INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

VISIBILITY GRAPHS TO SUPPORT ML FEATURE SELECTION AND DETECT DOS CYBERATTACKS

João Manuel Ferreira Lopes

Escola Superior de Tecnologia e Gestão

Instituto Politécnico
de Viana do Castelo

Visibility Graphs to Support ML Feature Selection and Detect DoS cyberattacks

Autor(a)

João Manuel Ferreira Lopes

Trabalho efetuado sob a supervisão de

Professor António Alberto dos Santos Pinto

Professor Pedro Filipe Cruz Pinto

Mestrado em Cibersegurança

18 de Dezembro de 2023

Mestrado em Cibersegurança
Master in Cybersecurity

Visibility Graphs to Support ML Feature Selection and Detect DoS cyberattacks

a master's thesis authored by

João Manuel Ferreira Lopes

supervised by

António Pinto
Professor Coordenador com Agregação, IPP

Pedro Pinto
Professor Adjunto, IPVC

and assisted by

Alberto Partida
Data, Complex Networks and Cybersecurity Sciences Technological Institute, Rey Juan Carlos University

This thesis was submitted in partial fulfilment of the requirements for the Master's degree in Cybersecurity at the Instituto Politécnico de Viana do Castelo

ipvc

18 of December, 2023

Abstract

The world economy depends on information systems. Business value resides in the data stored in information technology (IT) systems and the processes performed with that data. However, malicious parties target these IT systems to extract value from them using different *modus operandi*. Denial of Service (DoS) attacks are a common and harmful method of making Internet-connected IT systems and, consequently, the business processes running on them unavailable. Cybersecurity researchers from industry and academia are looking for early warning detection systems to quickly detect and be able to mitigate these DoS attacks. This thesis proposes two new ways to mitigate this problem, the first by supporting the selection of the best features to be used through Machine Learning (ML) methods to detect DoS/DDoS attacks, and the second consists of early detection of DoS/DDoS attacks based on the information provided by visibility graphs. From the results obtained, it can be seen that VGs are capable of selecting the best features and that they can also be used to detect DoS/DDoS attacks. These results can also be extracted through the analysis of the patterns obtained and the analysis of the respective degree and density values.

Keywords: Denial of Service. Visibility Graphs. Machine Learning. Cybersecurity. Early detection.

Resumo

A economia mundial depende de sistemas de informação. O valor comercial reside nos dados armazenados nos sistemas de tecnologia da informação (TI) e nos processos executados com esses dados. No entanto, partes mal-intencionadas visam estes sistemas de TI para extrair valor deles usando diferentes *modus operandi*. Os ataques de negação de serviço (DoS) são um método comum e prejudicial para tornar indisponíveis os sistemas de TI conectados à Internet e, consequentemente, os processos de negócios executados neles. Investigadores de cibersegurança, industriais e académicos, estão em busca de sistemas de detecção de alerta precoce para detectar rapidamente e serem capazes de mitigar esses ataques DoS. Esta tese propõe duas novas formas de mitigar este problema, a primeira apoiando a seleção das melhores features para serem usadas através de métodos de Machine Learning (ML) para detetar ataques DoS/DDoS e a segunda consiste em detetar precocemente ataques DoS/DDoS com base nas informações fornecidas pelos grafos de visibilidade (VGs). Dos resultados obtidos, verifica-se que os VGs podem ser capazes de selecionar as melhores features e que também podem ser utilizados na deteção de ataques DoS/DDoS. Estes resultados também podem ser extraídos através da análise dos padrões obtidos e da análise dos respectivos valores de graus e densidade.

Palavras-chave: Negação de Serviço . Grafos de Visibilidade. Machine Learning. Cibersegurança.

Aknowledgements

First, I would like to thank Prof. António Pinto and Prof. Pedro Pinto for their wonderful guidance and support throughout this work. They were undoubtedly two essential pillars along this path that made me always want more and better. I also want to thank Alberto Partida for his enormous help, for his knowledge, for the valuable suggestions that allowed me to overcome the difficulties that appeared, and for the encouragement given during the gray days.

I would like to thank the Escola Superior de Tecnologia e Gestão (ESTG) of Instituto Politécnico de Viana do Castelo (IPVC) for providing me with the necessary conditions to carry out this work and my master's colleagues with whom I spent time discussing ideas.

I would also like to thank my professional colleagues at Swogo, the 1WorldSync company, who have always encouraged me and provided the ideal conditions to pursue this goal. Special thanks to my managers Pedro Moreira and José Silva for giving me the freedom to complete this work and trusting me.

Finally, I would like to thank my parents and my brother for their endless support, my family and close friends for their support and understanding during this journey, and especially my girlfriend who has been my great support and source of energy throughout this journey.

Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

ADELE Automatic Machine Learning and Anomaly Detection

AI Artificial Intelligence

CPM Change Point Monitoring

DDoS Distributed Denial-of-Service

DL Deep Learning

DNS Domain Name System

DoS Denial-of-Service

HTTP Hypertext Transfer Protocol

HVG Horizontal Visibility Graph

ICMP Internet Control Message Protocol

IDS Intrusion Detection Systems

IPS Intrusion Prevention Systems

LIME Local Interpretable Model-agnostic explanations

MCA Multivariate Correlation Analysis

MDA Mean Decrease Accuracy

ML Machine Learning

NTP Network time protocol

SDN Software-defined networking

SHAP SHapley Additive exPlanations

SIEM Security Information and Event Management

SL Supervised Learning

TCP Transmission Control Protocol

UDP User Datagram Protocol

VG Visibility Graph

XAI Explainable Artificial Intelligence

Chapter 1

Introduction

Technology has a huge impact on today's global economy, it significantly streamlines operations and reduces operational costs for companies, making those that reduce technological investment also reduce their income significantly [50, 53, 19]. As with all the good things that come with technology, there is also a downside, cyberattacks. These, when successful, not only result in financial losses but also cause damage to the reputation of the affected companies [31, 32]. In recent years, society has witnessed an increase in the number and sophistication of cyberattacks [45]. The constant evolution of digital technologies and the emergence of new threats have challenged the security of systems and data around the world. These threats can range from Denial-of-Service (DoS) attacks to intrusions into corporate networks, resulting in theft of confidential information and violations of privacy [78]. Cybersecurity has therefore become a top concern for governments, businesses, and individuals [63, 87]. The financial and reputational costs associated with cyberattacks prompt organizations to invest in more advanced prevention and detection measures. In addition, protecting user data and preserving the integrity of systems has become imperative for trust in digital technologies [50, 53, 19]. Cyberattacks are constantly evolving and increasingly complex, and the large amount of data generated by systems makes the analysis and identification of suspicious patterns a complex task. Thus, the ever-changing threat landscape of cyberattacks requires fast and effective solutions to detect and prevent cyberattacks.

In this context, cyberattack detection research has been an active area of study [97, 41]. Several methods have been proposed to improve the early and accurate identification

of cyber threats [39] and, the application of complex network analysis methods, such as Visibility Graphs (VGs), emerges as a promising approach for detecting cyberattacks. This method can offer a new perspective in analyzing network traffic patterns and identifying malicious behavior.

1.1 Problem Statement and Motivation

The detection of cyberattacks is a challenging task. The techniques used by attackers are constantly evolving, seeking to exploit vulnerabilities in systems and bypass existing defenses. Attacks can also occur at multiple layers, from network infrastructure to applications and services, requiring a holistic approach to detection [75, 93].

In addition, the large amount of data generated in network environments and distributed systems makes it difficult to identify suspicious patterns. Traditional analysis methods may not be enough to deal with the high dimensionality and complexity of this data. As a result, false positives or negatives can occur, impairing the effectiveness of detection systems.

The motivation for the current work is to use VGs to detect cyberattacks since they can represent complex time series and identify patterns. This approach can help reduce data complexity and highlight anomalous behavior more efficiently. The use of complex network analysis methods can offer a more comprehensive view of network traffic patterns, allowing the detection of malicious activities that can go unnoticed in traditional methods.

Furthermore, the application of VGs in the detection of cyberattacks can contribute to the advancement of knowledge in the field of cybersecurity. Adopting an innovative approach and exploring new techniques can lead to valuable insights and pave the way for the development of more robust and effective systems for protecting against cyber threats.

1.2 Objectives

The main objective of this thesis is to explore and apply VGs as a way of effectively detecting cyberattacks. To achieve this goal, the following specific objectives are pursued:

- **Analysis of VGs in the Detection of Computer Attacks:** Conduct a compre-

hensive review of the literature on VGs. Understand their theoretical foundations, including their application in other areas and how they can be used to analyze network traffic and detect attack patterns.

- **Analysis of Cyberattack Detection:** Conduct a literature review on cybersecurity and methods of detecting computer attacks.
- **Develop a method for detecting cyberattacks through the use of VGs:** Propose and develop a detailed implementation for the application of VGs in detecting cyberattacks through the identification of patterns, developing a pipeline capable of collecting and processing network traffic data, generating VGs.

1.3 Contributions

This thesis provides two contributions:

- A new way to identify the best features of a network package to apply Machine Learning (ML) to detect distributed or non-distributed denial of service attacks.
- A new way to detect DoS attacks through the use of VGs.

These contributions resulted in the following two publications and a patent:

- João Lopes, Pedro Pinto, Alberto Partida, and António Pinto, "Using VGs for Feature Selection in Supervised Machine Learning Applied to Detect Distributed Denial-of-Service (DDoS) Attacks", in Proceedings of the SASYR Conference, Barcelos, Portugal, July 2023. **Best research work award**.
- João Lopes, Alberto Partida, Pedro Pinto, and António Pinto, "On the Use of VGs for Feature Selection in Supervised Machine Learning - A use case to detect distributed DoS attacks", in Proceedings of the OL2A Conference, Ponta Delgada, Portugal, September 2023.
- **National Patent** pending with the title "Detecting DoS attacks through the use of VGs"

1.4 Organization

The structure of this thesis is as follows. Chapter 2 analyzes the background related to cyberattacks, mainly DoS and DDoS attacks, discusses cyberattack detection systems, and explains the usefulness and potential of VGs in the real world. Chapter 3 reviews work related to feature selection and ways to detect DoS and DDoS attacks. In Chapter 4 VGs are used to find the best features to apply in ML to detect DDoS attacks. In Chapter 5 VGs and Horizontal Visibility Graphs (HVGs) are used to detect DDoS attacks through patterns. Finally, Chapter 6 concludes the thesis and discusses future work.

Chapter 2

Background

This Chapter presents the background to the work carried out. First, a set of cyberattacks are presented, mainly DoS attacks and their subtypes. Then, different cyberattack detection strategies are detailed and VGs, their capabilities, and current applications are also discussed.

2.1 Cyberattacks

It is widely recognized that cybersecurity is a challenge in today's digital age. Several cyberattacks occur daily, often with significant impact on businesses and organizations [34, 15]. As examples, the 2017 cyber attack on Equifax, which put the confidential data of millions of people at risk [89] and the 2020 attack on SolarWinds which demonstrated the vulnerability of software vendors to cyber attacks [52]. These incidents highlight the urgency of robust cyberattack detection solutions.

DoS and DDoS are among the most common cyberattacks since these types of attacks are of low execution complexity [96]. A few studies show that, in some regions of the world, DoS attacks account for the majority of network traffic [20]. Companies such as DynDNS have been the target of DDoS attacks, resulting in service interruptions and financial losses for large online platforms such as GitHub, Spotify, and Twitter, among others [16]. The anatomies of DoS and DDoS attacks are depicted in Figure 2.1, with the DoS attack aiming to overload a system or network and make it inaccessible to legitimate users, disrupting systems, causing slowdowns, and compromising operational security. In

addition, a DDoS attack coordinates a large number of distributed devices to achieve the same goal.

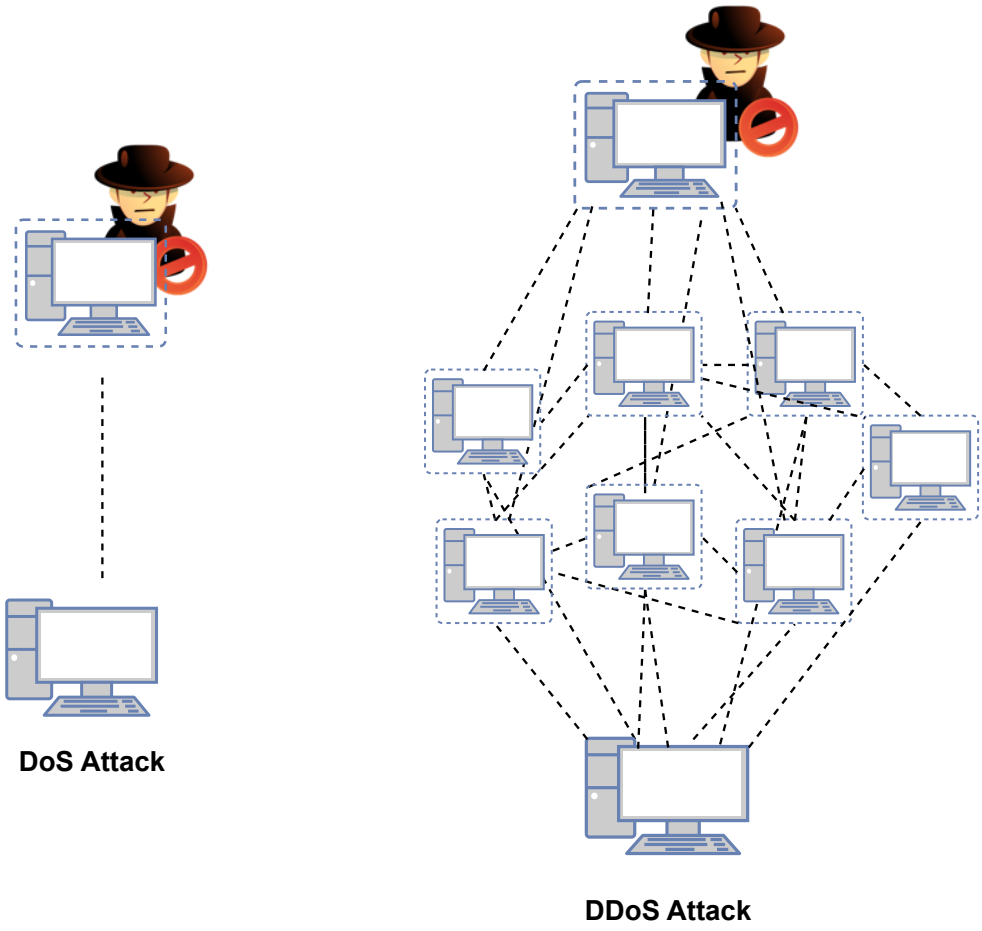

Figure 2.1: The anatomies of a DoS attack (left) and a DDoS attack (right)

DoS and DDoS attacks have characteristic patterns, such as sudden spikes in traffic, repeated requests from the same IP address, and increased load on infrastructure, which can facilitate detection [86]. These patterns are typically recorded in the logs of the attacked systems that are put in place to record all system activity. Logs are valuable information for detecting attacks and can provide important clues for identifying threats. They record information about network events, authentication, user access, and other activities, allowing administrators and security systems to monitor and analyze suspicious events.

In DoS and DDoS cyberattacks, several types of attacks can be carried out. DoS attacks can be a traffic flood attack, an Internet Control Message Protocol (ICMP)/Ping

attack, or a SYN flood attack, among others.

In a traffic flood attack is usually carried out from multiple sources (it could be a DDoS attack), which makes it difficult to block the traffic. This behavior is depicted in Figure 2.2. In this type of attack, the attacker sends a large amount of network traffic to the target, overloading its resources and making it unreachable for legitimate traffic.

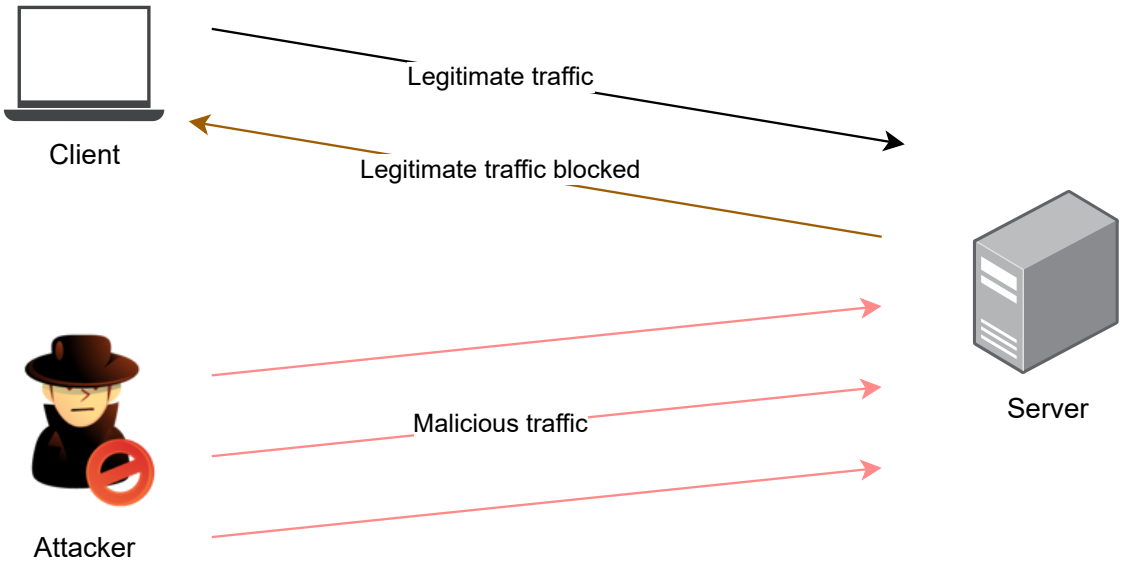

Figure 2.2: Traffic flood attack

In an ICMP/Ping attack, the attacker uses the ICMP to send a flood of echo requests to the target. This behavior is depicted in Figure 2.3. This type of attack consumes bandwidth and resources and can also be carried out via botnets or compromised devices, as described later in this section. Sometimes, the attacker uses random IP source addresses in the ICMP messages.

