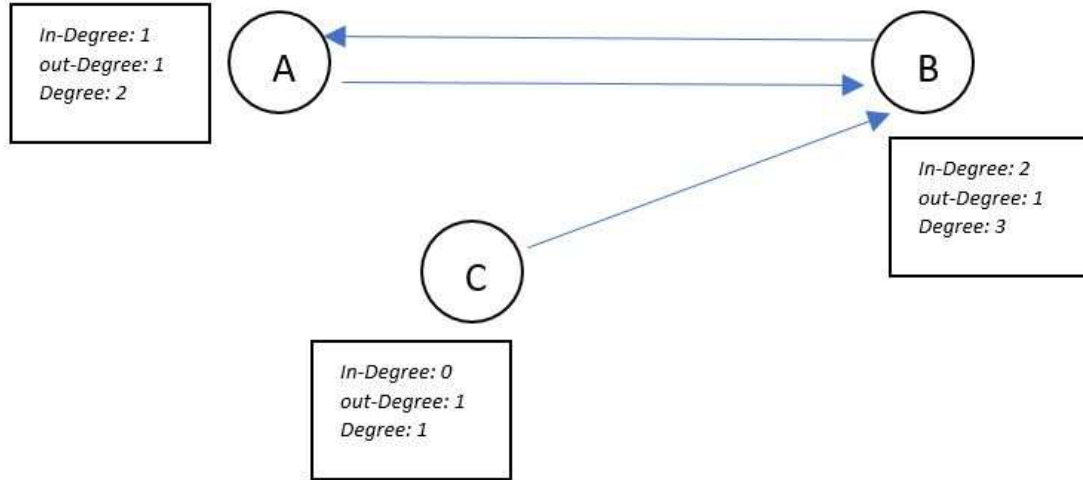


Figure 7: Degree Centralities, shows how Degree values are measures

4.5.1 Baseline Measures

In-Degree, *Out-Degree*, and *Degree* centralities are interested in the number of direct-connected edges (degree of length 1), although *Degree* measures are the simplest ones, yet it can be calculated more efficiently for large networks compared to other centrality measures, in addition, it can indicate to valuable information about graph density and internal communities communications. Figure 7 shows an example of a small graph, node A with two edges to and from node B. Therefore, it has centrality of degree two, one of *In-Degree* and the other as *Out-Degree*. Node B has more central value in terms of *Degree* centrality. At last, node C is the lowest one with only a single *Out-Degree* edge.

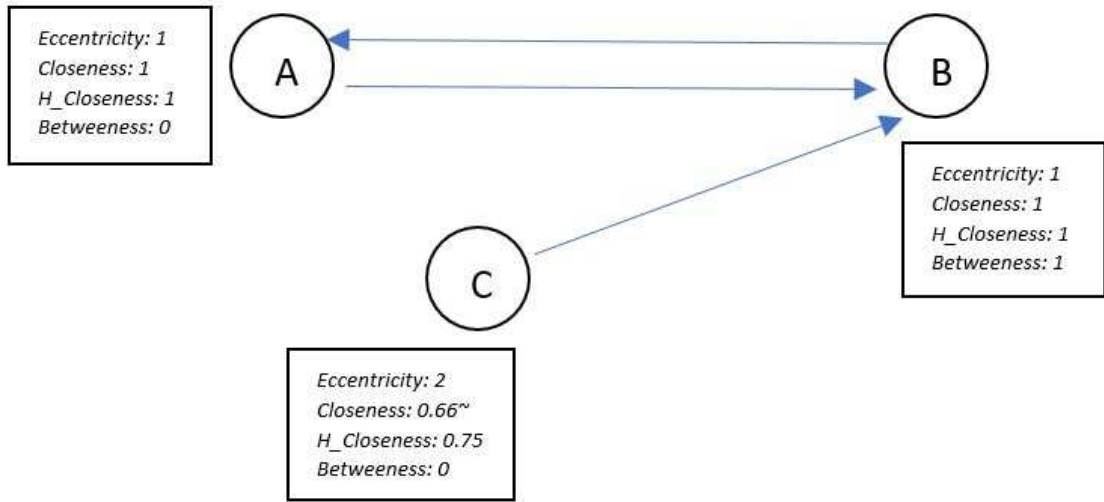


Figure 8: Diameter & Flow Centralities, shows how Diameter and Flow centralities are measures

4.5.2 Diameter-Based Measures

Diameter-Based measures are concerned with the number of hops that the node needs to pass in order to communicate with other nodes. *Closeness* and *Harmony Closeness* centralities are studying how close a node is to different graph parties by calculating the average of the shortest distances to all other nodes. So the higher *Closeness* value a node has the least hops it needs to reach to other nodes. In contrast, *Eccentricity* shows the maximum distance between a node and the furthest connected node to it. Diameter-based measures can be utilized in distinguishing low *Degree* nodes. To illustrate, if we have many nodes with limited attached neighbors that do not reveal much information about the node, we can utilize *Closeness* and *Eccentricity* to add a deeper level of analysis. We can achieve this type of analysis by considering relations with all neighbors, whether they are directly or indirectly connected. In a directed graph, for a node to have a Diameter-Based centrality value, it must have at

least one *Out-Degree* edge. Figure 8 shows the diameter-based metrics for each node. We can notice that the highest short path is the path that C needs to cut to reach A, due that the *Eccentricity* value of C is 2. *Closeness* and *Harmony Closeness* centralities are equal to 1 for both A and B since they have a direct connection to all nodes that attached to them. where for C the *Closeness* value is reduced due to the path that it needs to take to reach A. In fact, we can realize that whenever the *Eccentricity* for a node is equal to 1, the *Closeness* centrality will be 1 as well.

4.5.3 Flow-Based Measures

The last group is computed based on the information that the edges have. For this experiment, we will focus on the most common centrality measure, which is the *Betweenness* centrality. In *Betweenness* centrality, the node is earning more importance for every time it being part of the shortest path between other nodes. In other words, from the set of all paths that go through a node, only the shortest paths are counted. The higher *Betweenness* value might consider as an indicator of having higher influence in linking different parts of the network together, for instance, due to their connections, servers usually have higher *Betweenness* value. Therefore, it usually allocated in the center of the graph. For any node to have non zero *Betweenness* value, it must have at least two *Degree* values with different nodes, one as *In-Degree* and the other as *Out-Degree*. In Figure 8, only node B has fulfilled the *Betweenness* conditions. Thus it does have a value of 1, which represent the number of the only short path that node B participate in (the path from C to A)

4.5.4 Community Analysis

Modularity is a graph measure that addresses the communities within the graph. Applying *Modularity* means dividing the graph into clusters or groups of nodes, where nodes within the same community have a higher connection with each other compared with the connection with the nodes from outside the community. Implementing *Modularity* can help in detecting both attacker and victim trace since they usually exist within the same community based on the attack impact.

CHAPTER 5

EXPERIMENT & RESULTS

In our experiment, we will build a graph for each flow, compute graph metrics, and analyze those metrics to understand the abnormal behavior of the attacks. We start with the Benign flow to identify network components and to compare it with the deviation that may occur when analyzing the attack flow. Then, we visualize the graph in a way that can help in spotting the attack community. Finally, we conduct a comparison using the extracted metric to detect an anomaly, against existing network anomaly detection systems.

Table 3: Statistics of Graph-Based Datasets

Data	# of Network Flow Connections	Graph Model	# of Nodes	# of Edges
Benign	529,918	SOCKET_BASED	189,679	341,840
DDoS	225,745	SOCKET_BASED	87,458	115,006
BotNet	191,033	SOCKET_BASED	98,270	130,370
WebAttack	170,366	SOCKET_BASED	87,631	116,360
PortScan	286,463	IP_BASED	4,404	13,742
FTP-SSH	445,910	SOCKET_BASED	184,679	282,450
INF	288,603	SOCKET_BASED	122,586	218,935

5.1 Benign Data

Benign data represents the normal human activities, as shown in Table 3 around 530K records of network flow have been transformed into a graph-based structure with almost 190K nodes and 342K edges. After computing the graph metric for each node, the nodes metrics have been ordered based on the Degree value. Table 4 list the top 9 nodes, the higher number of connections are attached to the DNS and Active Directory server, which justify this massive number of *In-Degree* edges. Most of the values appear in Table 4 are either related to the local DNS server, global DNS server (like with 162.208.20.178), or legitimate highly visited sites.

On the other hand, Table 5 list the centrality measurements of the same nodes listed in Table 4. Here the focus will be on the type and value of the connections, for diameter-Based metrics, we can see the relationship between the eccentricity and closeness centralities, whenever we have a lower value of eccentricity the closeness centrality will be higher, for instance, the first row represents the DNS node, we can see that the eccentricity is equal to 1, and that is justified due to the direct connection that the DNS server usually has with all other nodes, therefore, increased the *Closeness* centrality value to be 1 (direct connection with all neighbors). *Betweenness*, on the other hand, is very low due to the low number of *Out_Degree* edges as illustrated in Table 4. In other words, most of the nodes are connected to the DNS server, but none of them can utilize these edges to reach other nodes through the DNS node. The rest of the table is shown nodes with high *Eccentricity* which reduced their *Closeness* centrality.

Table 4: *Degree & Modularity* Metrics for the top 9 nodes based on *Degree* value

Node	indegree	outdegree	Degree	Modularity
192.168.10.3:53	4125	0	4125	1721
23.111.11.111:443	1342	1268	2610	798
192.168.10.1:53	2499	0	2499	10
162.208.20.178:443	1601	0	1601	267
23.50.75.27:80	603	45	648	93
151.101.208.166:443	376	218	594	669
151.101.20.166:443	308	176	484	798
72.21.91.29:80	479	4	483	340
178.255.83.1:80	462	3	465	49

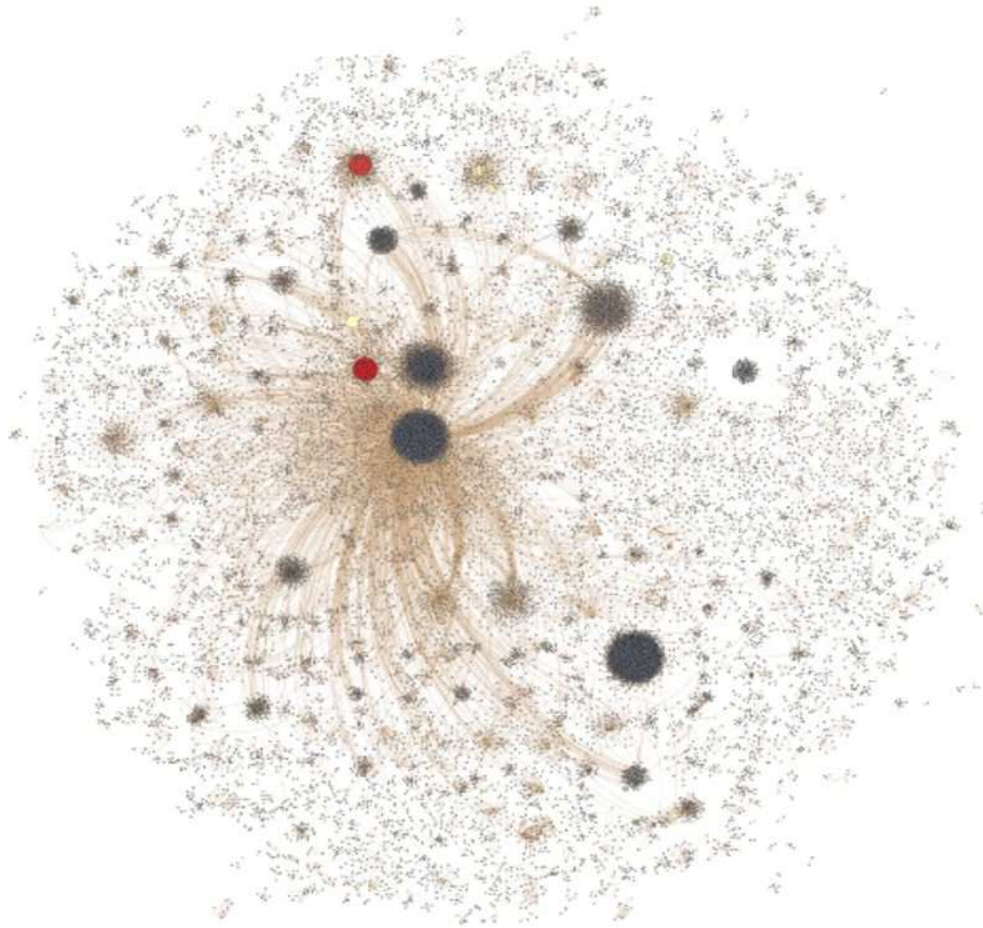


Figure 9: Benign Graph, shows how the benign traffic transformed into a graph

Table 5: *Centrality Metrics* for the top 9 nodes based on *Degree* value

Node	Eccentricity	Closeness	Harmonic	Betweenness
192.168.10.3:53	1	1	1	0.000865
23.111.11.111:443	15	0.205471	0.237436	0.329717
192.168.10.1:53	0	0	0	0
162.208.20.178:443	13	0.227097	0.263116	0.738312
23.50.75.27:80	13	0.212305	0.244214	0.430195
151.101.208.166:443	15	0.194558	0.213081	0.065663
151.101.20.166:443	15	0.189803	0.2072	0.048324
72.21.91.29:80	14	0.21352	0.24655	0.331681
178.255.83.1:80	17	0.132173	0.137945	0.072943

5.2DDoS

5.2.1 Graph Model

DDoS performed by exploiting many machines (IP) to maliciously overwhelm a specific service (PORT) at the victim machine. Which causes targeted service to be unreachable for other legitimate connections. In light of this fact, we chose the SOCKET_Based model to construct our graph. By applying that we can effectively detect attack behavior. For

instance, if DDoS attack is targeting the FTP server, all the malicious connections will be targeting PORT 21 at the server machine, leading to generate a noticeable change in the node metrics that represent (server_IP: FTP_Port).

5.2.2 Outcomes

Both Graphs in Figure 9 and Figure 10 have been built based on the *Degree* centrality. According to that, the more direct connections the node has, the more significant it appears in the graph. Over 225K of network flow records have been transformed into a directed graph, with over 87.4K nodes and 115K edge. Next, we compute the centrality measures for each node. Then by comparing the degree values in Table 4 with the ones in Table 6, we can see the sudden appearance of the node (192.168.10.50:80) with over 65K connected edge. The DDoS attack has affected the victim node to be the most connected node among others. The massive incoming traffic affects the *In_Degree* value and thereafter the *Degree* value significantly. Most of the nodes that participate in the attack have no previous appearance in the graph, that phenomena affect both the *Betweenness* and *Modularity* measures of the victim node. For *Betweenness*, since the only connections that these nodes have is either "to" or "from" the victim web-server, the victim node became a valuable factor in all shortest paths that connecting those nodes. Thus, the *Betweenness* value of the web-server rose sharply to reach the peak as illustrated in Table 7.

Table 6: *Degree's* & Modularity class metrics of the top 9 nodes after DDoS

Node	Indegree	Outdegree	Degree	Modularity
192.168.10.50:80	45392	20144	65536	771
192.168.10.3:53	22172	1	22173	15
192.168.10.1:53	2442	0	2442	1233
61.251.165.12:80	920	55	975	718
72.21.91.29:80	262	224	486	711
178.255.83.1:80	399	0	399	718
205.174.165.73:8080	350	0	350	15
23.61.187.27:80	206	139	345	711
199.244.48.55:443	309	23	332	19

On the other hand, this massive traffic with the victim node gathered all the nodes involved in the attack in a mutual community. This ease the process of detecting them using *Modularity class* (Table 8). To show that we recolor the graph based on modularity values,

each community has a unique color. Therefore we can spot the community of the victim machine -the one with the Red color- Figure 11 and Figure 12.

All these massive changes in graph topology occurred after the DDoS attack has been launched for only 10 minutes (15:56 – 16:16). Unfortunately, it was unfeasible to detect the attacker nodes since the firewall was replacing the IP of the attacker with its own IP.

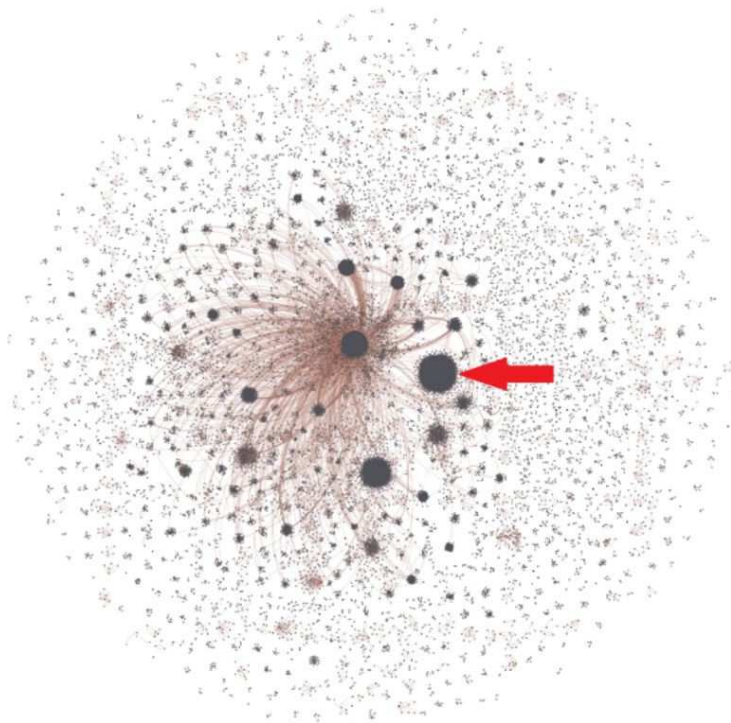


Figure 10: DDoS Graph, shows how the DDoS affect the graph, where the red arrow point to the nodes that participate in the attack

Table 7: *Centrality* metrics of the top 9 nodes based on *Degree* values after DDoS occurred

Node	Eccentricity	Closeness centrality	Harmonic closeness centrality	Betweenness centrality
192.168.10.50:80	2	0.999851	0.999926	914492479
192.168.10.3:53	1	1	1	29276
192.168.10.1:53	0	0	0	0
61.251.165.12:80	3	0.792683	0.905128	62484.82468
72.21.91.29:80	7	0.29846	0.544768	244512.9037
178.255.83.1:80	0	0	0	0
205.174.165.73:8080	0	0	0	0
23.61.187.27:80	5	0.326934	0.462248	268865.2487
199.244.48.55:443	5	0.549296	0.733761	13457

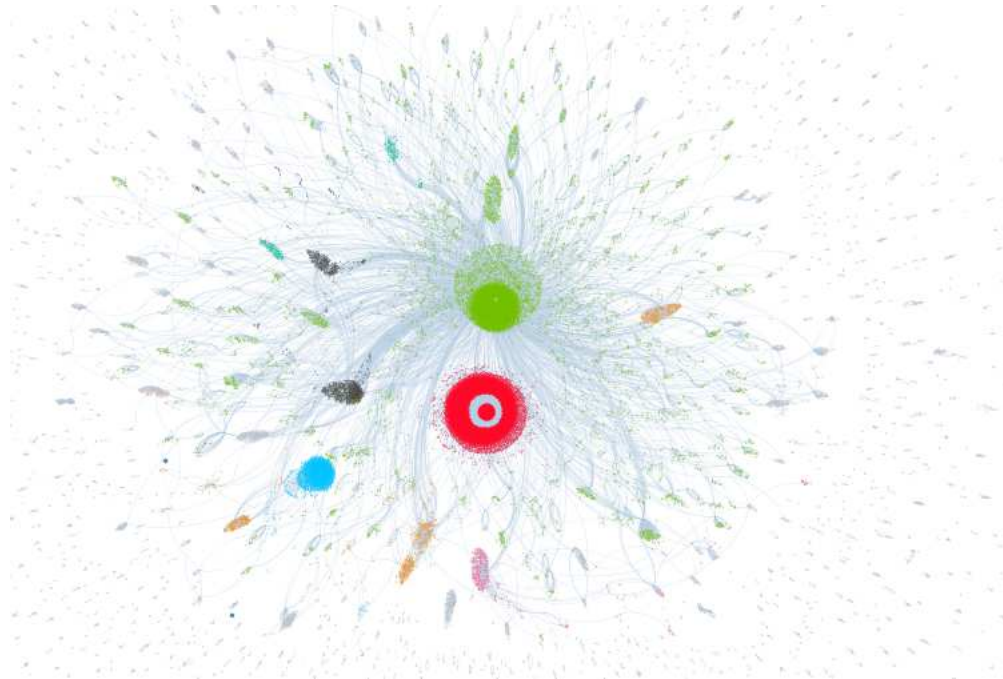


Figure 11: DDoS *Modularity* Graph I, shows the graph colored based on *Modularity* class