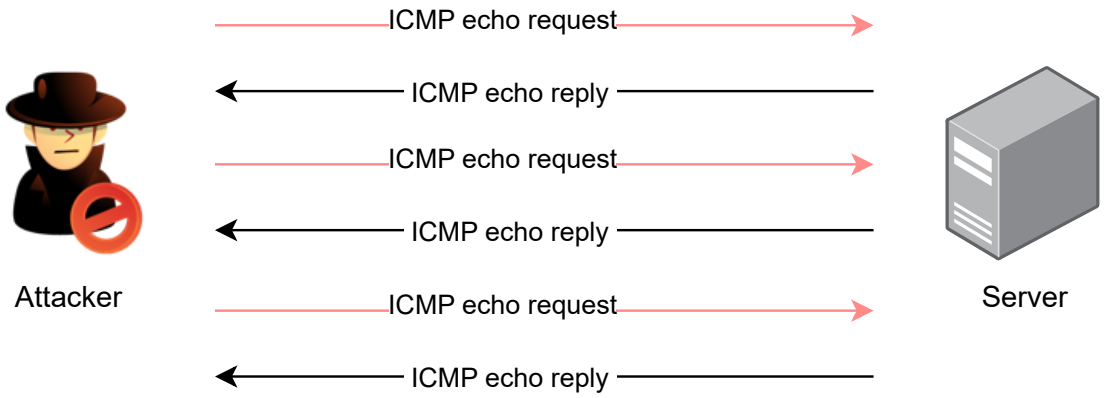

Figure 2.3: ICMP/Ping attack

In a SYN flood attack, the attacker sends a large number of SYN requests to establish a Transmission Control Protocol (TCP) connection with the target server. This behavior is depicted in Figure 2.4. In this type of attack, the attacker does not complete the connection process, exhausting the server's resources. Sometimes, the attacker uses random IP source addresses in the SYN requests.

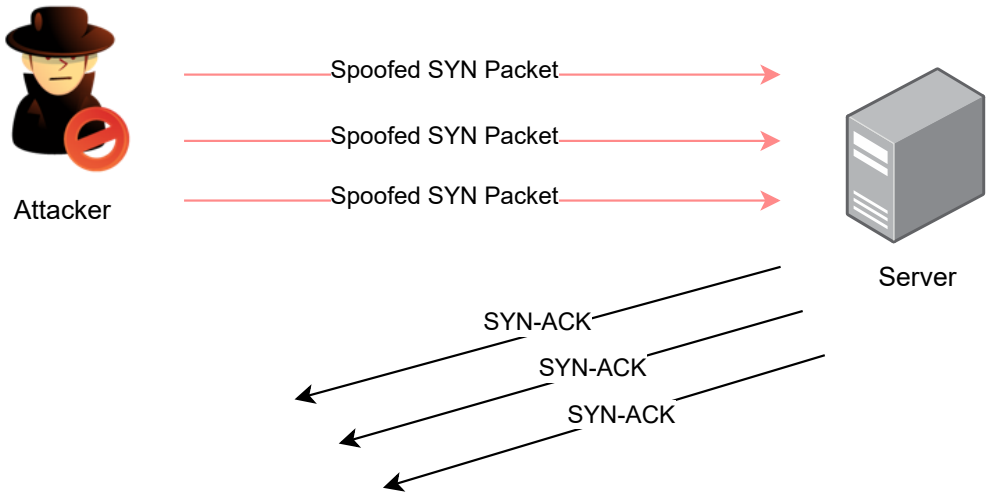

Figure 2.4: SYN flood attack

Similarly, there are other types of DDoS attacks such as botnet attacks, amplification attacks, reflection attacks, NTP/UDP reflection attacks, among others.

In a botnet attack, the attacker controls a large network of compromised devices (botnets). This behavior is depicted in Figure 2.5, where the attacker uses botnets to send malicious traffic to the target. This technique is scalable and difficult to mitigate because it involves a large number of sources.

In a Reflection Attack, the attacker sends spoofed requests to public servers that have its source address and the target's IP address. This behavior is depicted in Figure 2.6. The public servers then respond to the target with a large amount of data, increasing the intensity of the attack. This type of attack is often carried out through Domain Name System (DNS) servers or other publicly accessible servers.

An amplification attack is a specific type of reflection attack where the response generated by the target system is significantly larger than the initial request made by the attacker. This happens when the attacker uses misconfigured servers or services that unintentionally amplify targeted traffic. In other words, instead of sending a large number

of requests directly to the website, which security measures can easily block, this attack uses a poorly configured server that responds to spoofed requests. This type of attack is often carried out through DNS servers or other publicly accessible servers.

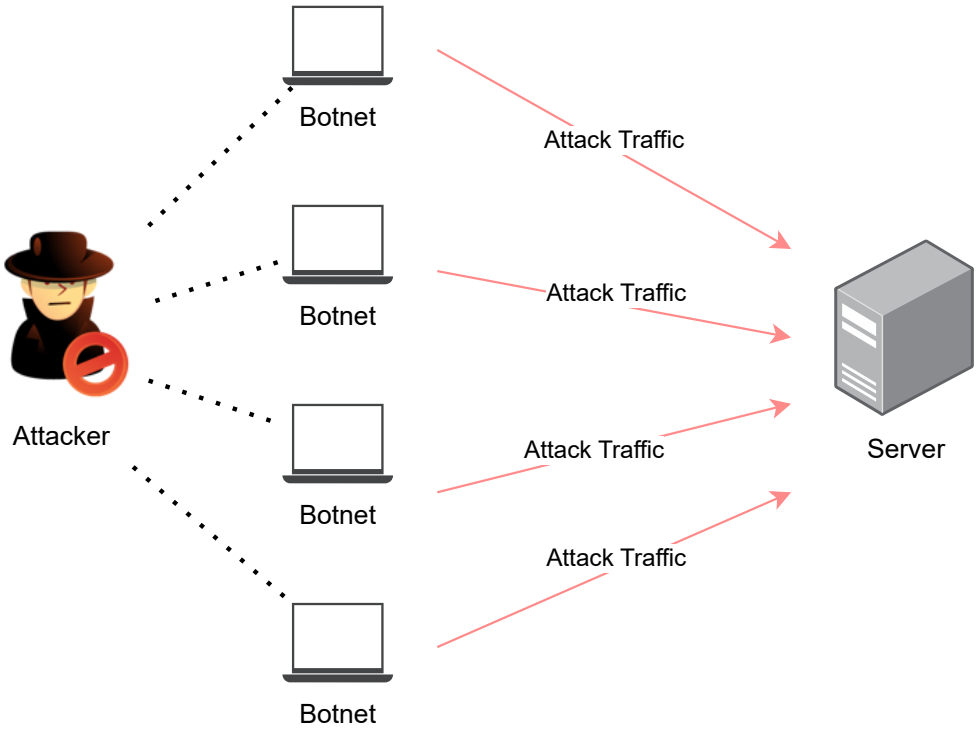

Figure 2.5: Botnet attack

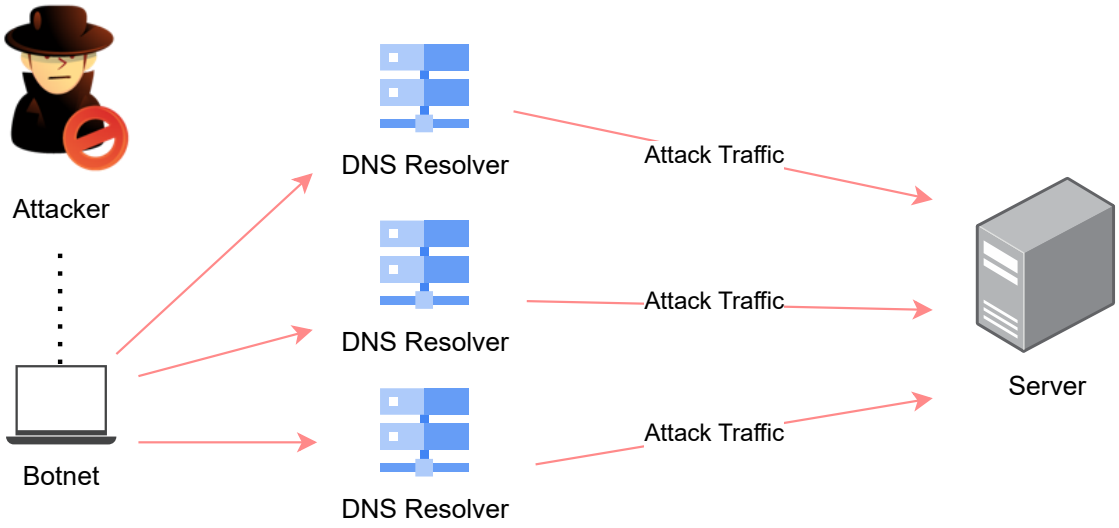

Figure 2.6: Reflection/Amplification attack

In an Network time protocol (NTP)/User Datagram Protocol (UDP) reflection attack,

the attacker uses NTP or UDP servers to amplify traffic. This type of attack is similar to the reflection attack, but instead of using public servers, the attacker uses NTP or UDP servers to amplify traffic. This behavior is depicted in Figure 2.7.

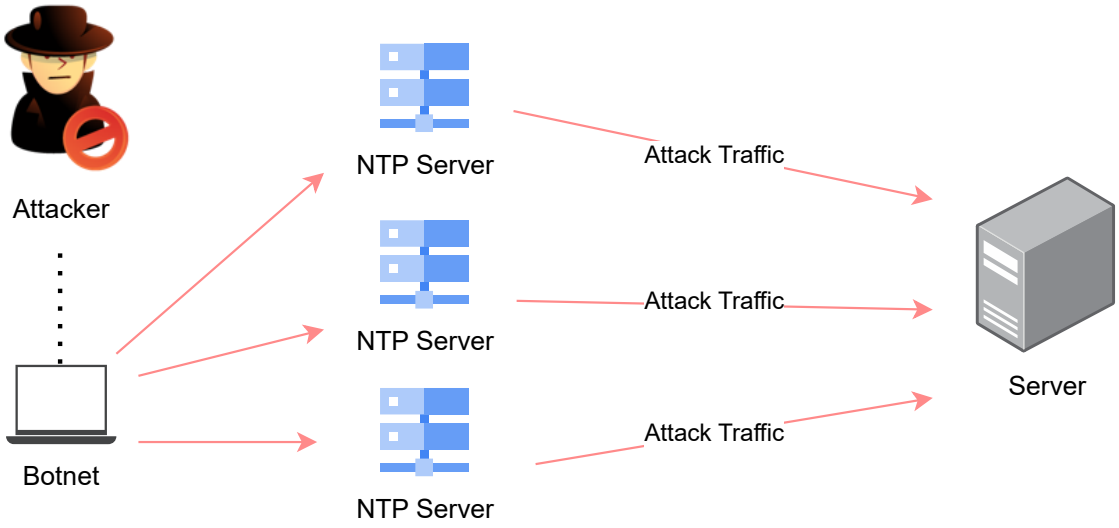

Figure 2.7: NTP/UDP reflection attack

Another type of attack that can be used to attack a web server is the Hypertext Transfer Protocol (HTTP) DoS attack, in which the server is flooded with legitimate requests until it cannot handle them. This can be done using a botnet. This behavior is depicted in Figure 2.8.

Another type of attack that targets a web server is the resource exhaustion attack. This involves exploiting vulnerabilities in server applications or systems to exhaust their resources, such as memory and processing power. Once the server's resources are exhausted, it becomes inaccessible to legitimate users.

The Slowloris attack is an attack in which multiple HTTP connections to the target server are kept open [72]. This way, the attacker can consume all available connection slots and make it impossible for legitimate users to connect to the server. This behavior is depicted in Figure 2.9.

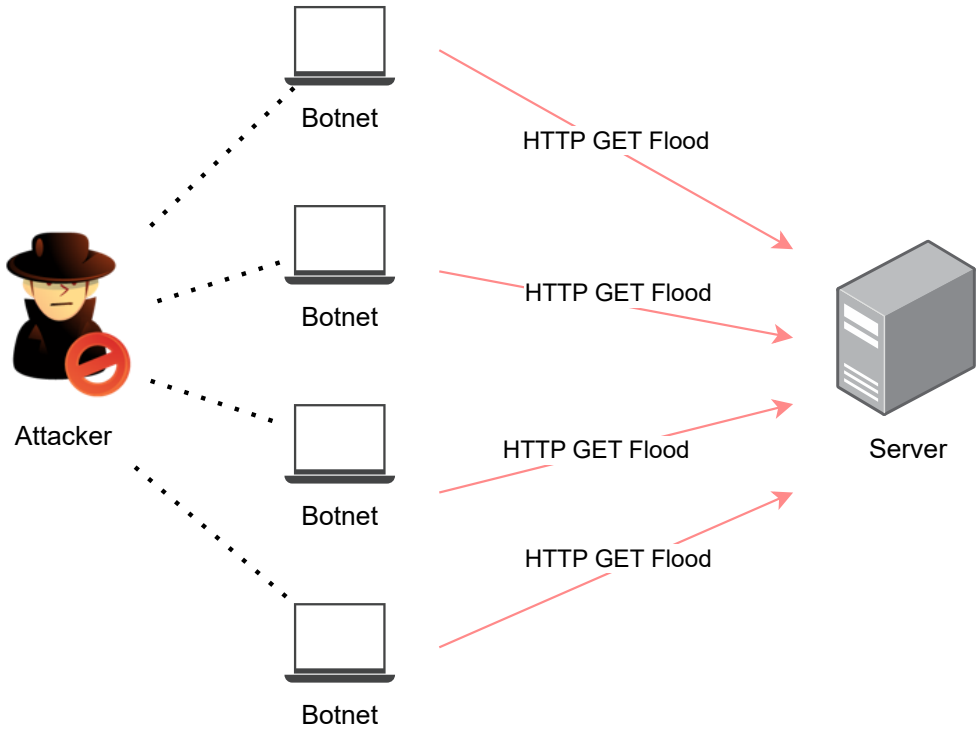

Figure 2.8: HTTP DoS attack

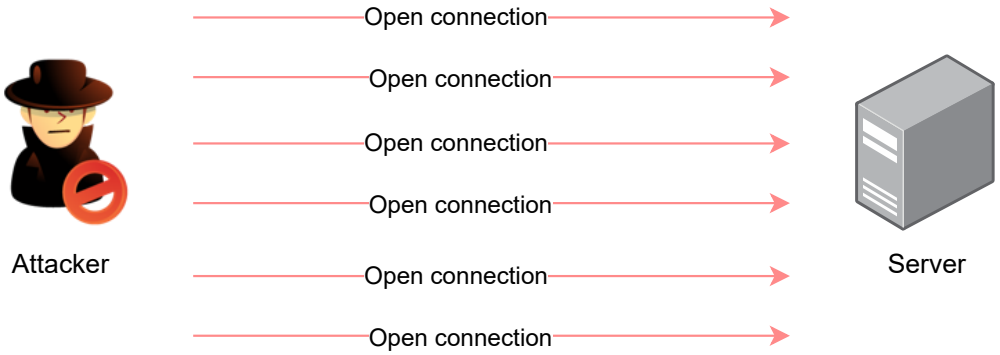

Figure 2.9: Slowloris Attack

2.2 Countermeasures to Cyberattacks

Cyberattacks have recently become a major threat to organizations of all sizes [10]. Cybercriminals are constantly developing new methods to attack corporate security systems and cause damage that is often irreversible [77]. They can steal sensitive data, disrupt business operations, and even demand ransom payments in exchange for restoring access.

To combat these threats, cyberattack detection tools have evolved significantly over the years. These tools have become increasingly sophisticated and effective in detecting

and preventing cyberattacks. They are now an essential part of any organization's security strategy. Some of the notable tools on the market include Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), and Firewalls.

IDS is a monitoring and analysis tool used to detect malicious activity in network traffic (see Figure 2.10). It is designed to detect unusual behavior or patterns in network traffic that could indicate a potential cyber attack. As soon as a threat is detected, the IDS system sends an alert to the security team, who can then take appropriate action to prevent the attack from being successful. IDS can also provide valuable information about the source and nature of the attack, which can help organizations improve their security defenses and prevent future attacks [6].

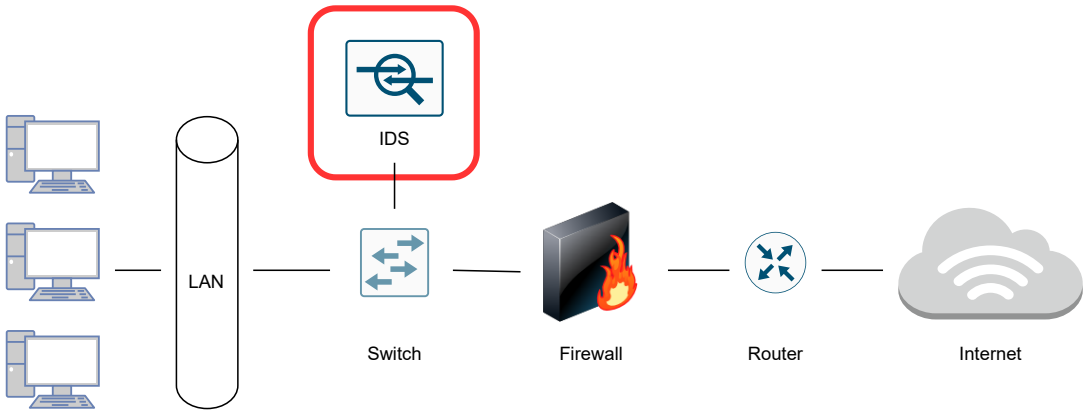

Figure 2.10: Placement of the IDS in a network

IPS, on the other hand, can be seen as an extension of the IDS, which not only detects threats but also takes measures to contain them (see Figure 2.11). It can automatically block or isolate any suspicious activity detected in network traffic. The IPS system is highly configurable, allowing organizations to define rules and policies that determine what actions should be taken when a threat is detected. This can include blocking traffic from specific IP addresses or ports, or even quarantining infected devices to prevent the spread of malware [90]. It is also necessary to consider the difference between IPS and Host-based IPS, as the latter run on a single client or server, monitor and respond to specific events. What differs from an IPS is that it is based on the network [83].

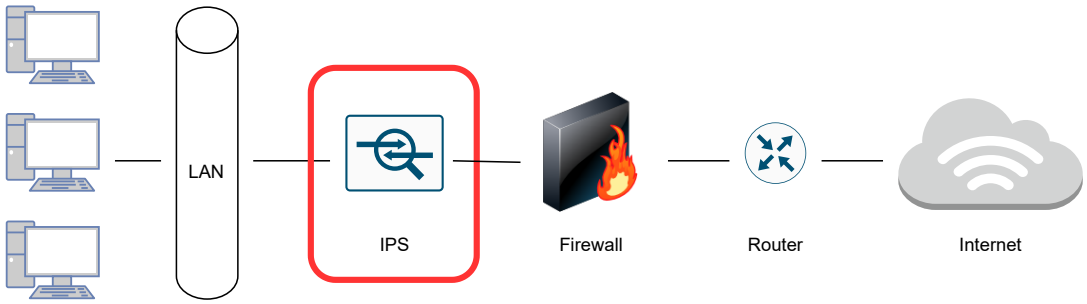

Figure 2.11: Placement of the IPS in a network

Firewalls are another essential tool for organizations looking to protect their networks against cyberattacks. A firewall acts as a barrier between a private internal network and the public Internet, filtering all incoming and outgoing traffic (see Figure 2.12). It can block traffic that does not meet certain criteria, such as traffic from known malicious IP addresses. Firewalls can also be configured to allow or block traffic based on specific protocols or applications [9].