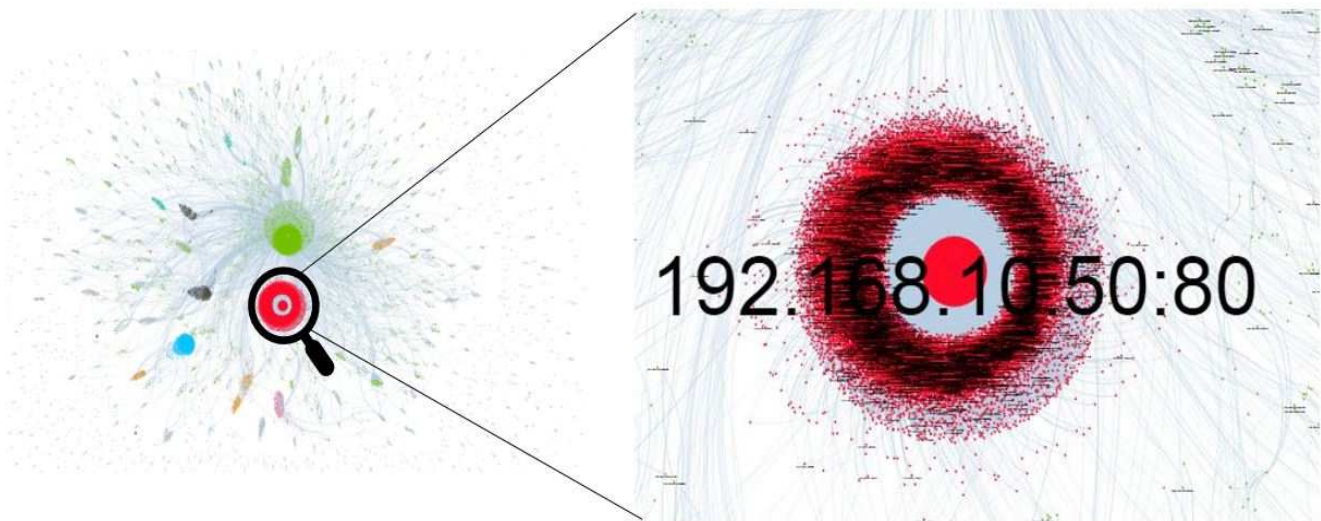


Figure 12: DDoS *Modularity* Graph II, shows how all nodes that participate in DDoS attack within the same, at the right site there is Close look of DDoS Community

Table 8: Samples of participant nodes in the DDoS attack

Node	Indegree	Outdegree	Degree	Modularity	Eccentricity	Closeness	Harmonic	Betweenness
192.168.10.50:80	45392	20144	65536	771	2	0.999851	0.999926	914492479
172.16.0.1:56411	1	2	3	771	3	0.5	0.500033	3240.785714
172.16.0.1:56414	1	2	3	771	3	0.5	0.500033	3240.785714
172.16.0.1:56471	1	2	3	771	3	0.5	0.500033	3240.785714
172.16.0.1:56472	1	2	3	771	3	0.5	0.500033	3240.785714
172.16.0.1:56568	1	2	3	771	3	0.5	0.500033	3240.785714
172.16.0.1:56569	1	2	3	771	3	0.5	0.500033	3240.785714
172.16.0.1:56604	1	2	3	771	3	0.5	0.500033	3240.785714
172.16.0.1:56627	1	2	3	771	3	0.5	0.500033	3240.785714

5.3 PortScan

5.3.1 Graph Model

PortScan Performed by targeting many PORTS at the same machine, during the attack source and destination IP's are swapped between the victim and attacker, while the port numbers are changing per each connection. Due to that, if we ignore the port number here, all the issued connections will play a part in raising the edge weight that linked the victim with the attacker. In light of that *IP_Based* model will be applied, where each node will represent a machine IP address.

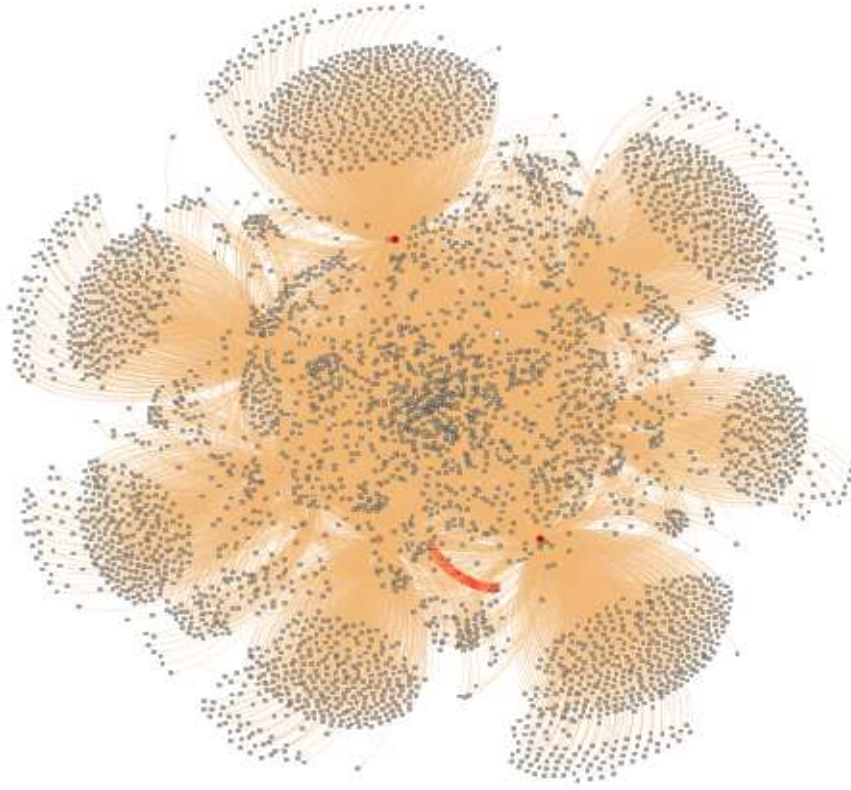


Figure 13: Shows the graph upon portscan attack

5.3.2 Outcomes

A higher edge_weight value interpreted as a strong connection. It can be used to demonstrate how intensive the interaction is. By monitoring the edge_weight in both (Table 9 and Table 10) we can see the massive growth in the edge weight that linked both attacker and victim node. Therefore, ignoring the port number in graph topology can assist in detecting the attacker path by making all the requests that the attacker sends to scan ports tunneling through the same edge. Figure 13 and Figure 14 shows how the thickness of the edge represents the PortScan activities raised after the attack.

Table 9: Edges weight of Benign flow

Benign		
Src_IP	Dst_IP	Edge_Weight
192.168.10.3	192.168.10.1	67,015
192.168.10.9	192.168.10.3	19,411
192.168.10.5	192.168.10.3	18,871
192.168.10.12	192.168.10.3	17,991
192.168.10.14	192.168.10.3	15,914
192.168.10.8	192.168.10.3	14,395
192.168.10.16	192.168.10.3	12,153
192.168.10.15	192.168.10.3	12,041
192.168.10.17	192.168.10.3	11,394
192.168.10.25	192.168.10.3	9,211
192.168.10.19	192.168.10.3	9,102
192.168.10.51	192.168.10.3	7,856
192.168.10.50	192.168.10.3	3,812

Table 10: Edges weight of PortScan flow

PortScan		
Src_IP	Dst_IP	Edge_Weight
172.16.0.1	192.168.10.50	158,930
192.168.10.3	192.168.10.1	17,983
192.168.10.14	192.168.10.3	6,573
192.168.10.8	192.168.10.3	6,547
192.168.10.15	192.168.10.3	5,447
192.168.10.17	192.168.10.3	3,874
192.168.10.19	192.168.10.3	3,842
192.168.10.16	192.168.10.3	3,569
192.168.10.9	192.168.10.3	3,431
192.168.10.5	192.168.10.3	3,424
192.168.10.12	192.168.10.3	2,778
192.168.10.50	192.168.10.3	1,560
192.168.10.25	192.168.10.3	520

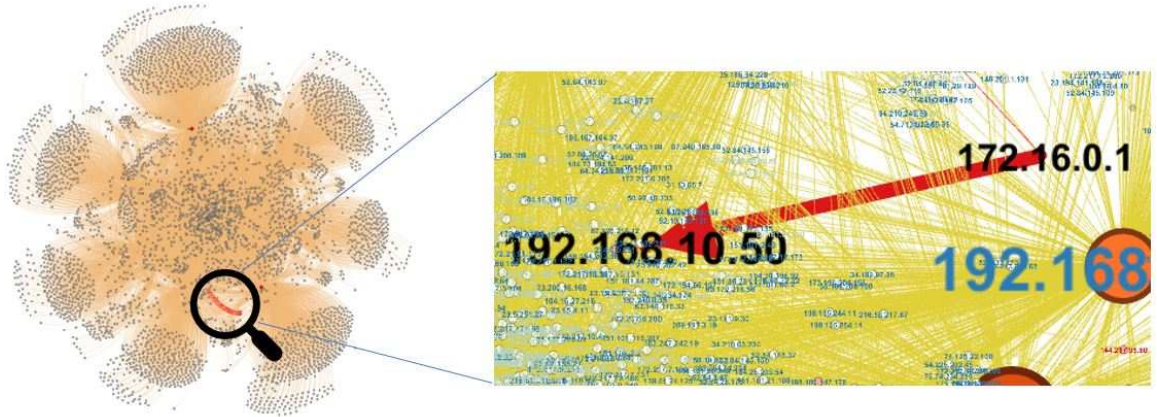


Figure 14: PortScan Graph, shows how the weight of the RED edge that targeting the victim machine's can be visible within the graph

5.4 Botnet

5.4.1 Graph Model

Botnet Ares attack [28] is a remote access tool that requires two programs to operate

- 1) *Command and Control* server (CNC): a Web interface to control agents, and
- 2) *Agent program* run on the compromised host to ensure the communication with the server. The CNC server is communicating with the agent's program at victim machines through a specific port (8080 in this scenario), thus, and since the identity of the server port is matters in detecting the CNC server. We will include the port number in creating graph topology. Therefore *SOCKET_Based* model will be applied to assist toward more in-depth analysis for the graph.

Table 11: Degree & Modularity Metrics after the occurrence of BotNet attack

Node	Indegree	Outdegree	Degree	Modularity
192.168.10.3:53	55098	6	55104	1
192.168.10.1:53	2500	0	2500	5
162.208.20.178:443	1270	1024	2294	1356
205.174.165.73:8080	1226	705	1931	1
72.21.91.29:80	754	652	1406	54
23.61.187.27:80	612	344	956	1181
192.168.10.50:22	490	464	954	1003
178.255.83.1:80	851	5	856	54

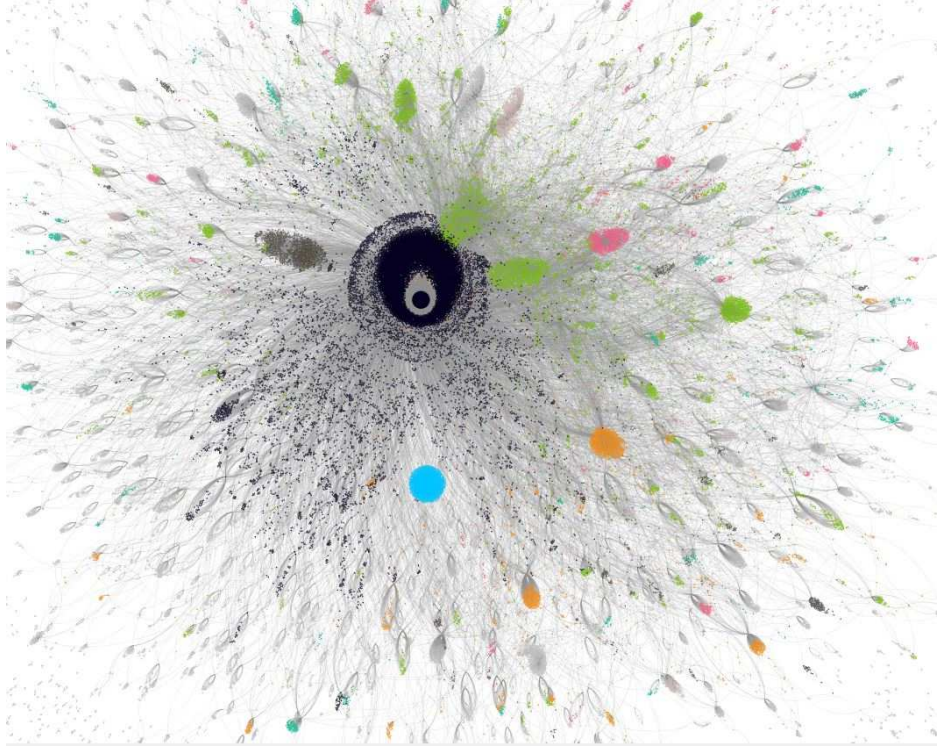


Figure 15: Botnet Graph, shows Botnet affect over the graph

5.4.2 Outcomes

191K of flow connections transformed into 98K nodes and over 130K edges. Data in Table 11 has been sort based on *Degree* values, we can see at the top the DNS & AD server, followed by node 162.208.2.178 which represents a global DNS server and that justifies the high number of communications that attached to them. Most of the other nodes have appeared earlier within the benign flows, See (Table 4 and Table 5). However, the sudden appearance of 205.174.165.73:8080 node, within the most connected nodes, indicates abnormal behavior. Due to the nature of Botnet Areas, it can be the CNC server that caused this high number of *Degree* values.

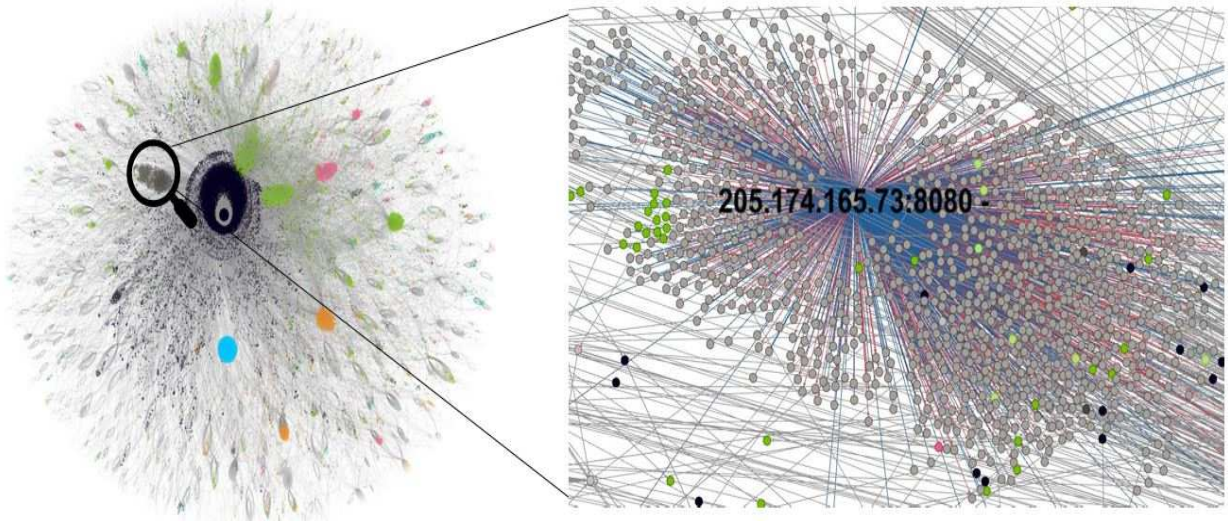


Figure 16: Botnet Graph, Shows the impact of CNC server traffic over the graph

Furthermore, Table 12 shows samples of affected nodes. We can see how the nodes with a high value of *Eccentricity* have a lower *Closeness* centrality value. Differently, *Betweenness* is maintaining a high value for most of the affected nodes. Both *Eccentricity* and *Betweenness* here are inherited from its community metrics, for instance, if a node (A) creates a connection with another node (B) in both directions, this action will transfer their criteria to each other. The furthest connected node from (A) node will high likely become the furthest connected node to the new neighbor (B). This phenomenon appeared clearly in both *Eccentricity* and *Betweenness* values of the affected nodes (Table12)

Although, the CNC server is not on top of the *Degree* table, yet it has a significant impact on the graph. Figure 15 and Figure 16 show the CNC server connections over the graph, where it has been colored base on the *Modularity class*.

Table 12: Centrality Metrics for affected nodes after the occurrence of BotNet attack

Node	Eccentricity	Closeness	Harmonic	Betweenness
205.174.165.73:8080	26	0.118575	0.166515	30649288.07
192.168.10.5:50059	25	0.132737	0.159984	17198797.98
192.168.10.5:50054	27	0.106022	0.128205	99756.31587
192.168.10.5:50730	23	0.16194	0.202268	2215163.177
192.168.10.5:52334	23	0.172465	0.217485	387409.6603
192.168.10.5:52207	23	0.172465	0.217485	387409.6603
192.168.10.5:52003	23	0.172465	0.217485	387409.6603
192.168.10.5:52336	23	0.172465	0.217485	387409.6603
192.168.10.5:52328	23	0.172465	0.217485	387409.6603

5.5 Web Attack

5.5.1 Graph Model

Web attack in this experiment is composed of three attacks, SQL injection, XSS and Brute force attack. All these attacks have been launched against the web server, similar to Botnet and DDoS the victim here is a specific port number at a specific machine, which guides us to use *SOCKET_Based* model in building the graph. About four hours of network flow with a total of 170K connections have been transformed into 87K nodes and 116K edges.

5.5.2 Outcomes

As per the list in Table 13, the attack traffics raised the *Degree* for the victim server to 3,848 edges, almost divided equally between *In-Degree* and *Out-Degree* measures, and that can be due to the type of communication those attacks performed to communicate with the web-server node. The next part of Table 13 shows how the *Betweenness* centrality of the victim node is high due to the central position that it got in linking all the attack nodes together.

Subsequently, the fact that most of the nodes appeared only to perform the attack made the centrality's (Table 14) of the involved nodes almost having similar centralities values - except for the victim node-, which helped in revealing attack trace and exposing this unusual behavior within the graph metrics. Similar to DDoS attack *Modularity* here exposed the nodes that participated in the attack and gathered them all within a mutual cluster Table 14. Figure 17 and Figure 18 show the graph after coloring its parts based on *Modularity* value.