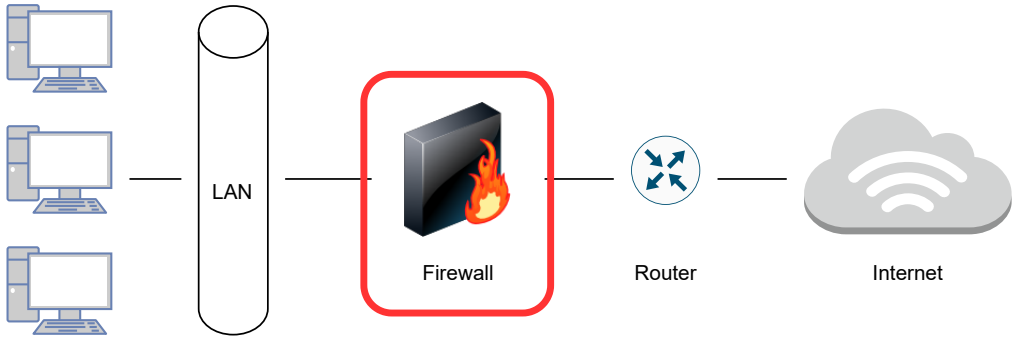

Figure 2.12: Placement of the Firewall in a network

Although all three systems are used to secure a network, there are differences between them. Firewalls are mainly used to perform actions such as blocking and filtering traffic, while IDS and IPS are used to detect, warn, and/or prevent a cyberattack, depending on their configuration. Table 2.1 presents the differences between the three systems in detail.

Table 2.1: Firewall versus IDS versus IPS

	Firewall	**IDS**	**IPS**
Purpose	Filters incoming and outgoing traffic, taking into account the configured rules.	Monitors a network service to detect malicious activity and send a warning.	Inspects, detects, classifies and blocks traffic when attacks are detected.
Function	Filters traffic based on port and IP address.	Detects traffic in real time, looks for traffic with suspicious patterns or behavior, and generates alerts.	Detects and inspects traffic in real time, looks for attack patterns or behaviors, and prevents cyberattacks in real time.
Setting and Placement	Typically on layer 3 and on the network line. (Fig.2.12)	Typically outside the network line. (Fig.2.10)	Typically after the firewall and on the network line. (Fig.2.11)
Patterns	Does not analyze	Analyzes	Analyzes

2.3 Visibility Graphs

VGs and HVGs are graphical representations of visibility interactions between points. They capture direct visibility relationships between points, where one point is visible to another if no obstacles are obstructing the direct line of sight between the two points [37]. Each point that forms the VG represents a value, and this value is characterized by the height of these points. The connection between two points is made by a straight line, called a ray, which connects the visible points in the time series [42, 44].

The creation of a VG includes the following steps:

1. **Data collection:** First, it is necessary to collect data. This may include network communication records, timestamps, and device information. Depending on the use

case and context in which a VG is created, it is important to have a dataset with data that can be analyzed.

2. **Identification of the feature to be analyzed:** A dataset consists of several features and in this step, it is necessary to determine for which feature a VG should be created.

3. **Handling of the data:** Most datasets contain data or empty rows that must be eliminated, as the goal is to keep the dataset as clean as possible so that the results are also as clean and optimized as possible. In this step, the dataset must be filtered so that only the feature selected in the previous step remains, the empty rows of the dataset must be eliminated. Data must be converted to numeric values if necessary since the VGs are characterized only by numeric values. Finally, a list of values of the selected and cleaned feature of the dataset must be created.

4. **Identify the edges that connect the visible points:** In this step, ordered pairs of values (x, y) are created for the values that are visible to each other. The goal is to identify the edges that are present in the VG.

5. **Visibility Graph Construction:** In this step, the ordered pairs of values (x, y) defined in the previous step are used and the points and the corresponding edges connecting them in the graph are drawn.

As a result of these steps, the VG is built and can be presented according to Figure 2.13.

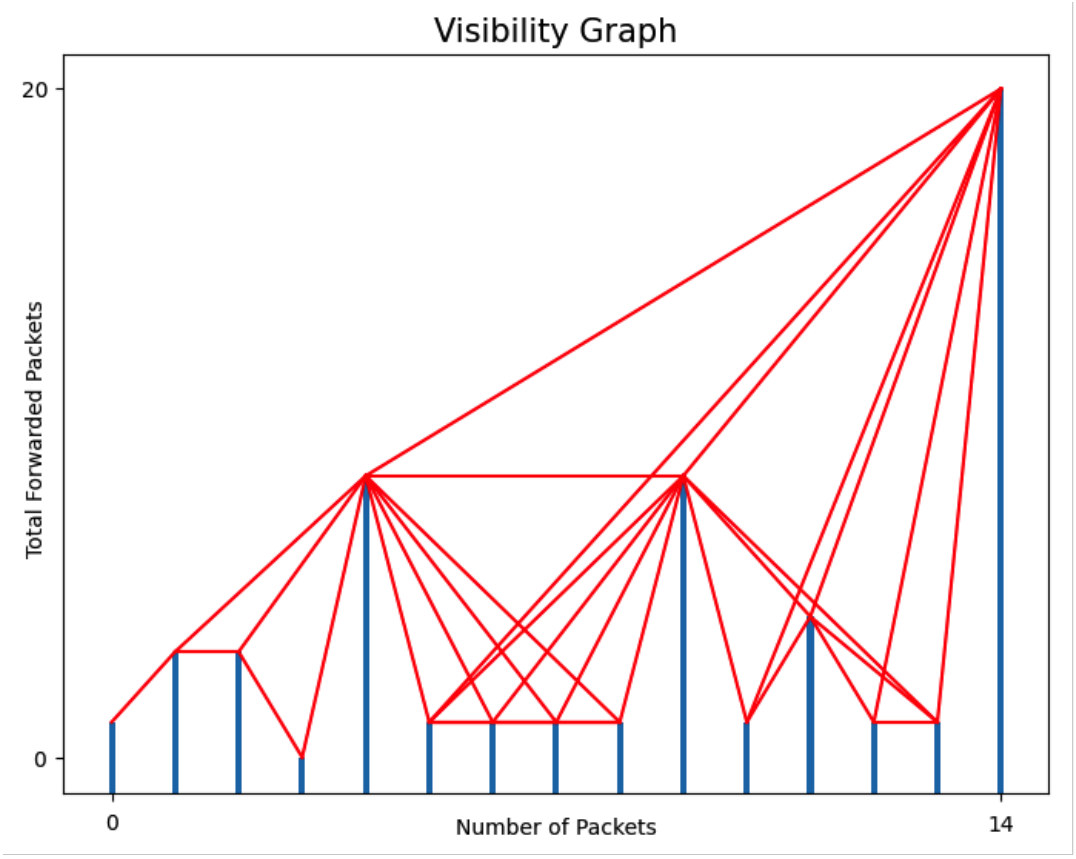

Figure 2.13: Example of a VG

VGs and HVGs have several capabilities that make them applicable in various fields. They are a very versatile tool that has countless applications in various areas. One of the most important capabilities of VGs is the ability to analyze temporal patterns [3]. VGs represent a time series of events and facilitate the detection of emergent patterns and anomalous behavior [94]. By analyzing visibility relationships between different points, VGs can help identify underlying patterns and trends that may not be apparent at first glance. Anomaly detection is another area where VGs excel. Anomaly detection is about identifying unusual or suspicious behavior that deviates significantly from the norm [17]. VGs can identify these anomalous behaviors [94, 47], by highlighting visibility relationships between different points that are not typical or expected. This can help identify potential security threats, fraud, or other abnormal behaviors that require further investigation. VGs also provide the advantage of dimensionality reduction. By transforming complex time series into simple graphical structures, VGs can reduce the complexity of data and

facilitate their interpretation. This is especially useful in areas such as finance, where large amounts of data can be overwhelming and difficult to analyze [51]. VGs provide a visual representation of data that is easy to understand and interpret, making it easier to spot trends and patterns. Finally, VGs are useful for visualizing data. They provide an intuitive visual representation of patterns and interactions between points, making results easier to interpret. This is especially useful in areas such as medicine [84], where complex data can be difficult to understand. VGs can provide a clear and concise representation of data that is easy to understand and interpret, facilitating informed decision-making. As technology continues to advance, VGs have the potential to become an increasingly important tool for data analysis and visualization.

VGs are a powerful tool used in many areas of the modern world. They help simplify and understand complex data sets, making it easier for researchers and professionals to draw meaningful conclusions and make informed decisions.

One area where VGs have proven particularly useful is finance. In this field, VGs are used to analyze time series in the stock market, allowing investors to track historical market events and make predictions about future trends [99]. For example, VGs have been used to analyze gold prices to help investors identify patterns and make informed decisions about the right time to buy or sell [46].

In biology, VGs are used to study the behavior of neurons, the dynamics of recurrent neural networks [12], and to decipher brain states [98]. This tool has proven important in helping researchers understand complex neural processes and develop new treatments for neurological disorders.

Science is another area where VGs have proven useful. Weather stations, for example, use VGs to analyze wind speeds [65], and other weather patterns to help predict storms and other severe weather events. In Italy, VGs have been used to analyze chronic obstructive pulmonary disease, which is allegedly related to pollution in the northeast of the country [5].

In the health sector, VGs have been used to analyze the spread of infectious diseases in different geographic regions. By analyzing data on the number of new cases, hospitalizations, and deaths [61], researchers can make predictions about future outbreaks and develop strategies to contain them. During the COVID-19 pandemic, VGs were used in

Greece to analyze the impact of COVID-19 control measures and help policymakers make informed decisions about public health policies [84].

VGs can also be applied in the blockchain field. As blockchain technology becomes more prevalent, it is important to understand the risks associated with various implementations. VGs can be used to evaluate blockchain implementations by analyzing intentional risks characterized by market volatility [64].

While VGs are a valuable tool for analyzing time series data, also HVGs offer unique benefits in more specific cases. HVGs, share many of the same potential applications as VGs but are particularly useful when the direction of events is irrelevant. One of the main differences between VGs and HVGs is how the edges are defined. In VGs, the edges are directed and connect each point in the time series to the next point in the sequence. This creates a linear structure that can be used to analyze trends, patterns, and other features of the data. In contrast, HVGs have bidirectional edges connecting each point to all previous and subsequent points that are visible from that same point. Since edges in HVGs do not have a specific direction, they represent mutual visibility between points rather than a specific causal relationship. This unique characteristic of HVGs makes them particularly suitable for analyzing time series data where the direction of events is not a primary concern. For example, HVGs can be used to identify network structures or patterns in data from a complex system like the stock market ([85]). In these cases, the emphasis is on identifying relationships between data points rather than understanding the causal mechanisms that drive these relationships.

Graphically, as can be seen in Figure 2.14, HVGs are identical to VGs. Only the connection between the nodes is made in both directions.

In summary, VGs and HVGs are powerful tools that can be used in many areas of the modern world. From finance to biology, science, health, and blockchain, VGs can help researchers and professionals make informed decisions and draw meaningful conclusions from complex data sets. As technology continues to advance, VGs and HVGs are likely to become even more important.

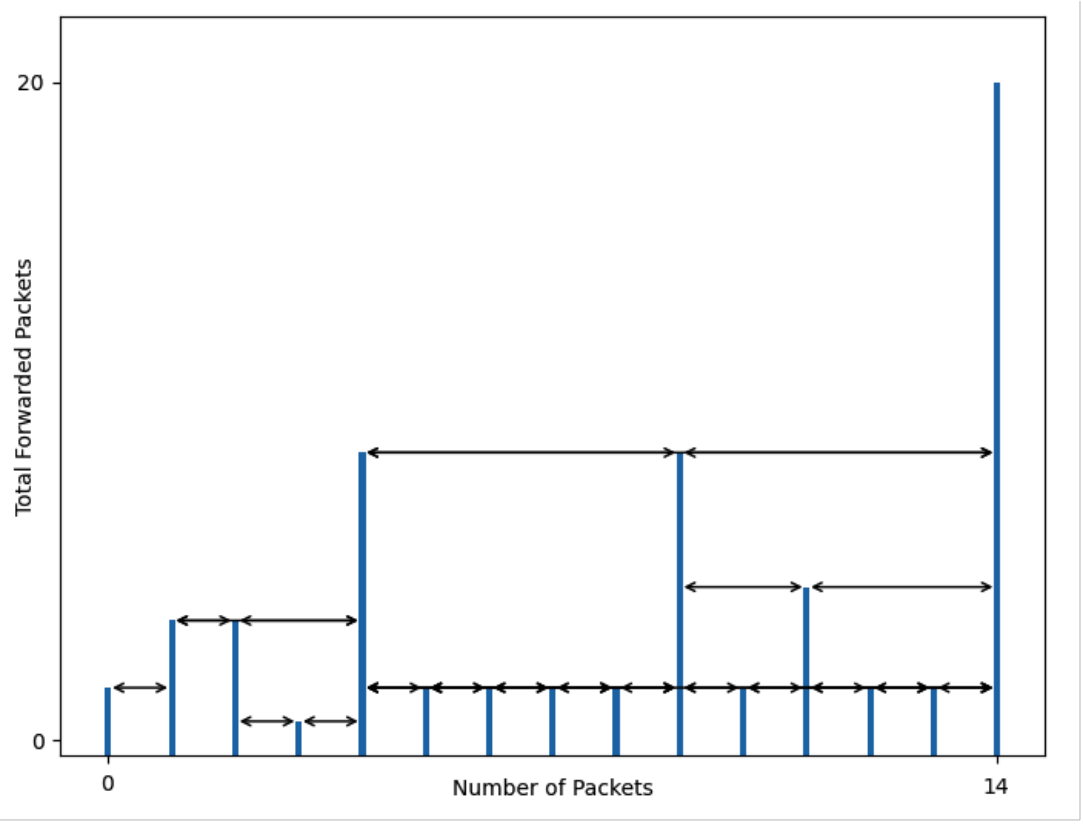

Figure 2.14: Example of a HVG

Chapter 3

Related Work

This related work is divided into two parts: the feature selection stage in ML and DoS attacks and their detection. In the first part, the existing ML types are presented and explained, and works in the area of feature selection are enumerated and detailed. In the second part, different types of DoS and DDoS attack detection strategies are also described.

3.1 ML feature selection

The use of ML to detect and prevent cyberattacks has become increasingly important in recent years, through its application in several studies [18, 66]. ML involves the use of models designed to extract insights from data and make informed decisions and predictions based on the analysis of input data using various algorithms.

One of the most important steps in the machine learning process is data collection. Collecting high-quality data is critical to algorithm performance once it ensures the model has access to a variety of data points that can be used to make accurate predictions. This data can be collected from a variety of sources, such as user behavior logs, network traffic data, and system logs [73].

Once the data has been collected, the next step in the ML process is data processing. This step plays an important role in data cleaning, transformation, and normalization. Data cleaning involves identifying and removing any outliers or anomalies in the data, which can negatively impact model accuracy. Data transformation involves converting

the data into a format that can be easily analyzed by the model. This step also includes normalizing the data to ensure all variables are on the same scale, which helps improve model accuracy [38].

After data processing, the data is divided into training and testing groups. Training data is used to train the model while testing data is used to evaluate the model's performance. This step is critical to ensure that the model accurately predicts cyberattacks and does not just memorize patterns in the training data [13].

Once the testing process is complete, the appropriate algorithm is applied to make predictions and/or decisions based on the data. There are a variety of algorithms that can be used in ML, including decision trees, support vector machines, and neural networks. The choice of algorithm depends on the specific use case and the type of data being analyzed [29]. For instance, decision trees are often used in cases where the data is structured and the result is binary (yes or no). Support vector machines, on the other hand, are often used in cases where data cannot be divided into two distinct groups. Neural networks are often used in cases where data is unstructured, such as image or voice recognition.

In addition to data collection, processing, and training, continuous model evaluation, and improvement are also essential in the ML process. As new cyberattack techniques emerge, the model must be continually updated and refined to remain effective. Overall, the use of ML in detecting and preventing cyberattacks is a rapidly evolving field that requires continuous research and development. As the threat of cyber attacks continues to grow.

In ML, the ability to accurately predict outcomes based on large data sets has become a critical component of many industries. To achieve this, models are developed that can analyze and extract patterns or relationships from data sets. However, the effectiveness of these models depends on their ability to make accurate predictions, and this is where the importance of testing and algorithm selection comes into play.

ML can be categorized into different types of learning: Supervised Learning (SL), Unsupervised Learning, Semi-Supervised Learning, and Reinforcement Learning [30], as shown in Table 3.1. Among these types, SL stands out as the most widely used form of ML [60]. SL consists of learning from labeled data, in other words, the algorithm learns

from a labeled dataset where each input data point is associated with a corresponding output value. Once trained, the algorithm can make predictions or classifications as new input data is input.

Table 3.1: Types of ML

	Supervised	**Unsupervised**	**Semi-Supervised**	**Reinforcement**
Info	Used when the data is labeled and is best for classification and prediction tasks	Used when the data is not labeled. It focuses on looking for patterns in the data itself	A mix of supervised and unsupervised learning. Only part of the data provided has labels or results It is useful when labels are difficult or expensive to obtain	An interactive form of learning that focuses on decision making and strategy determination. For each action performed, the machine receives feedback
Usage examples	Fraud detection	Extraction of association rules	Medical prediction	Complex decision problems

Several algorithms can be used in the training and testing phase of SL, each with its advantages and limitations. For example, the Support Vector Machine classifies the data and can solve classification and regression problems with high accuracy [21]. Linear regression predicts a continuous numerical value. Logistic regression is used in binary classification tasks. Naive Bayes classifies the input data as text [54]. Linear Discriminant Analysis aims to maximize separation between classes and reduce dimensions [26]. Decision Trees classify and generate decision rules [58]. The N-Nearest Neighbor algorithm sorts data when the decision boundary is nonlinear [22]. Perceptron multilayer neural networks recognize patterns when learning complex patterns in speech and image processing

applications [69]. Similarity learning understands the relationships between instances in a dataset [56, 57, 28]. There are multiple ML algorithms [4].

Feature Selection is a step of SL data processing in which the best features of the model are selected to achieve the best performance in the validation set. Different techniques can be used for feature selection, such as grid search, random search, Bayesian optimization, correlation analysis, and recursive feature elimination [27, 35, 1]. Grid search is a simple but computationally intensive method that exhaustively searches all possible combinations of the best features within a predefined range. Random search is a faster and more efficient method that randomly draws the best features from a predefined distribution. Bayesian optimization is a more advanced method that uses probabilistic models to efficiently search for the best features that lead to the best possible model performance [11, 74, 24]. The goal of this step is to identify the ideal set of features that promotes the best model performance in terms of both speed and quality.