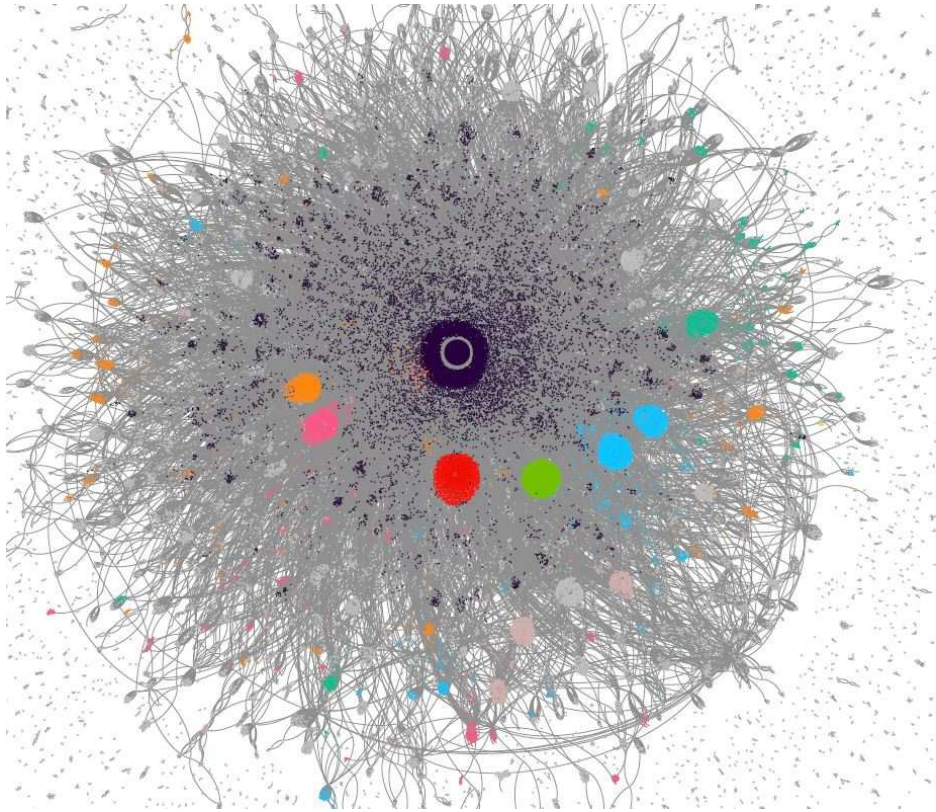


Figure 17: Web Attack Graph, shows graph components after web attack occurred

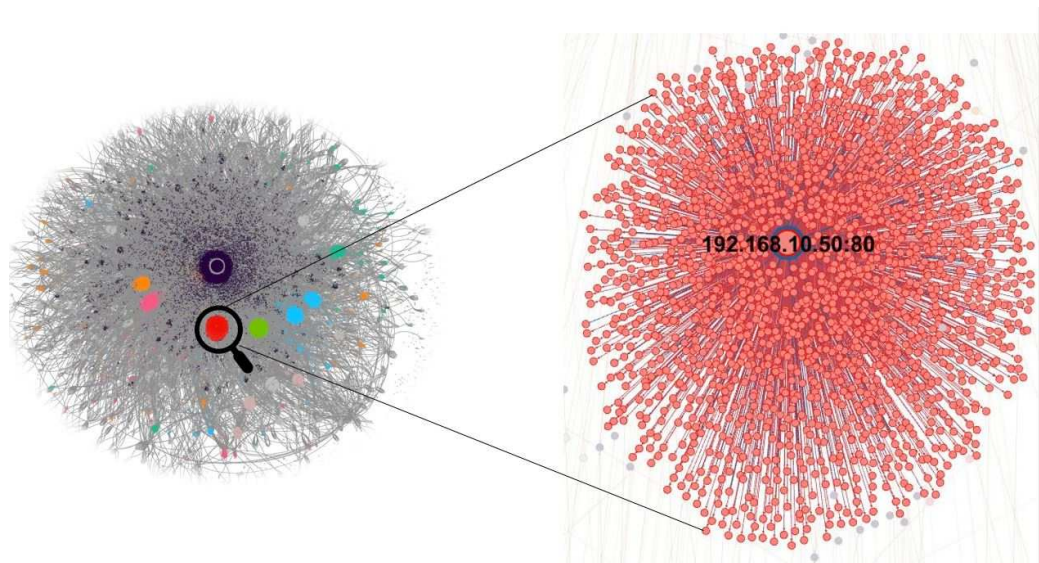


Figure 18: Web Attack Graph, shows how all nodes that participate in Webattack within the same Community, at the right site a close look to WebAttack Community

Table 13: *Centralities & Modularity* Metrics after the occurrence of WebAttack attack

Node	indegree	outdegree	Degree	Modularity	Eccentricity	Closeness	Harmonic	Betweenness
192.168.10.3:53	46180	2	46182	1	1	1	1	137510
192.168.10.50:80	2015	1833	3848	293	1	1	1	3691664
192.168.10.1:53	2500	0	2500	2	0	0	0	0
72.21.91.29:80	843	716	1559	776	17	0.187355	0.290997	35043019
23.50.75.27:80	795	432	1227	106	17	0.1646	0.246872	22860162
23.60.139.27:80	741	358	1099	47	19	0.165716	0.240061	16270231
192.168.10.50:22	507	484	991	273	21	0.142342	0.230409	14763413
178.255.83.1:80	982	8	990	47	3	0.6875	0.833333	91226.24

Table 14: Samples of affected nodes after the occurrence of WebAttack attack

Node	indegree	outdegree	Degree	Modularity	Eccentricity	Closeness	Harmonic	Betweenness
192.168.10.50:80	2015	1833	3848	293	1	1	1	3691664
172.16.0.1:44382	1	1	2	293	2	0.500136	0.50027	0
172.16.0.1:44384	1	1	2	293	2	0.500136	0.50027	0
172.16.0.1:44386	1	1	2	293	2	0.500136	0.50027	0
172.16.0.1:44388	1	1	2	293	2	0.500136	0.50027	0
172.16.0.1:44392	1	1	2	293	2	0.500136	0.50027	0
172.16.0.1:44466	1	1	2	293	2	0.500136	0.50027	0
172.16.0.1:44464	1	1	2	293	2	0.500136	0.50027	0

5.6 FTP-SSH Patator

5.6.1 Graph Model

This attack targeted the webserver node. The lunched attack executes the FTP-Patator in the morning and the SSH-Patator in the afternoon. In this case, we have two similar attacks that are targeting two specific ports (21-22), at a specific machine address (192.168.10.50). Due to the limited number of targeted ports, we build our graph in a SOCKET_Base form, where we end up with a graph that has over 184 K Nodes and 282 K edges.

5.6.2 Outcomes

We can notice from Table 15 that the victim servers are having higher Degrees values based on the massive number of connections caused by the FTP attack. However, again all the node that participates in the attack has been clustered in a common community (6). The eccentricity of affected nodes is all above the FTP server since all nodes that have a single out-degree edge to the server will inherit its paths to other nodes, therefore we can see it all the nodes in Table 15 are either 18 or 19 hops from furthest connected node.

On the other hand, closeness centrality is become lower due to the eccentricity value, where betweenness is only showing for the victim machines since most of the node doesn't have any other connection to the graph parties.

Table 15: Samples of affected nodes after the occurrence of FTP attack

Node	indegree	outdegree	Degree	Modularity	Eccentricity	Closeness	Harmonic	Betweenness
192.168.10.50:21	4489	2031	6520	6	17	0.178265	0.237834	438154117.4
192.168.10.50:22	4023	1035	5058	6	19	0.182688	0.227719	519978589.3
172.16.0.1:60132	0	1	1	6	18	0.151298	0.177857	0
172.16.0.1:60130	0	1	1	6	18	0.151298	0.177857	0
172.16.0.1:60124	0	1	1	6	18	0.151298	0.177857	0
172.16.0.1:60144	1	1	2	6	18	0.151295	0.177849	0
172.16.0.1:60120	1	1	2	6	18	0.151295	0.177849	0

5.7 Infiltration – Dropbox download

Due to the very limited number of connections that have been established to lunch the attack, it was hard to build a graph that can accurately detect the attack pattern, only 7 nodes out of 122K node actually participated in the attack within a highly connected graph with over 218K edges. Since all the attack connections were targeting a certain port (444) we implement a socket-based model (Table 16). Although these nodes are sharing some centrality values, yet the limited number of the samples reduces the efficiency of the detection.

Table 16: The affected nodes after the occurrence of Infiltration attack

Node	indegree	outdegree	Degree	Modularity	Eccentricity	Closeness	Harmonic	Betweenness
192.168.10.8:53966	0	2	2	234	2	0.6	0.666667	0
192.168.10.8:54119	0	1	1	234	1	1	1	0
192.168.10.8:54122	0	2	2	234	2	0.6	0.666667	0
192.168.10.8:54131	0	2	2	234	2	0.6	0.666667	0
192.168.10.8:54573	0	2	2	234	2	0.6	0.666667	0
192.168.10.8:1260	1	2	3	234	22	0.115454	0.12452	28830
205.174.165.73:444	35	0	35	234	0	0	0	0

CHAPTER 6

EVALUATION AND DISCUSSION

6.1 Evaluation

Three different well-known classification algorithms have performed to evaluate our approach. The main focus is to show how accurate detecting network-attacks behavior can be. The classifiers have been applied over two datasets that have been extracted from the same network flow but in dissimilar structures. In other words, both datasets are representing the same data traffics. To do so, firstly, we gathered all the Graph-Based metrics that we calculated by Gephi tool [20], afterward, we labeled all the nodes to be either affected by the attack or benign nodes, as a result, we end up with a dataset with a total of 8 Graph-Based metrics as illustrated in Table 8, the first three represented the *Degree* centrality's, following by three diameter-based centrality's, then *Betweenness* centrality and the last one was the modularity class. Secondly, the GeneratedLabelledFlows (GLF) datasets [29] which contain a collection of 80 networks statistical features for each attack, finally we compared the classification outcomes. In this evaluation, we considered three different attacks, DDoS, BotNet, and a set of three Web Attacks that have grouped in a common dataset due to its connection's similarities.

6.1.1 Preparing the Dataset

All the GLF features are pre-calculated and have been publicly published as a part of the CICIDS2017 dataset [8], yet, some incomplete cases have been removed from the datasets, we cleaned all the null and infinity values in both BotNet and WebAttack dataset, subsequently, to evaluate the importance of features, Boruta [31] has been applied, and we selected only the confirmed features. Therefore, the number of features has been reduced to 26 in GLF_DDoS, 62 for the GLF_BotNet dataset, and 61 confirmed features for GLF_WebAttack dataset. Finally, we applied SMOTE [19] to handle the imbalance samples in both BotNet and WebAttack datasets.

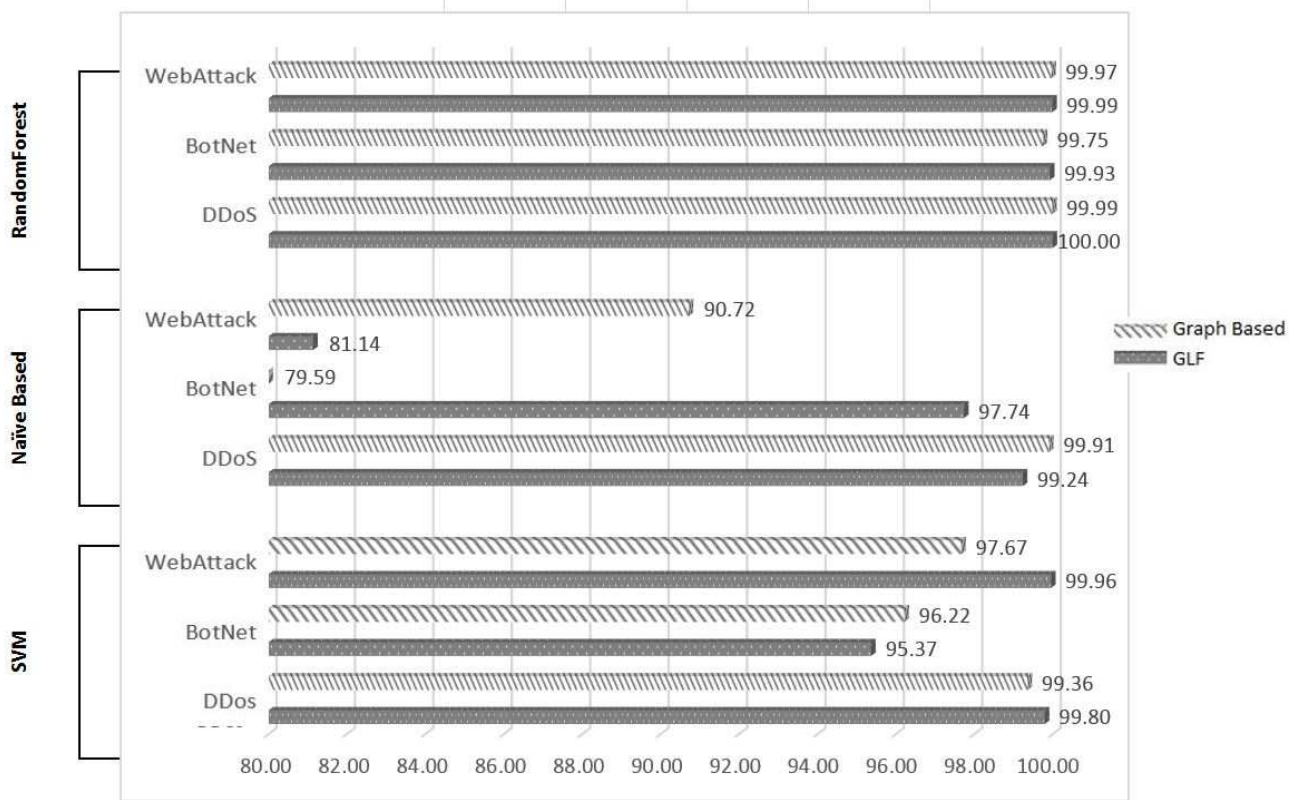


Figure 19: Evaluation of GLF and GB datasets, Average accuracy (%) per validation method for each dataset. Random Forest, Naïve Based, SVM

6.1.2 Effectiveness

To measure the effectiveness of the classifiers we considered accuracy as an evaluation metric, from Figure 19 we can notice that the accuracy that has achieved with DDoS detests are generally higher than Botnet and web attacks, and that could be correlated to the amount of impact that DDoS usually have over the network compared with others, which made it easier to be discovered, for instance, Random Forest [33] classifier reached the peak with GLF_DDoS while with the Graph-Based dataset scored a slightly less accuracy with 99.99 %, on the other hand, while maintaining very high accuracy for DDoS with Naive_Based [34] classifier, GB faced a considerable drop with Botnet datasets where the accuracy sharply decreased to 79.59% with GB dataset where it maintains around 97.47% with GLF. An interesting result has been achieved with SVM [30], with noticeable growth in the GB dataset with 96.22% accuracy in detecting botnet attack while the GLF dataset has lower accuracy of 95.37 %.

To generalize, although the GLF dataset utilized a visibly more significant number of features to achieve that accuracy, the Graph-Based datasets achieved practically the same measures with only eight features, reducing by that the time and cost of training and predicting process.

6.1.3 Performance

Although achieving high accuracy is important to be considered in designing IDS, yet, we can't neglect how critical the performance is in terms of how fast identifying unusual activities will take. To do so we counted the time that each classifier has taken for both

training and predicting process to provide a performance comparison for the same classifiers while changing the structure of the dataset, Table 17 has the average classification times in seconds. The fact that the DDoS cost more considerable amount of time can be correlated to the number of samples for each set since we handled the imbalance of datasets by applying SMOTE over BotNet and Webattack, which end-up with almost 3K sample per each class within the training set, on the other hand, DDoS dataset divided with almost over 30k sample for each class.

Table 17: AVERAGE CLASSIFICATION TIMES IN SECONDS.

Attacks	SVM		Naïve Based		Random Forest	
	GLF	GB	GLF	GB	GLF	GB
DDoS	108	396	1.16	0.33	2592	546
BotNet	4.9	1.6	1.43	0.13	11.16	6.5
WebAttack	104	0.8	0.93	0.15	10.9	1.74

However, we can notice that while Random Forest had the higher accuracy over the other classifiers, it didn't achieve same in term of speed, and in some cases like DDoS with GLF dataset, it cost over 43 minutes to complete the validation phase, yet, we can see that there is a huge decline when comparing GLF time with GB time where it takes less than 10 minutes to finish with GB. Subsequently, both BotNet and webattack were classified faster when dealing with Graph-Based datasets. Naive Based recorded as the lowest in terms of accuracy, yet, in contrast here, it succeeds to achieve the highest performance rate. In

general, comparing with other classifiers, it was clearly faster with GB than GLF in all cases. Finally, the SVM which preserve to be high for both effectiveness and performance, and again we can see clearly how superior the performance of the classifier with GB than GLF except for a single DDoS case.

Lastly, after implementing this evaluation we observed that: although the classifiers achieved better performance with GLF over the GB, however, there are two factors that we need to consider, first the number of features per each dataset: in GB and by using only 8 metrics we were able to achieve very high accuracy compared with the GLF which utilized over 60 features per each attack to accomplish this result. The second factor is the running time, from Table 17 we can notice how superior our approach was in completing the detection task which can be correlated to the number of features that need to observe the existence of an attack.

6.2 Outcome Verification

To verify the outcomes of our approach (Evaluation6.1) we re-performed the evaluation using 3 different network analysis tools. Three main types of measures have to be considered to implement our approach:

First, the directly connected neighbors (**Baseline Measures**). Therefore, we utilize *In-Degree*, *Out-Degree*, and *Degree* centralities which emphasize the number of direct-connected nodes (degree of length 1). Second, the node relation with other connected graph parties (**Diameter-Based Measures**). We explore how close or far the node is from all connected nodes. To do so, we use *Closeness*, *Harmoni Closeness*, and

Eccentricity centralities where they related to the number of hops that the node needs to pass to communicate with other nodes. Then, we determine the value of the node in connecting graph parties (**Flow-Based Measures**), and here we apply *Betweenness* centrality which computed based on the value that the edges add to the node. Finally, we discover node internal relationships within its community through *modularity* to observe how the behavior of a node can lead it to be part of a cluster with other similar nodes.

By applying that we cover the most related graph measures that explore node activities. For instance, the node that has been affected by an attack will usually have a high degree value due to the attack impact (DDos, BotNet, and webattack), which justify the need for degree metrics to observe the deviation in the direct connections metrics. This amount of connections will enhance the node chance to reach a higher amount of nodes directly, which affects the position of the node in the graph (how close and far to each node). Considering the new position of the node we need to study the impact that this new position has in connecting graph parties. And finally, observe the possibility of having similar nodes (the node that participates in the attack) within the same community based on their similar activities and relations.

To perform the verification, we import our socket to socket flow to graph-tools that mention in Table 18, then we remove the duplicate edges and replace them with edge weight, and finally, we considered only the metrics that mentioned within the 8 graph

metrics. However, since not all the tools are sharing the same set of graph metrics we reduce the number of metrics with only the one that matches. In addition to that, some tools (like Nodexl) perform some sort of normalization over the graph metrics which affect the diversity of the data, and therefore the uniqueness of node activity impact over the graph metrics, which was the main reason of having such a weak classification result for the metrics that computed by it.

On the other hand, Cytoscape and igraph package in R Language achieved much higher results with only 6 metrics, since it doesn't have Harmonic Closeness centrality nor modularity, which affect slightly the accuracy of the results. We can see that overall the metrics that computed using Gephi accomplished the higher result considering this attack (DDoS).

Table 18: Graph-based Outcomes Verification, for different Graph-based tools

Graph-Tools	# of Metrics	Attack	Accuracy		
			RF	NB	SVM
Nodexl	5	DDoS	66.92	0.4753	63.65
Cytoscape	6	DDoS	99.89	98.65	89.14
R Langugae	6	DDoS	99.91	99.84	99.92
Gephi	8	DDoS	99.99	99.91	99.36

6.3 Comparison with Existing NIDS (SNORT)

We conduct a comparison to evaluate the benefit of implementing our approach against a well-known NIDS -SNORT [50]. In our comparison, we focused on performance, scalability, dealing with alerts, and detecting unknown threats.

6.3.1 Snort

one of the most common open-source NIDS, which used to detect network cyberattacks that match a pre-defined set of rules. Each attack has its own pattern which requires a specific well-designed rule to encounter it. Considering the wide verity of known cyber-attacks, implementing a wide set of rules can dramatically drop IDS performance, where each packet needs to be compared with all enabled rules before being allowed to pass. Add to that the highly unwanted noise in generating alerts which increases the difficulty of monitoring them. This issue requires a strong understanding of the network components and a trad-balance of which threats may have a higher chance of affecting the network.

The fact that snort looks only for the pre-defined set of rules, without much concern of what's maybe considered normal and what behaves abnormally. This fact raised a real concern of how snort will act with new attacks that manly will pass the existing rules. Even Snort subscription rules are required to be up to date continually to cover all these new attacks (in case if they already have been analyzed and covered by new rules)

6.3.2 Prepare the Datasets

We ran SNORT version 2.9.111 and Graph-Based (Gephi 0.9.2 & R-3.6.2) over the same network flow, however, to implement the test we performed some changes to the data raw to be importable by each approach.

For snort, we split the PCAP file (“Friday-WorkingHours”) that contained the cyber network attack into three files (DDos.pcap, PortScan.pcap, and Botnet.pcap). Each file contains the related attack flow, then we play-back the PCAP files in snort to detect attack impact. On the other hand, for Graph-Based analysis, we extracted only the Socket-Based fields from the network PCAP file to end-up with three .csv files similar to (Table 1) which we used to create the graph and calculate the graph metrics. The fact that our approach is only requiring the IP’s and port numbers of the network flow, decreases the size of the imported files significantly.

Every record is treated differently in each approach, for instance, snort is monitoring packets fields to decide if the packet is a threat or not based on a pre-defined ruleset, where a Graph-Based analyzer considers all incoming connections. Every connection causes node metrics to change (in-degree, out-degree, etc.) affecting the centrality values of these nodes, then observe the anomaly by measuring the deviation in graph-metrics.

6.3.3 Performance

From Table 19, we can notice that overall SNORT achieved a slightly lower time against DDoS and Botnet Attacks, where Graph-Based succeed to overcome snort with a huge difference in Webattack case. In addition to that, we can observe how the node count, are not related to the number of connections but it does change based on the type of connections where all similar connections (in terms if socket-based fields) are represented by a single edge, with a higher weight.

Table 19: Performance comparison based on the processing time

Attacks	SNORT		Graph-Based	
	Time (Sec)	Packets	Time (Sec)	Node\edges
DDoS	1126.76	1.5 M	1322	47K\93K
BotNet	688.84	1.1 M	780	41K\79K
WebAttack	5449.36	4.35 M	18.7	7K\13.5K

6.4 Discussion

After conducting these experiments in terms of both accuracy and performance we can see how effective our approach is in detecting abnormal network activity, and here we will go in detail to analyze the result of the conducted experiments and to emphasize more in each point, as illustrated in Table 20.