In recent years, several works have sought to find the best ways to detect the best features. Man et al. [48] proposed a study on feature selection in ML and the stability of selected features about the intrinsic randomness of feature selection algorithms. When applied to random forests, the authors proposed a stability metric called instability index to compare the stability of three feature selection algorithms (Mean Decrease Accuracy (MDA), Local Interpretable Model-agnostic explanations (LIME), and SHapley Additive exPlanations (SHAP)). MDA is a technique that focuses on evaluating the stability of ML models concerning variation in input data, and LIME is an approach that aims to explain ML models at a local level by focusing on explaining a model's prediction for instance-specific data. Typically, features are selected by averaging many random iterations of a selection algorithm. Although the variability of the chosen features decreases as the number of iterations increases, it does not reach zero, and the features selected by the three algorithms do not necessarily converge to the same set. LIME and SHAP are considered more stable than MDA, and LIME is at least as stable as SHAP for the top-ranked features. However, according to the authors, the selected set of features of the three algorithms significantly improves several out-of-sample predictive metrics, and their predictive performance does not differ significantly. Related to the same topic, a work developed by Sun et al. [79] proposes a new method, which combines the ReliefF

algorithm with neighborhood information measures. First, the method evaluates differences between similar and heterogeneous samples using difference coefficients and mean distance, allowing for more effective sample selection. It then introduces the concept of sample margin and neighborhood for all models under each label, incorporating uncertainty measures based on neighborhood entropy, and to finish, an improved algorithm ML-ReliefF is designed to eliminate unrelated features and a direct algorithm Multi-label feature selection improves classification performance. Still related to ML-ReliefF, a study by Spolaôr et al. [76] presents a new multi-label feature selection algorithm, which extends the single-label feature selection algorithm ReliefF. RF-ML considers interactions between attributes without needing data transformation, and according to the authors, experiments show that RF-ML outperforms methods that transform multi-label data into a single label, highlighting relevant characteristics more frequently.

Existing feature selection techniques, although valuable and play a crucial role in reducing the dimensionality of high-dimensional datasets, are not perfect and face some limitations. Firstly, the choice of which feature selection technique to apply largely depends on the specific nature of the dataset and the problem at hand, which makes it difficult to determine a universally superior approach. Furthermore, these techniques often rely on underlying assumptions, such as independence between resources or linearity of relationships, which may not be valid in all cases. Moreover, feature selection can be computationally intensive, especially when dealing with large datasets or many features. This can make feature selection techniques impractical in some scenarios due to the time and computational resources required. Furthermore, feature selection can lead to the loss of relevant information since discarded features may contain helpful information in combination with other features.

Considering cyber attacks, ML allows detection systems to learn patterns from historical data and identify anomalous activities. Therefore, feature selection is the phase that identifies the most relevant attributes for detecting attacks, which helps to improve the detection model's efficiency and effectiveness, reduce the data's dimensionality, and avoid including irrelevant or redundant features.

VGs can be a good option to mitigate many of these feature selection problems. In addition to being visual and easily readable, they do not require high computational

resources to present reliable results, and there is no loss of information.

3.2 DoS and DDoS attacks detection

Identifying DoS and DDoS attacks is a crucial aspect of defending against cyber threats undermining online services' availability and integrity. As these attacks become more complex and sophisticated, the importance of advanced detection strategies also increases. Therefore, novel and diverse approaches and techniques have been developed to detect DoS and DDoS attacks.

In a study conducted by Carl et al. [14], the authors discuss various techniques for detecting DoS attacks and the challenges associated with identifying network-based flooding attacks. These techniques include activity profiling, change point detection, and wavelet-based signal analysis. Although each method shows promise in limited testing, none entirely solves the problem of detecting flood attacks. The study suggests combining multiple approaches and collaborating with networking experts can yield the best results in successfully identifying DoS flood attacks.

Zhiyuan et al. propose a DoS attack detection system based on Multivariate Correlation Analysis (MCA) to accurately characterize network traffic, identifying geometric correlations between traffic features [80]. DoS attack detection is accomplished through anomaly detection, enabling effective identification of known and unknown attacks while learning only legitimate network traffic patterns. Furthermore, a triangular area-based technique is presented to improve the MCA process. System evaluation is performed using the KDD Cup 99 dataset, demonstrating that it outperforms other approaches in terms of detection accuracy. Briefly, this paper introduced an MCA-based DoS attack detection system enhanced with triangular area technique and anomaly detection. Evaluations show the system achieves high precision when working with non-normalized data, although it has challenges with certain types of attacks. However, statistical normalization solves this problem, resulting in even greater accuracy. In another article published by the same author [81], the research proposes using multivariate correlation analysis to extract second-order statistics from network traffic records, revealing hidden correlative information between resources. This hidden information is leveraged to improve the accuracy of

DoS attack detection significantly. The KDD CUP 99 dataset evaluation demonstrates promising results, with an average detection rate of 99.96% and a false positive rate of 2.08%. This highlights the effectiveness of the proposed approach in protecting the reliability and availability of network services against DoS threats.

Iftikhar et al. present an approach to analyzing DoS attacks using a supervised neural network [2]. The methodology utilizes data from the Kddcup99 dataset and employs a multi-layer perceptron architecture and resilient backpropagation algorithm for training and testing. The system's performance is compared to other neural network approaches, demonstrating higher accuracy in detection rate. The study evaluates the system through two stages of testing, showing high detection rates for various types of DoS attacks, particularly denial-of-service attacks. The research aims to increase the attack detection rate while minimizing false positive and false negative rates.

Zhiyuan et al. propose an approach that treats network traffic records as images and DoS attack detection as a computer vision problem [82]. The system achieves high detection accuracy by utilizing MCA and the Earth Mover's Distance (EMD) metric, even for unknown attacks. Tests on the KDD Cup 99 and ISCX 2012 IDS Evaluation datasets demonstrate impressive detection rates. The authors emphasize the precise network traffic characterization and high accuracy in detecting attacks, outperforming state-of-the-art approaches. The system is also capable of handling high-speed network segments.

Haining et al. introduce a DoS attack detection mechanism called Change Point Monitoring (CPM) [88]. CPM is based on network protocol behavior and uses sequential change point detection, making it robust and applicable to diverse scenarios. It employs the non-parametric cumulative sum (CUSUM) method to ensure traffic and location variations insensitivity. CPM does not require per-flow state information and has low computational overhead. The study evaluates the effectiveness of CPM in detecting SYN flood attacks, demonstrating high accuracy and short detection time. CPM is particularly useful in scenarios with multihomed networks.

Bawany et al. explore Software-defined networking (SDN)-based solutions to address the challenges of DDoS attacks [8]. They classify and analyze existing solutions, highlighting their limitations in application-specific detection. Based on this analysis, the authors propose "ProDefense", a framework incorporating application-specific traffic threshold

criteria, enabling personalized DDoS attack detection. ProDefense adopts a distributed controller platform to improve scalability and resilience. The paper emphasizes the need for customizable detection and a distributed approach to mitigating DDoS attacks, as existing solutions are not universally effective.

Mittal et al. review the current literature on detecting DDoS attacks using Deep Learning (DL) approaches [55]. They categorize DL-based DDoS attack detection approaches into five categories and analyze their prevalence in the literature. The study acknowledges the high performance observed in many studies but highlights the challenges in direct comparison between approaches due to variability in datasets, experimental settings, and performance metrics. The lack of evaluations in production and real-time environments is also a limitation. The authors stress the need for standardization and more comprehensive assessments to ensure the security of online systems.

Keunsoo et al. propose a method for proactive detection of DDoS attacks using cluster analysis [43]. The technique analyzes the phases of a DDoS attack, selects variables based on these phases' characteristics, and applies cluster analysis to detect the attack early. Evaluation results show that the method effectively partitions traffic into groups corresponding to different stages of the DDoS attack, enabling early detection. The authors highlight the selection of parameters sensitive to the attack's phases and the application of cluster analysis as crucial factors in detecting precursors of DDoS attacks. The method is considered easy to implement and can aid in developing defense mechanisms against these attacks.

Feinstein et al. present a proposal that utilizes entropy calculation and frequency-ordered distributions of selected packet attributes to identify anomalies in DDoS attack characteristics [25]. Evaluation results using real-time traffic data demonstrate the effectiveness of these methods against current attacks while indicating room for improvement in detecting stealthier attacks. The developed detector prototype can determine whether a network is under attack and implement precise filtering rules, reducing the impact of the attack almost instantly. The system is adaptable to different network environments with minimal manual tuning, making it a practical solution against existing DDoS tools. However, further techniques are explored to enhance defense against future, more sophisticated attacks.

Yang et al. propose an AutoEncoder-based detection framework that builds the detection model using only normal traffic and can automatically update itself over time [95]. Experiment results on diverse datasets demonstrate that the proposed framework achieves better detection rates and can detect new and unknown DDoS attacks that challenge supervised machine learning algorithms. The authors acknowledge the promise of the framework but also highlight concerns that need further investigation, such as continuous updates to make the model more robust, resistance to poisoning attacks, and deployment in DDoS threat signaling environments with multiple intelligent agents.

Although promising, the DoS and DDoS attack detection techniques listed in section 3.2 are not entirely reliable due to several challenges intrinsic to this field of cybersecurity. First, the constant evolution of tactics attackers use makes it difficult for traditional detection techniques to keep up. Attackers are constantly looking for new ways to hide their activities and create more sophisticated DDoS attacks, which can result in inadequate detection techniques to identify new attack methods. Additionally, signature-based detection techniques, which look for specific traffic behaviors associated with known attacks, can generate false positives when legitimate traffic resembles an attack, generating false alerts and making it more difficult to distinguish true threats. Another problem is the ability of attackers to carry out stealth attacks that resemble normal traffic, making them almost indistinguishable from traditional detection techniques. This means these approaches may fail to detect more subtle attacks that seek to avoid detection. Additionally, traditional detection techniques often rely on historical data and behavior models that may not apply to emerging and unknown threats. Anomaly-based detection, for example, may not be effective in detecting DDoS attacks that have not yet been observed. In summary, although DoS and DDoS attack detection techniques have advanced over the years, they could be more foolproof. The constant evolution of cyber threats, the ability of attackers to evade detection, and the limitations of traditional approaches highlight the continued need for research and development of more robust and adaptable methods to protect networks against these threats. In this way, the use of VGs meets a current requirement, as through them, it is possible to detect a denial of service attack through the differences and analysis of patterns generated by benign and malignant traffic.

In summary, VGs can help select features in a scenario where ML is used, but they

can also completely replace the use of ML through their data representation capabilities.

Chapter 4

Using VGs for ML Features Selection

This chapter describes how VGs can be used to select the best features to use in ML, for instance, to detect DDoS attacks. Different tools that perform this task are compared, a use case is depicted, and the results obtained by using VGs are presented.

4.1 Context

Feature selection plays a fundamental role in the data preparation phase of SL. The goal is to select the best features for more reliable and optimized results. The current work explores VGs to identify the most relevant features to enable monitored ML based IDS to detect DoS/DDoS attacks. Figure 4.1 presents the steps used in ML [23, 40], detailing the phase in which the VGs are applied.

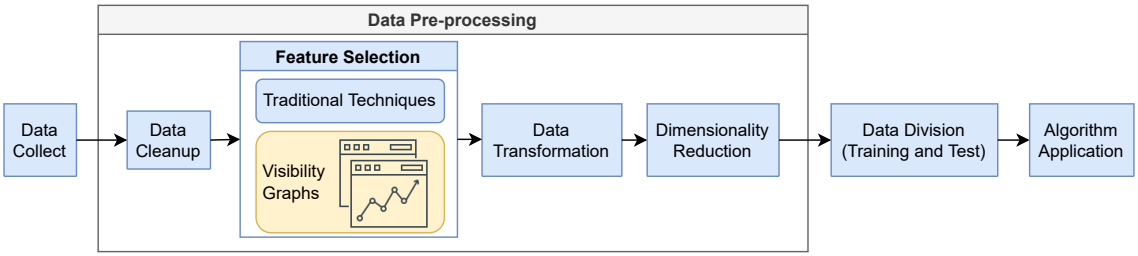

Figure 4.1: ML stages using VGs in Feature Selection.

As explained in Section 3.1, the feature selection can be performed using different

techniques. Table 4.1 contains information on the traditional techniques as its comparison to the VGs presented in Chapter 3, showing their potential and usefulness for feature selection in Supervised ML. For instance, the Grid Search technique is the most straightforward, the Random Search technique is an iterative but uninformed search, and the Bayesian Optimization uses probabilistic feature selection. At the same time, VGs is a non-iterative and uninformed search, fast, visual, and with low computing requirements.

Table 4.1: Comparison between supervised ML feature selection techniques, including VG proposal

Techniques	**Properties**
Grid Search	The most simple and straightforward Non-iterative and uninformed search method Computationally demanding Exhaustive exploration All possible feature combinations
Random Search	Iterative but uninformed search method Random sampling Reduced computational costs Good feature discovery probability
Bayesian Optimization	Iterative and informed search method Probabilistic feature selection Based on utility Maximises efficiency
Visibility Graphs	Non-iterative and uninformed search Fast, visual and with low computing requirements No probability work required Alternative to traditional methods

4.2 Use Case

A study conducted by Sarcevic et al. in 2022 [70] looks at using ML resources in combination with explainable Artificial Intelligence (AI) to classify network traffic. Explainable

Artificial Intelligence (XAI), also known as Interpretable AI, uses white-box ML where the reasoning process can be described and understood by humans. This is in contrast to black-box algorithms, where the decision-making process is less transparent.

In this specific study, the authors used the approach called SHAP to calculate the individual contributions of each resource used in an "if-then" decision tree. This technique, based on game theory, is intended to provide a deeper understanding of how each feature affects the overall performance of the ML model. More specifically, the method SHAP evaluates the importance of each feature and indicates the extent to which it affects the overall predictions of the model [49].

The research conducted by Sarcevic et al. aims to identify ML features that are effective in detecting DoS/DDoS attacks. This investigation is supported by Table 4.2, where the most relevant features in the detection of DoS/DDoS traffic are presented, enumerated by the research by Sarcevic et al.. The Table 4.2 presents four features, "Init Win bytes forward" represents the initial size of the amount of data transmitted, "Flow IAT Mean" measures the average interval between successive chunks of data in a flow continuous on the network, the "Destination Port" is the destination port number of the requests and the "Min Seg Size Forward" determines the minimum size of chunks of data divided for transmission on the network. With this in mind, one of the contributions of this thesis is to validate the effectiveness of the ML resources identified by Sarcevic et al. using an alternative and easily implementable approach called VGs.

Table 4.2: The most effective ML features for identifying DoS and DDoS attacks, according to [70].

No.	**Feature**	**Description**
1	Init Win bytes forward	The initial opening size controls the quantity of data transmitted on the network.
2	Flow IAT Mean	The interval between successive pieces of data in a continuous stream on the network.
3	Destination Port	A number indicating which service should receive the data upon arrival at its destination.

4	Min Seg Size Forward	The smallest size of data chunks that are divided for transmission over the network.

To illustrate the practical application, this research employs the DDoS application layer dataset accessible in Kaggle [91]. This dataset comprises network logs that describe both benign and malignant traffic. It has 77 resource columns and 1 target column, formatted in *csv*. Details of the dataset columns can be found in Appendix A. Each entry corresponds to a network packet, accompanied by an associated label. This label distinguishes the type of traffic into three categories:

- **Slow loris DDoS:** can be described as the IP network traffic that is present in an attack, where multiple TCP connections are opened towards a target and kept open for an indefinite period. These open connections are used to repeatedly make HTTP requests, resulting in a DDoS attack [72].
- **Hulk DDoS:** It is typically an attack that spoofs the UserAgent field and sends a large volume of HTTP GET requests, which are agnostic to the target and therefore help to make the attack go unnoticed [7].
- **Benign Traffic:** Refers to IP network traffic that does not contain any DoS or DDoS attacks.

The analysis carried out with the proposed dataset focuses on how VGs can help in selecting the best input features for ML algorithms to detect DDoS attacks.

The Table 4.3 presents four of the best features to detect DDoS attacks. This is based on the study by Sarcevic et al. [70] through the description of the usefulness of the selected features, as well as, relevant information that discriminates each of the features for a better understanding of the results.

Table 4.3: Features analyzed by the VGs.

No.	**Feature**	**Info**
1	Init Win bytes forward	High values (and many edges) of this feature and destination port equal to 80 indicate benign traffic, however low values (close to 0 and no edges) indicate DDoS attacks.
2	Flow IAT Mean	High values of this feature indicate DDoS attacks, lower values indicate benign traffic.
3	Destination Port	Main feature for classifying benign traffic and DDoS traffic. The port value is constantly changing in benign traffic (and there are many edges) while traffic to only one port, typically port 80 (and only a few edges) indicates DDoS attacks.
4	Min Seg Size Forward	Changes in values (many edges) indicate a high probability of benign traffic, while DDoS attack traffic is not constant and does not present change in its values (no edges).

The graphs presented in Figures 4.2, 4.3, 4.4 and 4.5 illustrate the VGs corresponding to the features highlighted in Table 4.3. The graphs on the left side of the images correspond to the graphical representation of benign network traffic, while those on the right side depict the graphical representations of network traffic associated with DDoS attacks.

The comparison of 500 network packets originating from the *Init Win bytes forward* feature is represented in Figure 4.2. This feature refers to the number of bytes in the initial send time window. The left side of the graph relates to benign network traffic, while the right side refers to traffic resulting from a DDoS attack. The graphs show that the maximum value of this feature for benign traffic is 65,535, while for malicious traffic, it is zero. Furthermore, the number of edges in the benign traffic graph is much larger than in the attack traffic graph, demonstrating the usefulness of this feature in distinguishing between the two types of traffic.

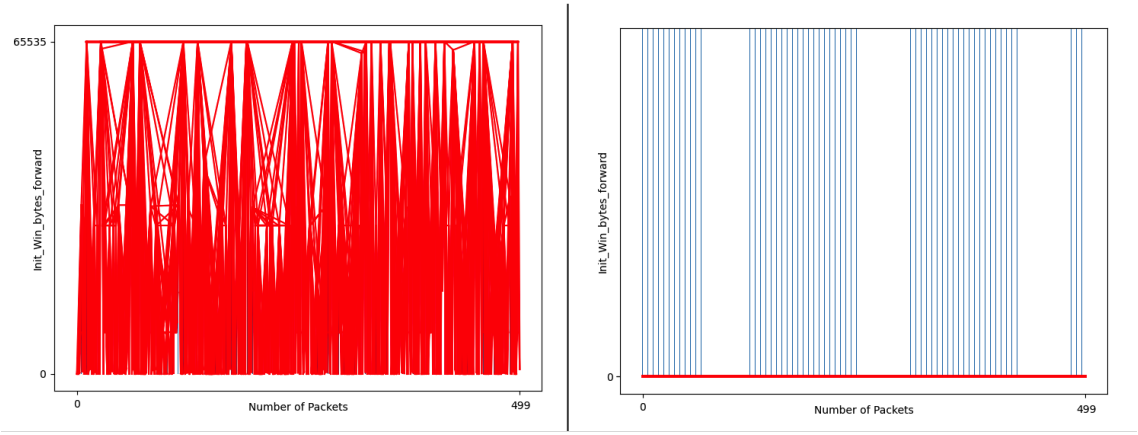

Figure 4.2: VGs Comparison Results - *Init Win Bytes Forward* feature.