Table 20: The criteria of the conducted comparison

IDS	SNORT	Graph-Based
Threat Identification	Rules Matching	Deviation in Centrality Measurements
Scalability	Do not scale well in a play-back analysis (single thread and single instance) Applying a wide number of rules can drawback the performance	Can scale up to handle millions of nodes and edges
Performance	Single-Threaded, Single Instance	Multi-Threaded
Unknown attacks	Need to design rules for it, and require frequent updates	Can be detected by observing its behavior
Monitoring\log	Complicated (log file)	Centrality values + visualizing graph

6.4.1 Threat Identification

Considering the possibility of identifying threats, SNORT was very efficient in detecting all known attacks, where each attack has its unique criteria that have been used to distinguish it from normal packets. However, since the identification process is relying on rules matching, there is a chance for any new attacks or any updated version fo attack to bypass this defense, not to mention the complexity of designing new rules to detect these

attacks. Add to that, the difficulty of rules management to decide which rules can catch the attack better since enabling all rules can affect the performance dramatically.

On the other hand, the Graph-Based approach works almost similarly with most of the targeted attacks, the fact that it relays on centrality and modularity measures makes it more adaptable. The high sensitivity of centrality measures can clearly assist in detecting the anomaly patterns that each attack can make. We can see from Figure 19 that the result of using Graph measures has achieved very promising results, considering that all this data has been created from the only Socket to Socket information.

6.4.2 Performance & Scalability

Our focus in this experiment was on the offline analysis, due to that the amount of data that need to be analyzed was huge, the number of resources that are required to investigate 2 or 4 hours of traffics, packet by packet and connection by connection can create a real challenge to any NIDS, in term of performance and accuracy.

The mechanism of SNORT is designed to go through all the traffics that exists within the .pcap file packet by packet, which makes this approach get slow considering the vast amount of captured packets. Moreover, it might require considering most of the packet components (over 80 metrics extracted from .pcap file for GLF dataset), meanwhile, the graph-based analysis was required only 8 metrics to detect the existence of the attack,

which reduces the need for both memory and storage significantly. Therefore, enhance the scalability of the new approach.

In addition to that, the possibility of implementing parallelize processing in computing centrality metrics can enhance the performance even more. However, in this study, we used only a single thread, to have a real comparison between both approaches.

6.4.3 Unknown attacks

One of the most critical aspects of our compersion is how the NIDS will act with unknown attacks. For the SNORT NIDS, the law is simple if the attack is known and a rule has been made to detect it, then the attack is high likely to be detected, otherwise, - and here we refer to new attacks-, the attack may bypass the defense without been detected or logged.

For the Graph-Based approach, things are different. From chapter four it was possible to find the deviation that all the attacks have made, except for *Infiltration* attack where the amount of connection that participates in the attack was very few compared with the logged file, which made its effect too small to be detected. However, the outcome of the classification results from Figure 19, proved that its possible to detect the attack by monitoring only its effect over only 8 graph-based metrics, with a very promising accuracy value.

6.4.4 Monitoring

By monitoring, we aim to study how complicated is to understand the log files or the Graph-Based metrics, to extract valuable information that can lead to spot attack trace. To simulate the fact of an unknown attack we ran snort with the community rules. After running the SNORT over the .pcap files, we end up with 11GB log files, that we need to explore to spot the exact attacks trace, the logs for a single attack like BotNet was 3 files with a total of 450 MB.

On the other hand, the .csv files that carry the graph metrics were readable, meaningful and all the attacks files did not exceed 100 MB, the BotNet graph metric file was about 38 MB, this fact help in getting the required result faster. Finally, the graph metrics can assist in visualizing the graph in a way that can provide an emphasis looks to the network activity, as shown in all graphs that showed in the experiment chapter.

6.5 Comparison with Existing Graph-Based Intrusion Detection Method

Utilizing graph statistical node properties to spot anomaly within the graph, has been performed by Chaudhary, Mittal, and Arora [51]. In their work, they transfer the datasets to an undirected graph. Thereafter, they compute three graph statistical measures (Betweenness, Closeness, and Degree centralities), they defined anomaly node as the node that either connected to very few or very large percents of nodes. They focused on detecting the second type. Although their approach was capable of detect anomaly under their definition, yet, the fact that they treat the data as an undirected graph eliminates the

possibility of going deeper with their analysis. For instance, the three-degree values (in, out and all) will not be applied over the undirected graph, add to that the limited number of the metrics that they used which can lead to false positives. Table 21 shows the result of a comparison with the mentioned method, where we built all the graphs as a directed graph.

Table 21: Graph-Based the result of the comparison with other Graph-Base Method

Methods	# of Metrics	Dataset	Node\edges	Type of Graph	SVM Accuracy %	NB Accuracy %	RF Accuracy %
Neural Networks	3	GB-DDoS	47K\93K	Directed	87.17	47.73	99.89
Graph-Based	8	GB-DDoS	47K\93K	Directed	99.36	99.91	99.99
Neural Networks	3	GB-BotNet	41K\79K	Directed	80.53	73.3	99.53
Graph-Based	8	GB-BotNet	41K\79K	Directed	96.22	79.59	99.75
Neural Networks	3	GB-WebAttack	7K\13.5K	Directed	60.91	91.23	83.32
Graph-Based	8	GB-WebAttack	7K\13.5K	Directed	97.67	90.72	99.97

CHAPTER 7

CONCLUSIONS

Accurately detecting threats within network traffic while maintaining performance aspects held has always been considered as a real challenge for Network Intrusion Detection Systems. The tremendous amount of traffic that goes continuously through the network is hard to monitor. In addition to the wide variety of existing attacks that keep evolving, new attacks keep showing, which rise a real concern about the capability of NIDS to provide accurate results in discovering all these cyber-threats traces. Moreover, the need for logging and monitoring network events which can create a significant noise caused by all the false-positive alerts.

To overcome this challenge, we managed in this thesis to accomplish promising results in both performance and accuracy by applying Graph-Based anomaly detection. Our approach focused on how each entity within the network is connected, how these connections affect node metrics, and the impact of these connections over other direct or indirect connected nodes. By exploring these criteria of the nodes that participate in the attack -either as attacker or victim-, we succeed to detect abnormal activities, by monitor the deviation in graph metrics.

To implement Graph-Based anomaly detection we transformed the wide mess of network traffic into Graph-Based structure. Then, we built the graph by using only source and destination information from each connection. Thereafter, we compute graph-metrics, for

both nodes and edges. Then, we observed the deviation before and after the occurrence of the attacks.

The importance of a node can be determined based on the position it occupied or the influence that it has over other nodes. Therefore, we explore these facts to detect not only the victim nodes but also the attack paths. To do so, we utilized three types of *Centrality* measures. First, the *Baseline Measures*: concentrate on the node direct connections. Come after, the *Diameter-Based Measures*: which concerned about the distance that the node needs to pass to communicate with other nodes to measure how close or far the node is from other connected nodes. After that, *Flow-Based Measures*: which computes based on the information that each edge has, or the importance of the nodes in connecting graph parties. Finally, we implemented modularity over each graph to group similar nodes in a common cluster.

The computed metrics have been employed to discover abnormality within the network flow. Over six different types of attacks, our approach was capable of detecting attacks impact in five of them, whereas the last one had a very insignificant amount of connections comparing with the benign flow in its dataset. In addition to that, *modularity* class as a community identifier assisted in discovering DDoS, web, and FTP attacks.

Furthermore, to evaluate our approach three well-known machine learning algorithms have been applied to measure both effectiveness and performance aspects. We compared the results of graph metric evaluation with the result of GLF datasets. From the result, we

proved the efficiency of our approach, Graph-Based structure of data achieved a very high-performance rate against the GLF evaluation results. We showed that how only eight graph metrics can clearly identify the attack trace and successfully managed to expose the anomaly behavior of the attack using its graph metrics.

Finally, we conduct two comparisons between our approach and other existing approaches, the first one with a well-known NIDS (SNORT). The fact that our approach is relying on the deviation in centrality measurements can make it more hopeful in detecting unknown attacks, in addition to that it can scale to handle millions of nodes and edges smoothly. Moreover, we can even reduce the processing time by implement multithreaded processing in computing graph-based metrics, which can enhance the performance of our approach. And finally, the visualization option gives the data a meaning full shape comparing with the huge amount of log files that are hard to be analyzed. The results of this comparison show how our approach achieved a very promising result even against signature-based NIDS, considering only eight graph-metrics. Finally, we compared our approach with an exciting Graph-Based analysis method, our approach success to accomplish better results, in terms of accuracy.

In the end, in this thesis, we proved that our proposed approach verified the possibility of effectively detecting network cyber-attacks behaviors using only graph-metrics (centralities and modularity measurements).

7.1 Summary of Contributions

- We succeed to detect Network attacks abnormal activities, by monitor the deviation in graph metrics.
- We accomplished promising results in both performance and accuracy.
- We succeeded to observe a unique pattern for almost all of the network attacks that have been applied by combining three types of Centrality measurements.
- Over six different types of attacks, our approach was capable of detecting attacks impact in five of them.
- We evaluated our approach using three well-known machine learning algorithms to assure maintaining both effectiveness and performance aspects. The proposed approach achieved a very promising result.
- Two more comparisons have been conducted against existing approaches, NIDS (SNORT), and existence Graph-Based method, where our approach proved its efficacy in terms of accuracy and performance.

7.2 Future Work

Graph-based Analysis is a very promising trend in improving anomaly detection. Many additional aspects can be considered to enhance our approach. We can consider adding different graph-metrics to expand our understanding of the network flow. Performance enhancement can be achieved by applying the current approach with multi-threaded or parallel processing methods. Study the graph metrics of the nodes that share the same modularity values. And many other trends that can provide an alternate way of observing the flow, which can raise the efficiency of detecting cyber-attacks behavior.