The comparison of 500 network packets originating from the *Flow IAT Mean* feature is represented in Figure 4.3. Displays the average interarrival time of network flows in both directions. The left side of the graph refers to benign network traffic, while the right side refers to traffic resulting from a DDoS attack. The maximum value of this feature is considerably higher in the case of a DDoS attack (120,000,000) than in the case of benign traffic (64,067,751). Although the differentiation in this case is quantitative and not visual, it still allows differentiation between the two types of traffic.

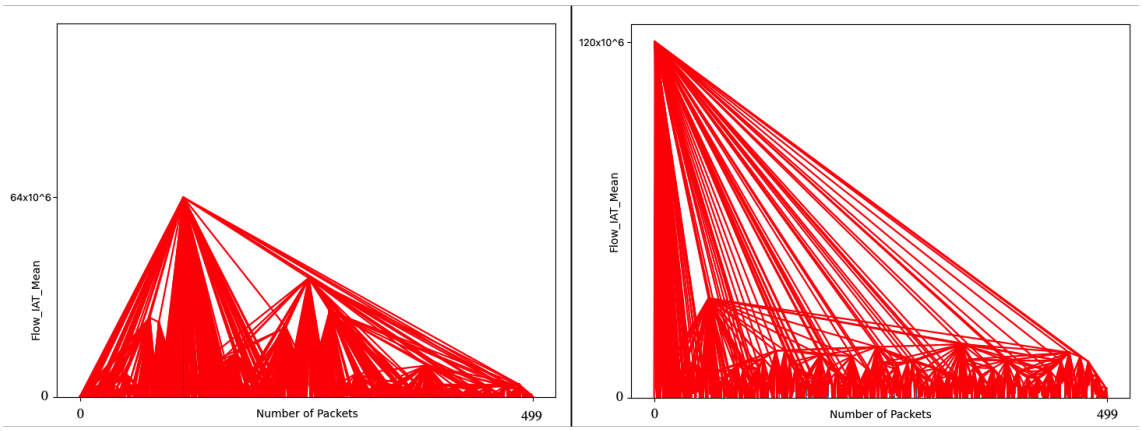

Figure 4.3: VGs Comparison Results - *Flow IAT Mean* feature.

The comparison of 500 network packets originating from the *Destination Port* feature is represented in Figure 4.4. The left side of the graph refers to benign network traffic, while the right side refers to traffic resulting from a DDoS attack. The graph remains

unchanged during a DDoS attack, taking into account the traditional HTTP port, port 80, while port numbers change frequently in benign traffic. Furthermore, the number of edges in the case of benign traffic is consequently more significant than in the case of DDoS attack traffic. This result confirms this functionality's usefulness in distinguishing between the two types of traffic: benign and DDoS.

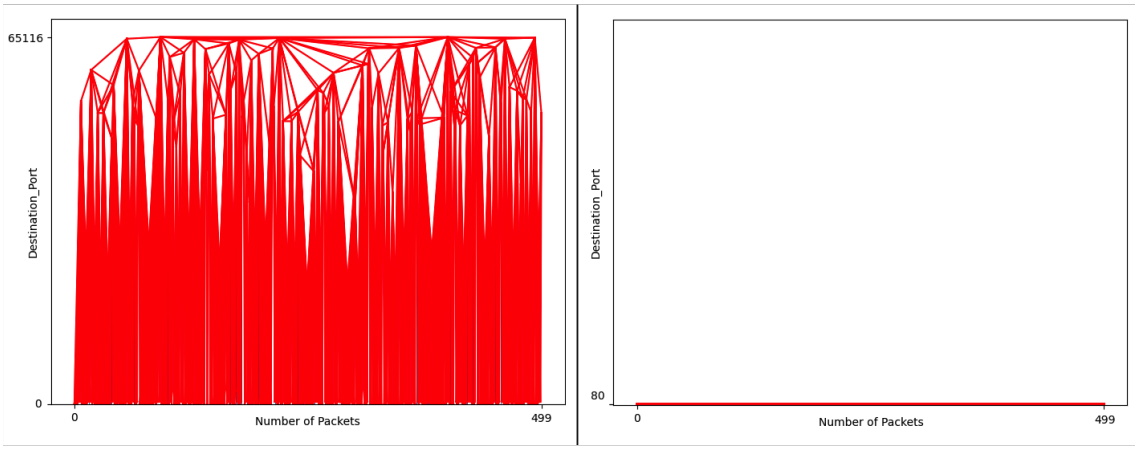

Figure 4.4: VGs Comparison Results - *Destination port* feature.

The comparison of 500 network packets originating from the *Min Seg Size Forward* feature is represented in Figure 4.5. Indicates the minimum IP network packet segment size. The left side of the graph refers to benign network traffic, while the right side refers to traffic resulting from a DDoS attack. Although this feature is good for distinguishing between DoS and DDoS attacks, according to Sarcevic et al. [70], this use case is also helpful for discriminating between DDoS traffic and benign traffic. The minimum and maximum values differ in both types of traffic, but the number of edges is more significant in benign traffic than in the DDoS attack. Therefore, this result suggests that this feature can also differentiate the two types of traffic.

Visibility patterns generated by benign network traffic and DDoS were analyzed in this study. The findings, as summarized in Figures 4.2, 4.3, 4.4, 4.5 and Table 4.3, demonstrate the effectiveness of using VGs to identify relevant features for supervised ML models. The study hypothesis is confirmed by the results, which show that VGs can differentiate between benign network traffic and DDoS attacks. Therefore, it is possible to say that using VGs can obtain the same results as traditional feature selection techniques.

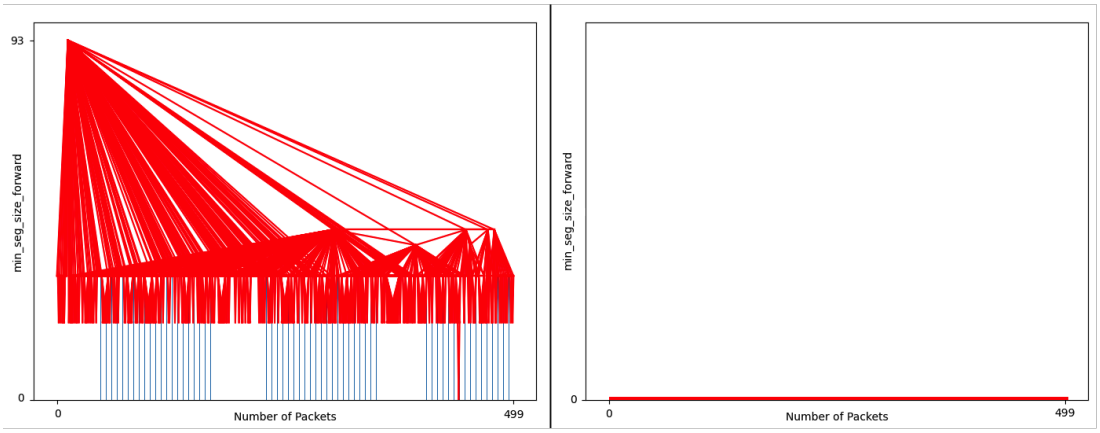

Figure 4.5: VGs Comparison Results - *Min Seg Size Forward* feature.

Within the scope of the study carried out, it was found that the process of creating a VG for a given set of data depended on the number of packages analyzed. This was observed through the analysis of several datasets and it was found that there is a linear relationship between the number of packets and the computation time required to generate the corresponding VG. This relationship is illustrated in Figure 4.6, which shows direct proportionality between dataset size and computational requirements.

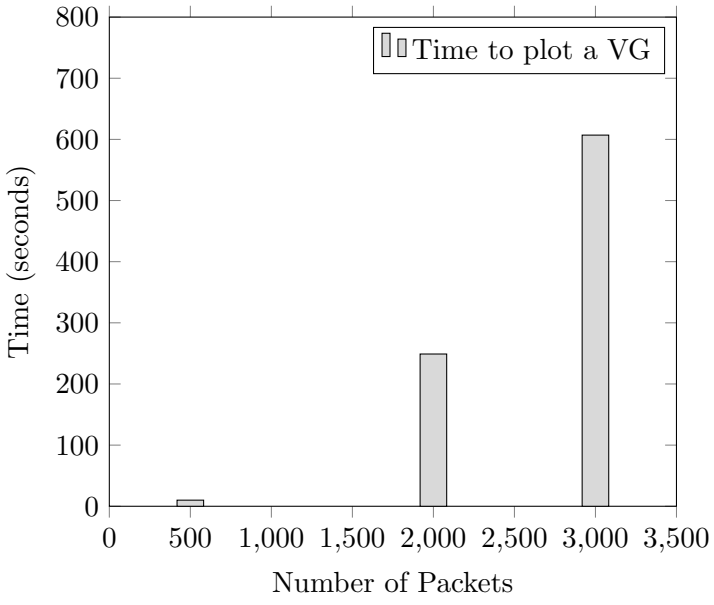

Figure 4.6: Time spent plotting a VG

4.3 Implementation

To achieve the expected results in the use case, a pipeline was created using the Python programming language (see Figure 4.7)[1].

1. The first step of the process, presented in Listing 4.1, loads the dataset that consists of network events. These records contain information from both benign and malicious nodes based on HTTP communication. In this case, the dataset, once loaded, is grouped by the value of the "Label" column, which identifies whether it is a benign or malicious network packet. After that, it creates a CSV file for each type of traffic.

```
# Read the CSV file
    df = pd.read_csv('logs/test_mosaic.csv')

# Group the rows by the value of the 'Label' column
    grouped = df.groupby('Label')

# Save each type to a separate CSV file
    for name, group in grouped:
        group.to_csv(f'{name}.csv', index=False)
```

Listing 4.1: First Step - ML Features Selection using VGs

2. Next, as presented in Listing 4.2, the data goes through a filtering phase in which the relevant feature is selected. This feature forms the basis for the creation of a VGs. Before moving on, the data is cleaned and optimized. If the selected feature consists of categorical values, it is automatically converted to numeric values so that it can be processed efficiently. This conversion is performed using the LabelEncoder function from the Python library sklearn. preprocessing. It involves identifying the columns with categorical or object data types and then traversing through those columns with the fit_transform() function. This preprocessing of the data is essential before the concept of VGs can be applied. To this end, a customized Python script is used to transform these values into important data points for creating the VGs.

[1] Appendix B contains the code for this implementation.

```
# Upload the CSV
    df = pd.read_csv("BENIGN.csv")

# Transform categorical values to numeric
    def Encoder(df):
      columnsToEncode = list(df.select_dtypes(include=
        ['category','object']))
      le = LabelEncoder()
      for feature in columnsToEncode:
          try:
              df[feature] = le.fit_transform(df[feature])
          except:
              print('Error encoding '+feature)
      return df
    df = Encoder(df)
```

Listing 4.2: Second Step - ML Features Selection using VGs

3. Once this step is completed, the values associated with the selected feature are stored in a preparatory data list, as presented in Listing 4.3.

```
# This script points to the column where the VG algorithm will be
    applied and turns it into a list
    ts = df["Total_Fwd_Packets"].tolist()
```

Listing 4.3: Third Step - ML Features Selection using VGs

4. In the following phase, VGs begin to be created. In this step, as presented in Listing 4.4, ordered pairs of values (x, y) are created that represent values visible to both sides. The primary concern here is to identify the edges that are present in the VGs.

```
# Necessary libraries are imported
    from matplotlib import pyplot as plt
    import numpy as np
    from visibility_graph import visibility_graph

# Creating ordered pairs of values (x, y)
    result = visibility_graph(ts)
```

Listing 4.4: Fourth Step - ML Features Selection using VGs

5. Finally, based on the ordered value pairs (x, y) defined in the previous step, the points and the corresponding rays are drawn in the graph, leading to the formation of the VGs, as presented in Listing 4.5.

```
fig = plt.figure()
N = 15
ax = fig.add_axes([0,0,1,1])
ind = np.arange(N)

ax.bar(ind,ts, width=0.1)
ax.set_xticklabels([])
ax.set_yticklabels([])
ax.set_xticks([])
ax.set_yticks([])
ax.set_xlabel('Number of Packets')
ax.set_ylabel('Total Forwarded Packets')
array_length = len(ind)
last_element = ind[array_length - 1]
edgelisteth = list(result.edges)
plt.xticks(list(ax.get_xticks()) + [min(ind), max(ind)])
plt.yticks(list(ax.get_yticks()) + [min(ts), max(ts)])

for i,j in edgelisteth:
 if ts[i]<ts[j]:
   plt.annotate(xy=(j,ts[(i)]), xytext=(i,ts[(i)]),
   arrowprops=dict(arrowstyle='<->'), text='')
 else:
   plt.annotate(xy=(j,ts[(j)]), xytext=(i,ts[(j)]),
   arrowprops=dict(arrowstyle='<->'), text='')

plt.tight_layout()
plt.show()
```

Listing 4.5: Fifth Step - ML Features Selection using VGs

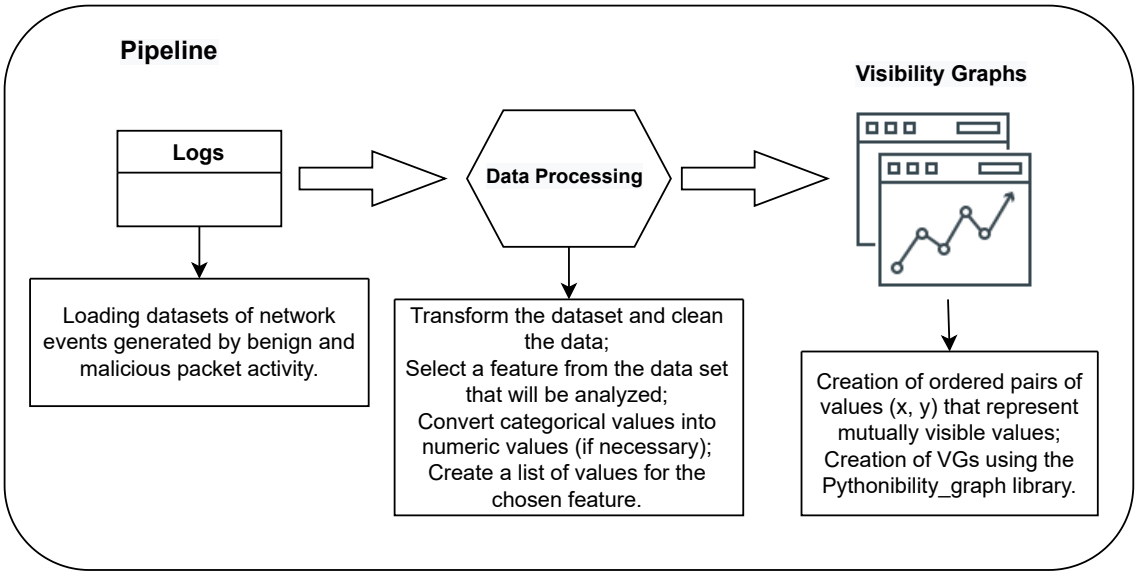

Figure 4.7: Visual representation of the steps followed in the implementation methodology

4.4 Summary

Taking into account the results obtained throughout this use case, it is possible to say that through the analysis of Figures 4.2, 4.3, 4.4, 4.5 and evaluating its values, there is potential in using VGs to identify relevant features for supervised ML models. The study proposal is confirmed by the results showing that VGs can differentiate benign network traffic and DDoS attacks. In this way, it is possible to say that the use of VGs can obtain the same results as traditional feature selection techniques and thus be a reliable alternative to be used and studied.

Chapter 5

Using VGs to detect DDoS attacks

This chapter describes how VGs and HVGs can be useful for DDoS attack detection. Some of the most commonly used tools for this purpose are discussed and tests with VGs are performed. The obtained results are also presented and discussed.

5.1 Context

Every day, online platforms and services fall victim to cyberattacks. These malicious activities often exploit various vulnerabilities with techniques aimed at compromising endpoints [71]. This chapter explores the role of VGs and HVGs as an innovative approach to detecting DDoS attacks, their utility compared to traditional tools (Table 5.1), and how they fit into the cybersecurity landscape. VGs and HVGs provide a unique way to visualize network behavior and enable real-time detection of anomalies based on the patterns and values displayed. They can capture complex relationships between network nodes and uncover patterns indicative of suspicious activity, and they can detect changes in traffic on a global scale, making them an effective means of detecting attacks that attempt to masquerade as normal traffic.

The specific patterns that DoS and DDoS attacks create in network logs are a potential source of information for detecting them [86]. Logs contain temporal information about network traffic, i.e., recorded events and actions that, when analyzed, trigger notifications or even alerts that require the intervention of incident response teams. With the increase in DDoS attacks, multiple detection and notification tools that understand this type of

malicious activity have appeared in the literature.

One of the most common types is the use of an IDS, which monitors network traffic for suspicious activity or malicious patterns. While this approach can be effective, its accuracy can vary depending on the configuration and quality of rules defined by users. Additionally, detection time can be slow, especially in the case of complex attacks. Furthermore, implementing traditional IDS can be complex and resource-intensive [36].

Another approach to intrusion detection is Security Information and Event Management (SIEM) systems. These systems aggregate and analyze security information from multiple sources to identify potential threats. They can correlate events effectively, but their accuracy can depend on the configuration and quality of the input data. SIEM systems are often resource-intensive and complex to implement.

Approaches based on AI and ML have gained popularity in recent years due to their high learning and adaptation capabilities. These systems can learn from past data and adapt to new threats, making them highly effective in detecting and preventing cyberattacks. However, its accuracy may depend on the quality of the training data and the training time may be long. Furthermore, AI and ML-based systems require significant resources and may require the creation of complex models.

A relatively new approach to intrusion detection is Automatic Machine Learning and Anomaly Detection (ADELE). This data-driven method is used to detect and predict anomalies and can continuously learn. It has high accuracy but requires a large amount of data for training. ADELE requires moderate resources and specialized implementation.

MapReduce, a programming model intended to provide a scalable approach for big data processing and pattern recognition, can be particularly useful for analyzing large data sets and identifying patterns that may be indicative of malicious activity. However, it is resource-intensive [92] and requires experience in distributed programming.

In conclusion, there are several types of intrusion detection and warning systems available, each with its own strengths and weaknesses. However, it is clear that effective detection and prevention against cyberattacks is crucial in today's threat landscape and these types of tools are not always effective enough to stop DDoS attacks. To resolve this issue, several studies have been carried out with the aim of early detection of DoS and DDoS attacks using different techniques. A study by Hajtmanekj et al. proposes the

use of Hurst and autoregression coefficients and the coefficient of variation to detect DoS attacks in their initial stages [33]. They suggest monitoring ports continuously to identify repeated requests from an IP/MAC address that exceeds a certain threshold.

Another study by Saritakumar et al. analyzes the use of a congestion control-based algorithm with a rate-limited queuing mechanism and an entropy-based algorithm with adaptive threshold estimation [59]. This involves low-level attack detection using entropy calculation and adaptive threshold estimation.

In addition to these techniques, other studies explore alternative algorithms as a means of early detection of DoS attacks. For example, an attack packet detection algorithm can be used to detect this type of attack, minimizing processing overhead and increasing security in Vehicular Ad Hoc Networks [68]. Detection methods that use long-term entropy have fewer fluctuations, which makes it easier to detect abnormal events using a threshold. However, these methods cannot detect the attack in its early stages and cannot track short-term attacks [62].

Developing effective tools and techniques to combat DDoS attacks is an ongoing process that requires ongoing research and development.

Table 5.1: Comparison between VGs and Traditional DoS/DDoS Attack Detection Tools

Characteristics	**VGs**	**Traditional Tools**
Graph-Based View	VGs use a graphical approach to model network behavior and provide a unique view of traffic relationships.	Traditional tools often rely on event and alert logs.
Traffic Relationship Modeling	VGs comprehensively capture the dynamics of traffic relationships and enable anomaly detection.	Conventional tools can reach their limits when modeling detailed traffic relationships.

Robustness against Attack Obfuscation	VGs are resistant to covert attack tactics due to their analysis of patterns in complex networks.	The effectiveness of traditional tools can be compromised by evasion tactics.
Integration with SIEM Systems	VGs can be integrated with SIEM systems to improve security data analysis.	SIEM integration can be an additional feature in some traditional tools.
Learnability (AI/ML)	VGs have a limited learning capability because they focus on modeling networks.	Some traditional tools include AI/ML to improve detection.
Scalability (MapReduce)	The scalability of VGs is limited compared to MapReduce approaches for processing large amounts of data.	MapReduce-based approaches are highly scalable.

5.2 Use Case

To detect DDoS attacks using VGs/HVGs, research was first conducted to find a dataset. The dataset used [67] is part of a site called "The Open University" that provides datasets that can be used for research. The dataset chosen [67] features a unique file containing network events generated by benign and malicious nodes, with a focus on HTTP DDoS attacks. The file is in PCAP format, a file format used by applications to monitor network traffic. This dataset is composed of the following features:

- **No.** - This is the number that identifies each of the lines corresponding to each of the captured network events.
- **Time** - This is the date and time when a particular event was captured.
- **Source** - This is the origin of the event.
- **Destination** - This is the location to which the event was sent.

- **Protocol** - A set of standards that allow each endpoint to connect to another over the network.
- **Info** - Information about the sending of packets and/or the type of orders.

To assess this use case, the two features that contain the most information in this dataset were selected: Protocol and Length. The dataset was split into two parts, with the first set containing benign traffic logs and the second set containing benign traffic logs and traffic with DDoS attacks. For each of the datasets and each of the selected features, two different durations of the network logs were considered, 0.45 and 1 second, respectively. The Table 5.2 contains detailed information about the specific tests performed on the *pcap* file using the *Length* feature: benign or DDoS logs, dataset size, type of graph constructed, period time of the analyzed logs, the time needed to build the graph and reference to the corresponding figure. The Table 5.3 contains detailed information about the specific tests performed on the *pcap* file using the *Protocol* feature. The reasoning behind setting a time interval for analyzing the traffic is related to the large amount of data in the records.

Table 5.2: Detailed information about VGs tests - *Length* Feature.

Data set (logs)	**Data set size (packets)**	**Graph type**	**Time span (sec)**	**Computing time**	**Figure**
Benign	60	VG	0.45	6 sec	5.1
Benign+DDoS	1000	VG	0.45	3 min	5.1
Benign	130	VG	1	19 sec	5.2
Benign+DDoS	2000	VG	1	5 min	5.2

Table 5.3: Detailed information about VGs tests - *Protocol* Feature.

Data set (logs)	**Data set size (packets)**	**Graph type**	**Time span (sec)**	**Computing time**	**Figure**
Benign	60	VG	0.45	5 sec	5.3
Benign+DDoS	1000	VG	0.45	3 min	5.3
Benign	130	VG	1	16 sec	5.4
Benign+DDoS	2000	VG	1	6 min	5.4

Figures 5.1 to 5.4 show the overall visual result of applying the data processing pipeline described in Section 5.3 using the data summarised in Tables 5.2 and 5.3. In Figures 5.1 and 5.2, the graphs on the left represent the dataset with benign logs. It is worth noting that the *Length* feature varies between 60 and 218. The graphs on the right side display the dataset with all logs including DDoS. The *Length* feature varies in a larger range: between 54 and 1514. The range of values in the *Length* feature is considerably larger.

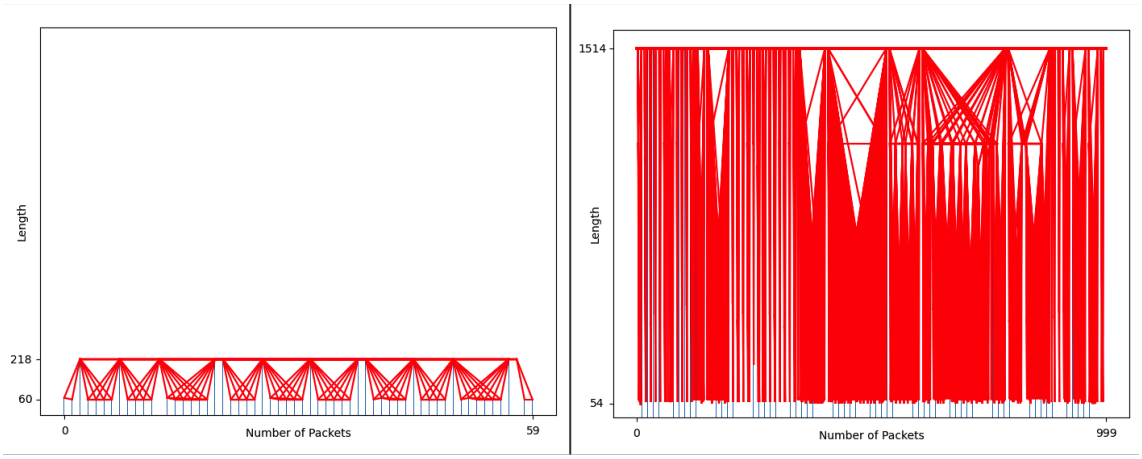

Figure 5.1: VG Results - Comparing Feature *Length* (Benign - 60 Packets VS DDoS - 1000 Packets) - Time frame duration of 0.45 seconds

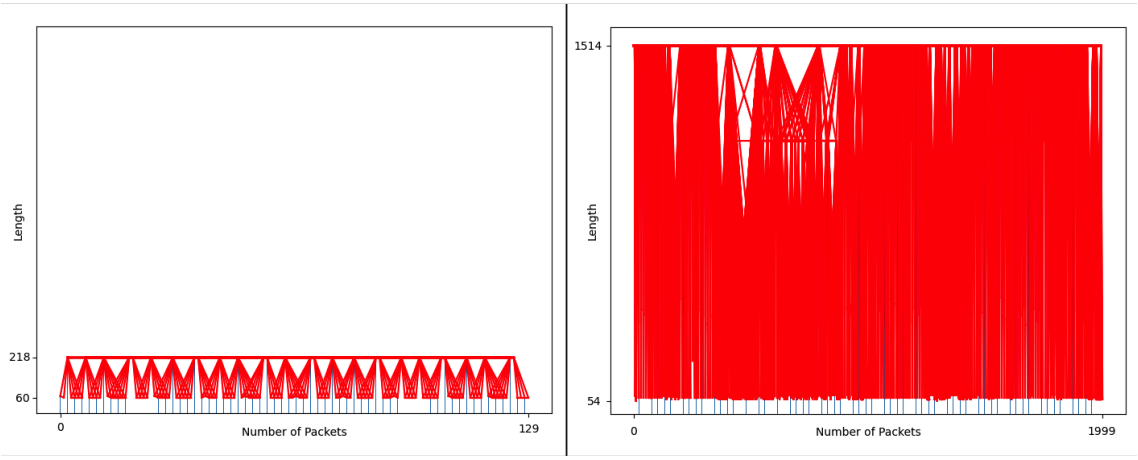

Figure 5.2: VG Results - Comparing Feature *Length* (Benign - 130 Packets VS DDoS - 2000 Packets) - Time frame duration of 1 second

The analysis of Figures 5.3 and 5.4, based on the *Protocol* feature, reveals fewer differences between benign and DoS traffic. The value of the *protocol* feature varies between 0

and 1, with the value 0 corresponding to the HTTP protocol and the value 1 corresponding to the TCP protocol.

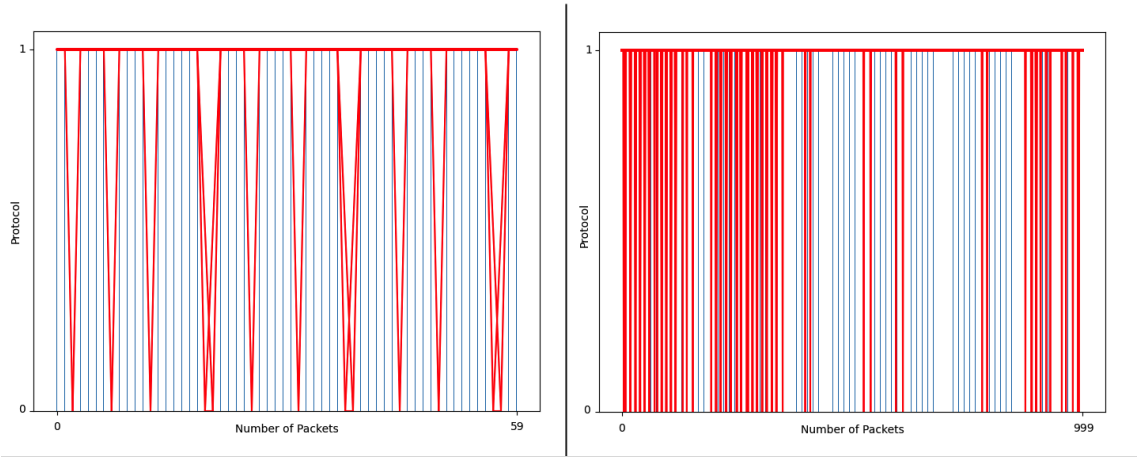

Figure 5.3: VG Results - Comparing Feature *Protocol* (Benign - 60 Packets VS DDoS - 1000 Packets) - Time frame duration of 0.45 seconds

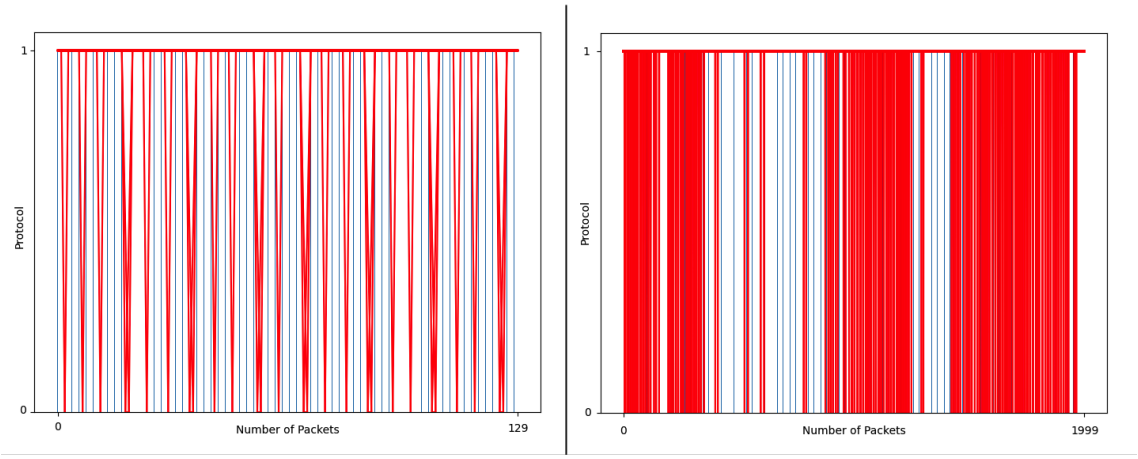

Figure 5.4: VG Results - Comparing Feature *Protocol* (Benign - 130 Packets VS DoS - 2000 Packets) - Time frame duration of 1 second

Considering all results presented there is a higher frequency of sent packets in the graphs that represent DDoS attack logs plus a greater oscillation in the exchange of protocols. Considering the visual results presented, plotting VG based on short periods of network logs seems to be a useful method to identify DDoS attacks, as DDoS traffic has a different pattern compared to benign traffic, both in terms of the range of values on the y-axis and the frequency of visibility rays. The feature *Length* seems to be more suitable for identifying these attacks.

Now, using HVGs, for each of the datasets and each of the selected features, two different durations of the network logs were considered, 0.45 and 1 second, respectively. The Table 5.4 contains detailed information about the specific tests performed on the *pcap* file using the *Length* feature: benign or DDoS logs, dataset size, type of graph constructed, period time of the analyzed logs, the time needed to build the graph and reference to the corresponding figure. The Table 5.5 contains detailed information about the specific tests performed on the *pcap* file using the *Protocol* feature.

Table 5.4: Detailed information about HVGs tests - *Length* Feature

Data set (logs)	**Data set size (packets)**	**Graph type**	**Time span (sec)**	**Computing time**	**Figure**
Benign	60	HVG	0.45	13 sec	5.5
Benign+DDoS	1000	HVG	0.45	4.5 min	5.5
Benign	130	HVG	1	33 sec	5.6
Benign+DDoS	2000	HVG	1	9 min	5.6

Table 5.5: Detailed information about HVGs tests - *Protocol* Feature

Data set (logs)	**Data set size (packets)**	**Graph type**	**Time span (sec)**	**Computing time**	**Figure**
Benign	60	HVG	0.45	12 sec	5.7
Benign+DDoS	1000	HVG	0.45	6 min	5.7
Benign	130	HVG	1	25 sec	5.8
Benign+DDoS	2000	HVG	1	11 min	5.8

In Figures 5.5 and 5.6, the graphs on the left represent the dataset with benign logs. It is worth noting that the *Length* feature varies between 60 and 218. The graphs on the right side display the dataset with all logs including DDoS. The *Length* feature varies in a larger range: between 54 and 1514. The range of values in the *Length* feature is considerably larger.

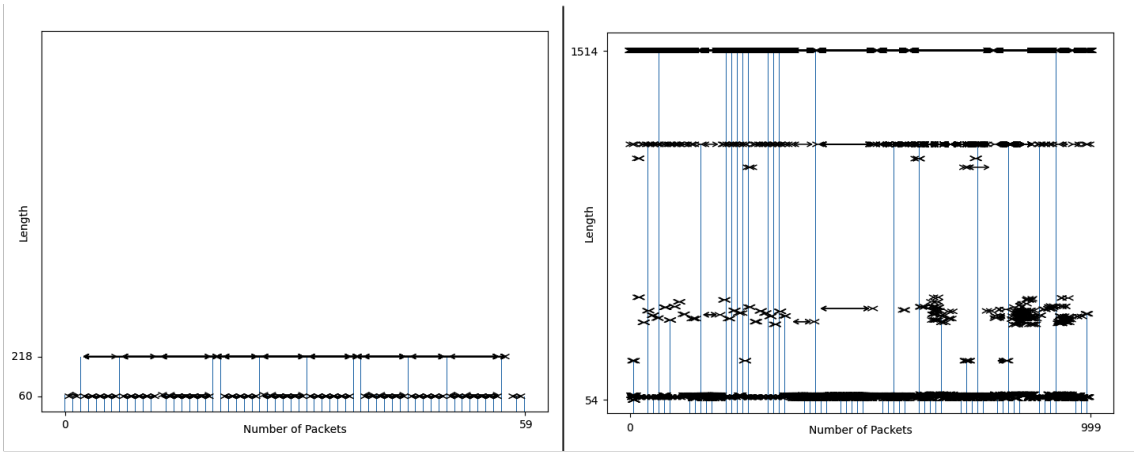

Figure 5.5: HVGs Results - Comparing *Length* Feature (Benign - 60 Packets VS DDoS - 1000 Packets) - Time frame duration of 0.45 seconds

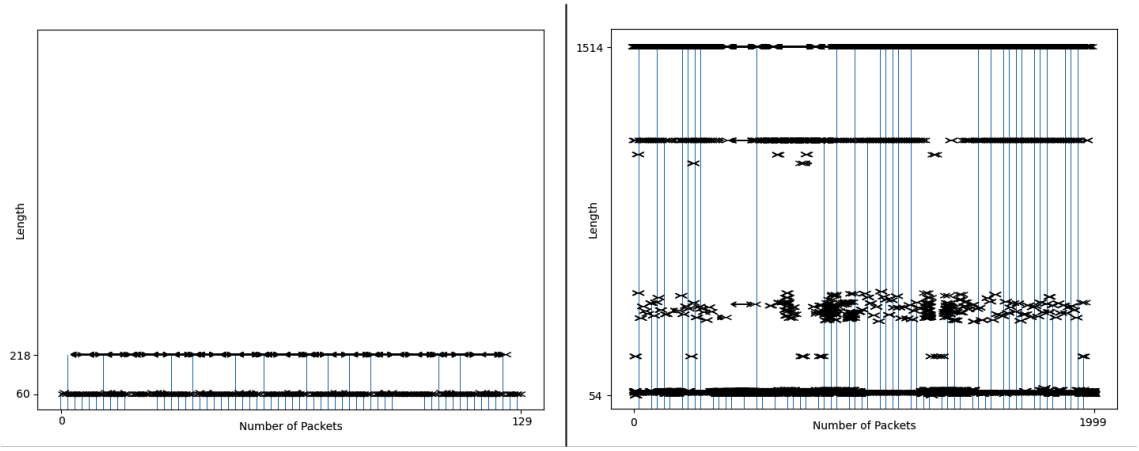

Figure 5.6: HVGs Results - Comparing *Length* Feature (Benign - 130 Packets VS DDoS - 2000 Packets) - Time frame duration of 1 second

The results using HVGs, obtained in Figures 5.5 and 5.6 are similar to the ones analyzed using VGs. DDoS attacks create a considerable increase in the value range of the *Length* feature.

The analysis of Figures 5.7 and 5.8, based on the *Protocol* feature, reveals fewer differences between benign and DDoS traffic. The value of the *Protocol* feature varies between 0 and 1, with the value 0 corresponding to the HTTP protocol and the value 1 corresponding to the TCP protocol.