References

- 1- Neuman, B.C. and Ts'o, T., 1994. Kerberos: An authentication service for computer networks. IEEE Communications Magazine (pp.33-38).
- 2- Hagberg, A., Lemons, N., Kent, A. and Neil, J., 2014. Connected components and credential hopping in authentication graphs. In Signal-Image Technology and Internet-Based Systems (SITIS), 2014 Tenth International Conference on (pp. 416-423).
- 3- Neil, J., Uphoff, B., Hash, C. and Storlie, C., 2013. Towards improved detection of attackers in computer networks: New edges, fast updating, and host agents. In Resilient Control Systems (ISRCS), 2013 6th International Symposium on (pp. 218-224).
- 4- Grana, J., Wolpert, D., Neil, J., Xie, D., Bhattacharya, T. and Bent, R., 2016. A likelihood ratio anomaly detector for identifying within-perimeter computer network attacks. Journal of Network and Computer Applications, 66, pp.166-179.
- 5- Kent, A.D. and Liebrock, L.M., 2013. Differentiating user authentication graphs. In Security and Privacy Workshops (SPW), 2013 IEEE (pp. 72-75).
- 6- Ammann, P., Wijesekera, D. and Kaushik, S., 2002. Scalable, graph-based network vulnerability analysis. In Proceedings of the 9th ACM Conference on Computer and Communications Security (pp. 217-224).
- 7- van Erven, G.C., Holanda, M., and Carvalho, R.N., 2017. Detecting Evidence of Fraud in the Brazilian Government Using Graph Databases. In World Conference on Information Systems and Technologies (pp. 464-473).
- 8- Sharafaldin, I., Lashkari, A.H. and Ghorbani, A.A., 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In ICISSP (pp. 108-116).. <https://csr.lanl.gov/data/auth/>
- 9- Sundaram, A., 1996. An introduction to intrusion detection. Crossroads, (pp.37).
- 10- Grandjean, M., 2016. A social network analysis of Twitter: Mapping the digital humanities community. Cogent Arts & Humanities,(p.1171458).

- 11- Morgado, C., Baioco, G.B., Basso, T. and Moraes, R., 2018. A Security Model for Access Control in Graph-Oriented Databases. In 2018 IEEE International Conference on Software Quality, Reliability and Security (QRS) (pp. 135-142).
- 12- Ashraf, N., Ahmad, W. and Ashraf, R., 2018. A Comparative Study of Data Mining Algorithms for High Detection Rate in Intrusion Detection System. Annals of Emerging Technologies in Computing (AETiC).
- 13- Aleroud, A. and Karabatis, G., 2013. Toward zero-day attack identification using linear data transformation techniques. In 2013 IEEE 7th International Conference on Software Security and Reliability (pp. 159-168).
- 14- Han, J., Haihong, E., Le, G. and Du, J., 2011. Survey on NoSQL database. In 2011 6th international conference on pervasive computing and applications (pp. 363-366).
- 15- Portnoy, L., 2000. Intrusion detection with unlabeled data using clustering (Doctoral dissertation, Columbia University).
- 16- Alazab, M., Venkatraman, S., Watters, P. and Alazab, M., 2011. Zero-day malware detection based on supervised learning algorithms of API call signatures. In Proceedings of the Ninth Australasian Data Mining Conference-Volume 121 (pp. 171-182). Australian Computer Society.
- 17- Dimopoulos, G., Barlet-Ros, P., Dovrolis, C. and Leontiadis, I., 2017. Detecting network performance anomalies with contextual Anomaly detection. In Measurement and Networking (M&N).
- 18- Mdini, M., Blanc, A., Simon, G., Barotin, J. and Lecoeuvre, J., 2017. Monitoring the network monitoring system: Anomaly Detection using pattern recognition. In Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on (pp. 983-986).
- 19- Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P., 2002. SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, (pp.321-357).
- 20- BASTIAN, Mathieu, HEYMANN, Sebastien, et JACOMY, Mathieu. Gephi: an open source software for exploring and manipulating networks. In: Third international AAAI conference on weblogs and social media. 2009.

- 21- Akoglu, L., Tong, H. and Koutra, D., 2015. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery*, (pp.626-688).
- 22- HEYMANN, Sébastien et LE GRAND, Bénédicte. Visual analysis of complex networks for business intelligence with gephi. In: 2013 17th International Conference on Information Visualisation. IEEE, 2013. (p. 307-312).
- 23- HUSSAIN, S., Muhammad, L.J. and YAKUBU, A., 2018. Mining Social Media and DBpedia Data Using Gephi and R. *Journal of Applied Computer Science & Mathematics*, 12(1), (pp.14-20).
- 24- Yannick Rochat. Closeness centrality extended to unconnected graphs: The harmonic centrality index (PDF). *Applications of Social Network Analysis, ASNA 2009*.
- 25- Freeman, L.C., 1977. A set of measures of centrality based on betweenness. *Sociometry*, (pp.35-41).
- 26- Kelly, L.M., Athanassiou, N. and Crittenden, W.F., 2000. Founder centrality and strategic behavior in the family-owned firm. *Entrepreneurship Theory and Practice*, (pp.27-42).
- 27- Cherven, K., 2015. Mastering Gephi network visualization. Packt Publishing Ltd.
- 28- Botnet ares, 2017 (accessed December 3, 2018). [On-line]. Available: <https://github.com/sweetsoftware/Ares>.
- 29- A. H. L. Iman Sharafaldin and A. A. Ghorbani, Ci-cids2017 datasets, 2017 (accessed December 10, 2018).[Online]. Available: <https://iscxdownloads.cs.unb.ca/iscxdownloads/CIC-IDS-2017/#CI-IDS-2017>.
- 30- J. A. Suykens and J. Vandewalle, “Least squares support vector machine classifiers,” *Neural processing letters*, (pp. 293–300)
- 31- Boruta v6.00, 2018 (accessed February 10, 2019). [Online]. Available: <https://www.rdocumentation.org/packages/Boruta>

- 32- Yang, J., Cheng, C., Shen, S. and Yang, S., 2017. Comparison of complex network analysis software: Citespace, SCI 2 and Gephi. In 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA), (pp. 169-172).
- 33- Ho, T.K., 1995. Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition, (Vol. 1, pp. 278-282).
- 34- Maron, M.E., 1961. Automatic indexing: an experimental inquiry. Journal of the ACM (JACM), (pp.404-417).
- 35- Eberle, W. and Holder, L., 2007. Discovering structural anomalies in Graph-Based data. In Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007), (pp. 393-398).
- 36- Noble, C.C. and Cook, D.J., 2003. Graph-Based anomaly detection. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, (pp. 631-636).
- 37- Akoglu, L., Tong, H. and Koutra, D., 2015. Graph based anomaly detection and description: a survey. Data mining and knowledge discovery, (pp.626-688).
- 38- Shah, N., Beutel, A., Hooi, B., Akoglu, L., Gunnemann, S., Makhija, D., Kumar, M. and Faloutsos, C., 2016. Edgecentric: Anomaly detection in edge-attributed networks. In 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW) (pp. 327-334).
- 39- Gamachchi, A., Sun, L. and Boztas, S., 2018. A graph-based framework for malicious insider threat detection. arXiv preprint arXiv:1809.00141.
- 40- Chen, Z., Hendrix, W. and Samatova, N.F., 2012. Community-based anomaly detection in evolutionary networks. Journal of Intelligent Information Systems, 39(1), (pp.59-85).
- 41- Mathew, A.B. and Kumar, S.M., 2014. An efficient index-based query handling model for Neo4j. IJCST, 3(2), (pp.12-18).
- 42- Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y. and Wilkins, D., 2010. A comparison of a graph database and a relational database: a data provenance perspective. In Proceedings of the 48th annual Southeast regional conference (p. 42).

- 43- Neo4j manual. [Online]. Available: <https://neo4j.com/docs/ogm-manual/current/introduction/>
- 44- Ho, T.K., 1995. Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition (Vol. 1, pp. 278-282).
- 45- Arash Habibi Lashkari, Gerard Draper-Gil, Mohammad Saiful Islam Mamun and Ali A. Ghorbani, "Characterization of Tor Traffic Using Time-Based Features", In the proceedings of the 3rd International Conference on Information System Security and Privacy, SCITEPRESS, Porto, Portugal, 2017
- 46- "Facebook Announces Its Third Pillar "Graph Search" That Gives You Answers, Not Links Like Google". TechCrunch. 2013. Retrieved 16 January 2013.
- 47- Batra, S. and Tyagi, C., 2012. Comparative analysis of relational and graph databases. International Journal of Soft Computing and Engineering (IJSCE), (pp.509-512).
- 48- Gudivada, V.N., Baeza-Yates, R. and Raghavan, V.V., 2015. Big data: Promises and problems. Computer, (pp.20-23).
- 49- "Graph Databases, 2nd Edition". O'Reilly | Safari. Retrieved 2018-10-23.
- 50- Roesch, M., 1999. Snort: Lightweight intrusion detection for networks. In Lisa (Vol. 99, No. 1, pp. 229-238).
- 51- Chaudhary, A., Mittal, H. and Arora, A., 2019. Anomaly Detection using Graph Neural Networks. In 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon) (pp. 346-350).
- 52- Saito, R., Smoot, M.E., Ono, K., Ruscheinski, J., Wang, P.L., Lotia, S., Pico, A.R., Bader, G.D. and Ideker, T., 2012. A travel guide to Cytoscape plugins. Nature methods, (pp.1069).
- 53- Smith, M.A., Shneiderman, B., Milic-Frayling, N., Mendes Rodrigues, E., Barash, V., Dunne, C., Capone, T., Perer, A. and Gleave, E., 2009. Analyzing (social media) networks with NodeXL. In Proceedings of the fourth international conference on Communities and technologies, (pp. 255-264).

- 54- Csardi, G. and Nepusz, T., 2006. The igraph software package for complex network research. InterJournal, complex systems, (pp.1-9).
- 55- Cline, M.S., Smoot, M., Cerami, E., Kuchinsky, A., Landys, N., Workman, C., Christmas, R., Avila-Campilo, I., Creech, M., Gross, B. and Hanspers, K., 2007. Integration of biological networks and gene expression data using Cytoscape. Nature protocols, (p.2366).

Vitae

Name :Ahmed Mohammed Salih El-Nazir

Nationality :Sudani

Date of Birth :10/27/1989

Email :Ahmed.alnzeer@gmail.com

Address :KSA- Al Khobar Al Janubiyah, Al Khobar ZIP 34622

Academic Background : Have a Bachelor degree in “Computer Systems and Networks” at Sudan University of Science and Technology in 2011, then joined to master program in “Security and Information Assurance” at King Fahd University of Petroleum and Minerals after 4 years of work experiences. Managed with the team Dr. Farage Azzedin and Mr. Dijamaluddin to publish a work in addressing new cyber protection method under the name of "Web Deception towards Moving Target Defense", which has been cited at International Carnahan Conference on Security Technology (ICCST) 2018.