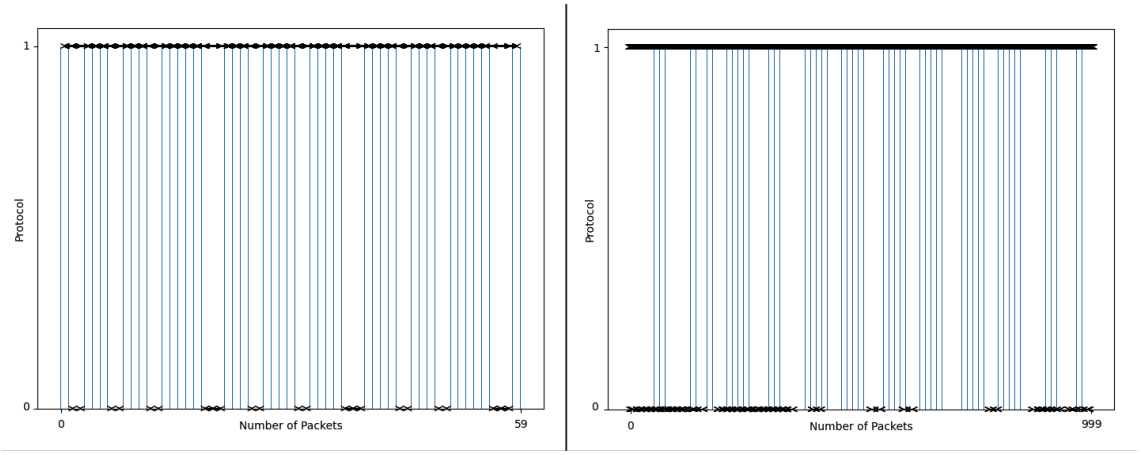

Figure 5.7: HVGs Results - Comparing Feature *Protocol* (Benign - 60 Packets VS DDoS - 1000 Packets) - Time frame duration of 0.45 seconds

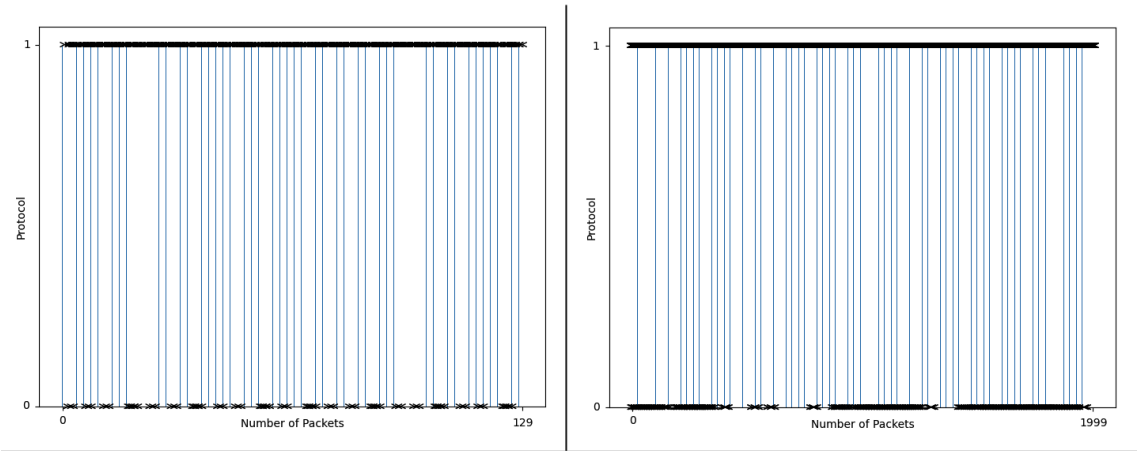

Figure 5.8: HVGs Results - Comparing Feature *Protocol* (Benign - 130 Packets VS DoS - 2000 Packets) - Time frame duration of 1 second

Regarding the results using HVGs depicted in Figures 5.7 and 5.8 it is possible to verify that there is a higher frequency of sent packets in the graphs that represent DDoS attack logs plus a greater oscillation in the exchange of protocols.

Considering all the visual results presented, HVGs plotting based on short periods of network logs enables the identification of DDoS attacks since this traffic shows a different pattern than benign traffic regarding the range of value on the y-axis and frequency of visibility edges. However, comparing the features used, *Length* seems more suitable for identifying these attacks through HVGs.

In summary, the results generated by implementing VGs and HVGs are promising.

It is possible, through them, to detect patterns that differentiate the types of traffic and make it possible to detect anomalous behavior in the network.

In addition to the differentiation of models, which can be detected visually, it is also possible to verify the differences through the properties of the respective VGs, namely regarding the *Average Degree* and *Density* values. The values Average Degree are obtained using the Eq. 5.1 and the values Density are obtained using the Eq. 5.2.

```
1    average_degree = sum(dict(degree).values()) / len(dict(degree))
```

Listing 5.1: Average degree equation

```
1    density = nx.density(graph)
```

Listing 5.2: Density equation

Table 5.6 presents the results of the properties *Average Degree* and *Density* for the use case where five blocks of 200 packets of each of the types of traffic (benign and DDoS) are compared of the Feature Length, after applying VGs. From these results, it can be verified that the values are always higher when DDoS attack traffic is assessed.

Table 5.6: Differentiate traffic using Average Degree and Density

	Average Degree		**Density**	
	Benign	**DDoS**	**Benign**	**DDoS**
1	15.547	66.97	0.078	0.437
2	16.02	77.27	0.080	0.388
3	15.21	50.49	0.0764	0.253
4	16.11	53.31	0.080	0.267
5	15.54	52.37	0.078	0.263

5.3 Implementation

To achieve the expected results in the use case, a pipeline was created using the Python programming language (see Figure 4.7) [1]. This pipeline is similar to the one described in Section 4.3 of Chapter 4.

[1] Appendix C contains the code of this implementation.

1. The first step in the process is to load the dataset consisting of network events generated by benign and malicious nodes focusing on DDoS attacks. After uploading, event records go through a transformation process where they are converted from PCAP format to CSV to facilitate processing, as presented in Listing 5.3.

```
1       from scapy.all import *
2       import csv
3
4   # Read the PCAP file with Scapy
5       packages = rdpcap("caption.pcap")
6
7   # Create CSV file
8       with open("logs.csv", "w") as f:
9           csv_writer = csv.writer(f)
10          csv_writer.writerow(["Timestamp", "Source",
11          "Destination", "Protocol", "Length"])
12
13      # Iterate over each package and write the data to the CSV file
14          for package in packages:
15              timestamp = package.time
16              src = package[IP].src
17              dst = package[IP].dst
18              proto = package[IP].proto
19              length = package[IP].len
20              csv_writer.writerow([timestamp, src, dst,
21              proto, length])
22
23  # Read CSV file
24      df = pd.read_csv('logs.csv')
```

Listing 5.3: First Step - Using VGs to detect DDoS Attacks

2. Then, the data goes through a filtering phase in which the protocol and length features are selected. The data is cleaned and optimized and if the feature protocol is selected, it is automatically converted into numerical values so that it can be processed efficiently. This conversion is performed using the LabelEncoder function from the sklearn.preprocessing Python library. It involves identifying the columns

with categorical or object data types and then looping through those columns with the fit_transform() function. This data preprocessing is essential before the concept of VGs can be applied. This step presented in Listing 5.4.

```
# Transform categorical values to numeric
    def Encoder(df):
      columnsToEncode = list(df.select_dtypes(include=
        ['category','object']))
      le = LabelEncoder()
      for feature in columnsToEncode:
          try:
              df[feature] = le.fit_transform(df[feature])
          except:
              print('Error encoding '+feature)
      return df
    df = Encoder(df)
```

Listing 5.4: Second Step - Using VGs to detect DDoS Attacks

3. Once this step is complete, the values associated with the selected resource are stored in a list of numeric data, as presented in Listing 5.5.

```
# This script points to the column where the VG algorithm
  will be applied and turns it into a list
    ts = df["Protocol"].tolist()
```

Listing 5.5: Third Step - Using VGs to detect DDoS Attacks

4. In the next phase, VGs begin to be created. In this step, as presented in Listing 5.6, ordered pairs of values (x, y) are created that represent values visible to both sides. The main concern here is to identify the edges that are present in VGs.

```
# Necessary libraries are imported
    from matplotlib import pyplot as plt
    import numpy as np
    from visibility_graph import visibility_graph

# Creating ordered pairs of values (x, y)
    result = visibility_graph(ts)
```

Listing 5.6: Fourth Step - Using VGs to detect DDoS Attacks

5. Finally, based on the pairs of ordered values (x, y) defined in the previous step, the corresponding points and radii are drawn on the graph, leading to the formation of VGs, as presented in Listing 5.7.

```
fig = plt.figure()
N = 130
ax = fig.add_axes([0,0,1,1])
ind = np.arange(N)
ax.bar(ind,ts, width=0.1)
ax.set_xticklabels([])
ax.set_yticklabels([])
ax.set_xticks([])
ax.set_yticks([])
ax.set_xlabel('Number of Packets')
ax.set_ylabel('Protocol')
array_length = len(ind)
last_element = ind[array_length - 1]
edgelisteth = list(result.edges)

for i,j in edgelisteth:
  x1, y1 = [i, j], [ts[i], ts[j]]
  plt.plot(x1, y1, color='red')

plt.title("Visibility Graph", fontsize = 15)
plt.xticks(list(ax.get_xticks()) + [min(ind), max(ind)])
plt.yticks(list(ax.get_yticks()) + [min(ts), max(ts)])
plt.tight_layout()
plt.show()
```

Listing 5.7: Fifth Step - Using VGs to detect DDoS Attacks

6. The same occurs for HVGs. Based on the pairs of ordered values (x, y) defined in the previous step, the corresponding points and radii are drawn, leading to the formation of HVGs, as presented in Listing 5.8.

```
fig = plt.figure()
N = 130
ax = fig.add_axes([0,0,1,1])
ind = np.arange(N)
```

```
5     ax.bar(ind,ts, width=0.1)
6     ax.set_xticklabels([])
7     ax.set_yticklabels([])
8     ax.set_xticks([])
9     ax.set_yticks([])
10    ax.set_xlabel('Number of Packets')
11    ax.set_ylabel('Protocol')
12    array_length = len(ind)
13    last_element = ind[array_length - 1]
14    edgelisteth = list(result.edges)
15
16    for i,j in edgelisteth:
17     if ts[i]<ts[j]:
18       plt.annotate(xy=(j,ts[(i)]), xytext=(i,ts[(i)]),
19       arrowprops=dict(arrowstyle='<->'), text='')
20     else:
21       plt.annotate(xy=(j,ts[(j)]), xytext=(i,ts[(j)]),
22       arrowprops=dict(arrowstyle='<->'), text='')
23
24    plt.xticks(list(ax.get_xticks()) + [min(ind), max(ind)])
25    plt.yticks(list(ax.get_yticks()) + [min(ts), max(ts)])
26    plt.tight_layout()
27    plt.show()
```

Listing 5.8: Sixth Step - Using VGs to detect DDoS Attacks

5.4 Summary

From the results presented in Figures 5.1 to 5.8, it can be verified that VGs and HVGs enable to detetc differences in patterns between benign traffic and DDoS attacks. Considering all the results, it is also possible to state that, when comparing the features used the *Lenght* feature appears to be better for identifying anomalous patterns. In any case, the malicious traffic presents a different pattern than benign traffic in relation to the value range on the y-axis, and the frequency of the visibility rays also differs. Thus, it is possible to detect patterns that differentiate the types of traffic and make it possible to detect anomalous behaviors on the network.

Chapter 6

Conclusions

This thesis presents two new approaches that can help in detecting DoS and DDoS attacks. The first approach aims to select features in supervised machine learning using VGs as a technique that requires low computational resources. The second approach aims to detect DoS and DDoS attacks on a network at an early stage, based on representing information from network logs through VGs and HVGs as an easy-to-implement data processing model. The goal is to detect DoS/DDoS attacks in a highly visual way using network traffic patterns.

Both proposals have advantages over traditional approaches. VGs are mathematical structures that visually represent a time series as a complex connected network. This graphical representation facilitates analysis of the underlying properties and dynamics of data, revealing patterns and relationships that can be relevant when used as input features in a supervised ML model. It should be noted that even short intervals of monitoring records produce promising results. In addition, VGs can be easily created with low technological investment in terms of computational resources (compared to most traditional tools). Plotting VGs to represent network logs coming from, for example, a network connection to a web application constitutes a promising approach for detecting patterns in the network. As a positive factor, this early detection allows defensive measures to be activated in the event of an attack.

However, the use of VGs also has some limitations. The processing time required to construct the VGs increases linearly with the size of the data set and, consequently, with the length of the time series. Therefore, when constructing a VG from a extensive time

series, it is expected that the graph creation time grows linearly with the number of data points.

In summary, the proposals to use VGs as a feature selection tool in supervised ML and to use VGs and HVGs as a way of early detection of DoS/DDoS attacks seem promising and open the way for new research and applications in several domains. Such as the detection of other network attacks or even to detect fraud in payment transactions. Analysis of the complex networks that VGs produce in terms of degree distribution and other complex network parameters could contribute to additional ways of identifying patterns of different types of events.

References

[1] Lalit B Abbas, Muhammad A Sadiq, and Mohsin O Ahmad. "Machine learning-based detection of DDoS attacks: A review". In: *Future Generation Computer Systems* 111 (2020), pp. 799–811.

[2] Iftikhar Ahmad, Azween B. Abdullah, and Abdullah S. Alghamdi. "Application of Artificial Neural Network in Detection of DOS Attacks". In: *Proceedings of the 2nd International Conference on Security of Information and Networks*. SIN '09. Famagusta, North Cyprus: Association for Computing Machinery, 2009, pp. 229–234. ISBN: 9781605584126. DOI: 10.1145/1626195.1626252. URL: https://doi.org/10.1145/1626195.1626252.

[3] Mehran Ahmadlou, Hojjat Adeli, and Amir Adeli. "Improved visibility graph fractality with application for the diagnosis of autism spectrum disorder". In: *Physica A: Statistical Mechanics and its Applications* 391.20 (2012), pp. 4720–4726.

[4] Ethem Alpaydin. *Introduction to machine learning*. Cambridge, MA: MIT Press, 2010.

[5] Alaitz Aranburu-Imatz et al. "Environmental pollution in North-Eastern Italy and its influence on chronic obstructive pulmonary disease: time series modelling and analysis using visibility graphs". In: *Air Quality, Atmosphere & Health* 16 (2023), pp. 793–804. DOI: 10.1007/s11869-023-01310-7.

[6] Asmaa Shaker Ashoor and Sharad Gore. "Importance of intrusion detection system (IDS)". In: *International Journal of Scientific and Engineering Research* 2.1 (2011), pp. 1–4.

[7] Ekele A. Asonye, Ifeoma Anwuna, and Sarhan M. Musa. "Securing ZigBee IoT Network Against HULK Distributed Denial of Service Attack". In: *2020 IEEE 17th International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET)*. 2020, pp. 156–162. DOI: 10.1109/HONET50430.2020.9322808.

[8] Nabeel Z. Bawany, Jawad A. Shamsi, and Khaled Salah. "DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions". In: *Arabian Journal for Science and Engineering* 42 (2017), pp. 425–441. DOI: 10.1007/s13369-017-2414-5.

[9] Steven M Bellovin and William R Cheswick. "Network firewalls". In: *IEEE communications magazine* 32.9 (1994), pp. 50–57.

[10] Andreea Bendovschi. "Cyber-Attacks – Trends, Patterns and Security Countermeasures". In: *Procedia Economics and Finance* 28 (2015). 7th INTERNATIONAL CONFERENCE ON FINANCIAL CRIMINOLOGY 2015, 7th ICFC 2015, 13-14 April 2015,Wadham College, Oxford University, United Kingdom, pp. 24–31. ISSN: 2212-5671. DOI: https://doi.org/10.1016/S2212-5671(15)01077-1. URL: https://www.sciencedirect.com/science/article/pii/S2212567115010771.

[11] James Bergstra and Yoshua Bengio. "Random search for hyper-parameter optimization". In: *International Conference on Learning Representations*. 2012.

[12] Filippo Bianchi et al. "Multiplex visibility graphs to investigate recurrent neural network dynamics". In: *Scientific Reports* 7 (2017), p. 44037. DOI: 10.1038/srep44037.

[13] Christopher Brown. "Data Division Strategies in Machine Learning". In: *Proceedings of the International Conference on Machine Learning*. 2017, pp. 234–245.

[14] G. Carl et al. "Denial-of-service attack-detection techniques". In: *IEEE Internet Computing* 10.1 (2006), pp. 82–89. DOI: 10.1109/MIC.2006.5.

[15] Brian Cashell et al. "The economic impact of cyber-attacks". In: *Congressional research service documents, CRS RL32331 (Washington DC)* 2 (2004).

[16] Brad Chacos. "Major ddos attack on dyn dns knocks spotify, twitter, github, paypal, and more offline". In: *PCWorld. PCWorld* 21 (2016).

[17] Varun Chandola, Arindam Banerjee, and Vipin Kumar. “Anomaly detection: A survey”. In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[18] Michał Choraś and Rafał Kozik. *Machine learning techniques applied to detect cyber attacks on web applications*. 2015.

[19] Paolo Dal Cin and Jeremy Jurgens. *Global Cybersecurity Outlook 2023*. 2023. URL: https://www.weforum.org/reports/global-cybersecurity-outlook-2023/.

[20] Cloudfare. *Cloudflare DDoS threat report for 2022 Q4*. 2023. URL: https://blog.cloudflare.com/ddos-threat-report-2022-q4/.

[21] Corinna Cortes. “Support-Vector Networks”. In: *Machine Learning* 20.3 (1995), pp. 273–297. DOI: 10.1007/bf00994018.

[22] Thomas Cover and Peter Hart. “Nearest neighbor pattern classification”. In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27. DOI: 10.1109/TIT.1967.1053964.

[23] Pedro Domingos. “A Few Useful Things to Know About Machine Learning”. In: *Communications of the ACM* 55.10 (2012), pp. 78–87.

[24] Stefan Falkner, Aaron Klein, and Frank Hutter. “BOHB: Robust and efficient hyperparameter optimization at scale”. In: *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 1436–1445.

[25] L. Feinstein et al. “Statistical approaches to DDoS attack detection and response”. In: *Proceedings DARPA Information Survivability Conference and Exposition*. Vol. 1. 2003, 303–314 vol.1. DOI: 10.1109/DISCEX.2003.1194894.

[26] Ronald A. Fisher. “The use of multiple measurements in taxonomic problems”. In: *Annals of Eugenics* 7.2 (1936), pp. 179–188. DOI: 10.1111/j.1469-1809.1936.tb02137.x.

[27] Abdul Gani, Saif Ullah, and Khurram Khan. “Detection of Denial of Service (DoS) Attacks Using Machine Learning Techniques”. In: *2019 International Conference on Computer and Information Sciences (ICCIS)*. IEEE. 2019, pp. 1–6.

[28] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[29] Maria Gonzalez. *Algorithm Applications in Machine Learning*. Springer, 2019.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[31] B. B. Gupta and Omkar P. Badve. "Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a Cloud computing environment". In: *Neural Computing and Applications* 28.12 (2017), pp. 3655–3682. ISSN: 1433-3058. DOI: 10.1007/s00521-016-2317-5.

[32] Bhavesh Gupta, Ritu Gupta, and Sandeep Kumar Tyagi. "Taxonomy of DDoS attacks and their prevention techniques: a review". In: *Journal of Network and Computer Applications* 126 (2019), pp. 48–73. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2018.10.009. URL: https://doi.org/10.1016/j.jnca.2018.10.009.

[33] Roman Hajtmanek et al. "One-Parameter Statistical Methods to Recognize DDoS Attacks". In: *Symmetry* 14.11 (2022). ISSN: 2073-8994. DOI: 10.3390/sym14112388. URL: https://www.mdpi.com/2073-8994/14/11/2388.

[34] Stefan Iovan and Alina-Anabela Iovan. "From cyber threats to cyber-crime". In: *Journal of Information Systems & Operations Management* (2016), p. 425.

[35] S M Riazul Islam et al. "Detecting DDoS Attacks with Machine Learning Techniques". In: *Information Sciences* 254 (2014), pp. 1–14.

[36] Neyole Misiko Jacob and Muchelule Yusuf Wanjala. "A review of intrusion detection systems". In: *Global Journal of Computer Science and Information Technology Research. Framingham: Global Journals Inc* 5.4 (2017), pp. 1–5.

[37] Mark Johnson and Linda Smith. "Visibility Graphs: A Survey". In: *IEEE Transactions on Visualization and Computer Graphics* 21.8 (2015), pp. 933–952.

[38] Michael Jones and Emily Brown. "Data Pre-processing Techniques in Machine Learning". In: *International Journal of Data Science* 8.2 (2016), pp. 789–804.

[39] Jalindar Karande and Sarang Joshi. "Real-Time Detection of Cyber Attacks on the IoT Devices". In: *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. 2020, pp. 1–6. DOI: 10.1109/ICCCNT49239.2020.9225487.

[40] John D Kelleher, Brian Tierney, and Brian Tierney. *Data science: An introduction*. 2nd. CRC Press, 2018. Chap. 5.

[41] Tran Viet Khoa et al. "Collaborative Learning Model for Cyberattack Detection Systems in IoT Industry 4.0". In: *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. 2020, pp. 1–6. DOI: 10.1109/WCNC45663.2020.9120761.

[42] Lucas Lacasa et al. "From time series to complex networks: The visibility graph". In: *Proceedings of the National Academy of Sciences* 105.13 (2008), pp. 4972–4975.

[43] Keunsoo Lee et al. "DDoS attack detection method using cluster analysis". In: *Expert Systems with Applications* 34.3 (2008), pp. 1659–1665. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2007.01.040. URL: https://www.sciencedirect.com/science/article/pii/S0957417407000395.

[44] Jie Liu and Jinghua Chen. "Visibility Graphs for Analyzing Complex Systems: A Review". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28.4 (2018), p. 041101.

[45] Simon Liu and Bruce Cheng. "Cyberattacks: Why, What, Who, and How". In: *IT Professional* 11.3 (2009), pp. 14–21. DOI: 10.1109/MITP.2009.46.

[46] Yu Long. "Visibility graph network analysis of gold price time series". In: *Physica A: Statistical Mechanics and its Applications* 392.16 (2013), pp. 3374–3384. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2013.03.063. URL: https://www.sciencedirect.com/science/article/pii/S0378437113002963.

[47] YB Luo et al. "FL-LPVG: an approach for anomaly detection based on flow-level limited penetrable visibility graph". In: (2013).

[48] Xin Man and Ernest Chan. "The Best Way to Select Features? Comparing MDA, LIME, and SHAP". In: *The Journal of Financial Data Science* 3.1 (2021), pp. 127–

139. DOI: 10.3905/jfds.2020.1.047. URL: https://ssrn.com/abstract=3880618.

[49] Sujith Mangalathu, Seong-Hoon Hwang, and Jong-Su Jeon. "Failure mode and effects analysis of RC members based on machine-learning-based SHapley Additive exPlanations (SHAP) approach". In: *Engineering Structures* 219 (2020), p. 110927. ISSN: 0141-0296. DOI: 10.1016/j.engstruct.2020.110927. URL: https://www.sciencedirect.com/science/article/pii/S0141029620307513.

[50] CIO Marco Antonio Cavallo. *The growing importance of the technology economy.* 2016. URL: https://www.cio.com/article/236899/the-growing-importance-of-the-technology-economy.html.

[51] Carlo Mari and Cristiano Baldassari. "Optimization of mixture models on time series networks encoded by visibility graphs: an analysis of the US electricity market". In: *Computational Management Science* 20.1 (2023), p. 28.

[52] Jeferson Martínez and Javier M Duran. "Software supply chain attacks, a threat to global cybersecurity: SolarWinds' case study". In: *International Journal of Safety and Security Engineering* 11.5 (2021), pp. 537–545.

[53] TIM MAURER and ARTHUR NELSON. *Cyber threats to the financial system are growing, and the global community must cooperate to protect it.* 2021. URL: https://www.imf.org/external/pubs/ft/fandd/2021/03/global-cyber-threat-to-financial-systems-maurer.htm.

[54] Andrew McCallum and Kamal Nigam. "A Comparison of Event Models for Naive Bayes Text Classification". In: *AAAI-98 Workshop on Learning for Text Categorization.* Vol. 752. 1998, pp. 41–48.

[55] Megha Mittal, Kuldip Kumar, and Sukhpreet Behal. "Deep learning approaches for detecting DDoS attacks: a systematic review". In: *Soft Computing* 27 (2023), pp. 13039–13075. DOI: 10.1007/s00500-021-06608-1.

[56] Andreas C Müller and Sarah Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists.* O'Reilly Media, 2016.

[57] M. Narasimha Murty and Xiang Li. *Support Vector Machines and Perceptrons: Learning, Optimization, Classification, and Application to Social Networks*. Springer, 2014.

[58] Anthony J Myles et al. "An introduction to decision tree modeling". In: *Journal of Chemometrics: A Journal of the Chemometrics Society* 18.6 (2004), pp. 275–285.

[59] Saritakumar N and Anusuya K V. "Early Detection and Mitigation of DoS Attacks in SDN Controller". In: *2022 International Conference on Intelligent Innovations in Engineering and Technology (ICIIET)*. 2022, pp. 315–322. DOI: 10.1109/ICIIET55458.2022.9967650.

[60] Vladimir Nasteski. "An overview of the supervised machine learning methods". In: *HORIZONS.B* 4 (Dec. 2017), pp. 51–62. DOI: 10.20544/HORIZONS.B.04.1.17.P05.

[61] Kangyu Ni et al. "Characterizing Disease Spreading via Visibility Graph Embedding". In: *2021 IEEE International Conference on Big Data (Big Data)*. 2021, pp. 2656–2661. DOI: 10.1109/BigData52589.2021.9671810.

[62] Shunsuke Oshima, Takuo Nakashima, and Toshinori Sueyoshi. "Early DoS/ DDoS Detection Method using Short-term Statistics". In: *2010 International Conference on Complex, Intelligent and Software Intensive Systems*. 2010, pp. 168–173. DOI: 10.1109/CISIS.2010.53.

[63] Sushma Devi Parmar. "Cybersecurity in India: An evolving concern for national security". In: *The Journal of Intelligence and Cyber Security* 1.1 (2018).

[64] Alberto Partida et al. "The chaotic, self-similar and hierarchical patterns in Bitcoin and Ethereum price series". In: *Chaos, Solitons & Fractals* 165 (2022), p. 112806. ISSN: 0960-0779. DOI: https://doi.org/10.1016/j.chaos.2022.112806. URL: https://www.sciencedirect.com/science/article/pii/S0960077922009857.

[65] Jorge O. Pierini, Michele Lovallo, and Luciano Telesca. "Visibility graph analysis of wind speed records measured in central Argentina". In: *Physica A: Statistical Mechanics and its Applications* 391.20 (2012), pp. 5041–5048. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2012.05.049. URL: https://www.sciencedirect.com/science/article/pii/S0378437112004207.

[66] Md Mamunur Rashid et al. “Cyberattacks detection in iot-based smart city applications using machine learning techniques”. In: *International Journal of environmental research and public health* 17.24 (2020), p. 9347.

[67] Andy Reed. “HTTP DoS Dataset in PCAP format for Wireshark”. In: (June 2022). DOI: 10.21954/ou.rd.17206289.v2. URL: https://ordo.open.ac.uk/articles/dataset/HTTP_DoS_Dataset_in_PCAP_format_for_Wireshark/17206289.

[68] S. RoselinMary, M. Maheshwari, and M. Thamaraiselvan. “Early detection of DOS attacks in VANET using Attacked Packet Detection Algorithm (APDA)”. In: *2013 International Conference on Information Communication and Embedded Systems (ICICES)*. 2013, pp. 237–240. DOI: 10.1109/ICICES.2013.6508250.

[69] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0.

[70] Ana Šarčević et al. “Cybersecurity Knowledge Extraction Using XAI”. In: *Applied Sciences* 12.17 (2022). DOI: 10.3390/app12178669. URL: https://www.mdpi.com/2076-3417/12/17/8669.

[71] Neil Shah. “FARE: Schema-Agnostic Anomaly Detection in Social Event Logs”. In: *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 2019, pp. 337–350. DOI: 10.1109/DSAA.2019.00049.

[72] Tanishka Shorey et al. “Performance Comparison and Analysis of Slowloris, GoldenEye and Xerxes DDoS Attack Tools”. In: *2018 International Conference on Advances in Computing, Communications and Informatics (ICA CCI)*. 2018, pp. 318–322. DOI: 10.1109/ICACCI.2018.8554590.

[73] John Smith and Sarah Johnson. “Data Collection for Machine Learning”. In: *Journal of Machine Learning Research* 12.4 (2018), pp. 1234–1256.

[74] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. “Practical bayesian optimization of machine learning algorithms”. In: *International Conference on Neural Information Processing Systems*. 2012.

[75] Emmanouil G. Spanakis et al. “Cyber-attacks and threats for healthcare – a multi-layer thread analysis”. In: *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. 2020, pp. 5705–5708. DOI: 10.1109/EMBC44109.2020.9176698.

[76] Newton Spolaôr et al. “ReliefF for Multi-label Feature Selection”. In: *2013 Brazilian Conference on Intelligent Systems*. 2013, pp. 6–11. DOI: 10.1109/BRACIS.2013.10.

[77] Victoria Stanciu and Andrei Tinca. “Exploring cybercrime–realities and challenges”. In: *Accounting and Management Information Systems* 16.4 (2017), pp. 610–632.

[78] K.Muthamil Sudar et al. “Analysis of Cyberattacks and its Detection Mechanisms”. In: *2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. 2020, pp. 12–16.

[79] Lin Sun et al. “Multilabel feature selection using ML-ReliefF and neighborhood mutual information for multilabel neighborhood decision systems”. In: *Information Sciences* 537 (2020), pp. 401–424. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2020.05.102. URL: https://www.sciencedirect.com/science/article/pii/S0020025520305211.

[80] Zhiyuan Tan et al. “A System for Denial-of-Service Attack Detection Based on Multivariate Correlation Analysis”. In: *IEEE Transactions on Parallel and Distributed Systems* 25.2 (2014), pp. 447–456. DOI: 10.1109/TPDS.2013.146.

[81] Zhiyuan Tan et al. “Denial-of-Service Attack Detection Based on Multivariate Correlation Analysis”. In: *Neural Information Processing*. Ed. by Bao-Liang Lu, Liqing Zhang, and James Kwok. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 756–765. ISBN: 978-3-642-24965-5.

[82] Zhiyuan Tan et al. “Detection of Denial-of-Service Attacks Based on Computer Vision Techniques”. In: *IEEE Transactions on Computers* 64.9 (2015), pp. 2519–2533. DOI: 10.1109/TC.2014.2375218.

[83] Diogo Teixeira et al. “OSSEC IDS Extension to Improve Log Analysis and Override False Positive or Negative Detections”. In: *Journal of Sensor and Actuator Networks*

8.3 (2019). ISSN: 2224-2708. DOI: 10.3390/jsan8030046. URL: https://www.mdpi.com/2224-2708/8/3/46.

[84] Dimitrios Tsiotas and Lykourgos Magafas. "The Effect of Anti-COVID-19 Policies on the Evolution of the Disease: A Complex Network Analysis of the Successful Case of Greece". In: *Physics* 2.2 (2020), pp. 325–339. ISSN: 2624-8174. URL: https://www.mdpi.com/2624-8174/2/2/17.

[85] Michail D Vamvakaris, Athanasios A Pantelous, and Konstantin M Zuev. "Time series analysis of S&P 500 index: A horizontal visibility graph approach". In: *Physica A: Statistical Mechanics and its Applications* 497 (2018), pp. 41–51.

[86] Sumeet S. Vernekar and Amar Buchade. "MapReduce based log file analysis for system threats and problem identification". In: *2013 3rd IEEE International Advance Computing Conference (IACC)*. 2013, pp. 831–835. DOI: 10.1109/IAdCC.2013.6514334.

[87] T. Vinnakota. "A Second Order Cybernetic Model for Governance of Cyber Security in Enterprises". In: *2016 IEEE 6th International Conference on Advanced Computing (IACC)*. 2016, pp. 706–710. DOI: 10.1109/IACC.2016.136.

[88] Haining Wang, Danlu Zhang, and K.G. Shin. "Change-point monitoring for the detection of DoS attacks". In: *IEEE Transactions on Dependable and Secure Computing* 1.4 (2004), pp. 193–208. DOI: 10.1109/TDSC.2004.34.

[89] Ping Wang and Christopher Johnson. "Cybersecurity incident handling: a case study of the Equifax data breach." In: *Issues in Information Systems* 19.3 (2018).

[90] Zongjian Wang and Xiaobo Li. "Intrusion prevention system design". In: *Proceedings of the International Conference on Information Engineering and Applications (IEA) 2012: Volume 3*. Springer. 2013, pp. 375–382.

[91] Warda. *Application-Layer DDoS Dataset*. https://www.kaggle.com/datasets/wardac/applicationlayer-ddos-dataset?select=test_mosaic.csv. 2020.

[92] Miguel Gomes Xavier, Marcelo Veiga Neves, and Cesar Augusto Fonticielha De Rose. "A Performance Comparison of Container-Based Virtualization Systems for MapReduce Clusters". In: *2014 22nd Euromicro International Conference on Par-*

allel, Distributed, and Network-Based Processing. 2014, pp. 299–306. DOI: 10.1109/PDP.2014.78.

[93] Yi Xie and Shun-Zheng Yu. "Monitoring the Application-Layer DDoS Attacks for Popular Websites". In: *IEEE/ACM Transactions on Networking* 17.1 (2009), pp. 15–25. DOI: 10.1109/TNET.2008.925628.

[94] Aidong Xu et al. "Graph-based time series edge anomaly detection in smart grid". In: *2021 7th IEEE Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. IEEE. 2021, pp. 1–6.

[95] Kun Yang et al. "DDoS Attacks Detection with AutoEncoder". In: *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. 2020, pp. 1–9. DOI: 10.1109/NOMS47738.2020.9110372.

[96] Temechu Girma Zewdie and Anteneh Girma. "An Evaluation Framework for Machine Learning Methods in Detection of DoS and DDoS Intrusion". In: *2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. 2022, pp. 115–121. DOI: 10.1109/ICAIIC54071.2022.9722661.

[97] Yiyun Zhou et al. "Deep learning approach for cyberattack detection". In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2018, pp. 262–267. DOI: 10.1109/INFCOMW.2018.8407032.

[98] Li Zhu et al. "Decoding cortical brain states from widefield calcium imaging data using visibility graph". In: *Biomed. Opt. Express* 9.7 (2018), pp. 3017–3036. DOI: 10.1364/BOE.9.003017. URL: https://opg.optica.org/boe/abstract.cfm?URI=boe-9-7-3017.

[99] Enyu Zhuang, Michael Small, and Gang Feng. "Time series analysis of the developed financial markets' integration using visibility graphs". In: *Physica A: Statistical Mechanics and its Applications* 410 (2014), pp. 483–495. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2014.05.058. URL: https://www.sciencedirect.com/science/article/pii/S0378437114004397.

Appendices

Appendix A

Dataset Columns of Use Case (Section 4.2) of Chapter 4

Dataset columns	
1 - Destination_Port	40 - Max_Packet_Length
2 - Flow_Duration	41 - Packet_Length_Mean
3 - Total_Fwd_Packets	42 - Packet_Length_Std
4 - Total_Backward_Packets	43 - Packet_Length_Variance
5 - Total_Length_of_Fwd_Packets	44 - FIN_Flag_Count
6 - Total_Length_of_Bwd_Packets	45 - SYN_Flag_Count
7 - Fwd_Packet_Length_Max	46 - RST_Flag_Count
8 - Fwd_Packet_Length_Min	47 - PSH_Flag_Count
9 - Fwd_Packet_Length_Mean	48 - ACK_Flag_Count
10 - Fwd_Packet_Length_Std	49 - URG_Flag_Count
11 - Bwd_Packet_Length_Max	50 - CWE_Flag_Count
12 - Bwd_Packet_Length_Min	51 - ECE_Flag_Count
13 - Bwd_Packet_Length_Mean	52 - Down_Up_Ratio
14 - Bwd_Packet_Length_Std	53 - Average_Packet_Size
15 - Flow_Bytes_Sec	54 - Avg_Fwd_Segment_Size
16 - Flow_Packets_Sec	55 - Avg_Bwd_Segment_Size

17 - Flow_IAT_Mean	56 - Fwd_Avg_Bytes_Bulk
18 - Flow_IAT_Std	57 - Fwd_Avg_Packets_Bulk
19 - Flow_IAT_Max	58 - Fwd_Avg_Bulk_Rate
20 - Flow_IAT_Min	59 - Bwd_Avg_Bytes_Bulk
21 - Fwd_IAT_Total	60 - Bwd_Avg_Packets_Bulk
22 - Fwd_IAT_Mean	61 - Bwd_Avg_Bulk_Rate
23 - Fwd_IAT_Std	62 - Subflow_Fwd_Packets
24 - Fwd_IAT_Max	63 - Subflow_Fwd_Bytes
25 - Fwd_IAT_Min	64 - Subflow_Bwd_Packets
26 - Bwd_IAT_Total	65 - Subflow_Bwd_Bytes
27 - Bwd_IAT_Mean	66 - Init_Win_bytes_forward
28 - Bwd_IAT_Std	67 - Init_Win_bytes_backward
29 - Bwd_IAT_Max	68 - act_data_pkt_fwd
30 - Bwd_IAT_Min	69 - min_seg_size_forward
31 - Fwd_PSH_Flags	70 - Active_Mean
32 - Bwd_PSH_Flags	71 - Active_Std
33 - Fwd_URG_Flags	72 - Active_Max
34 - Bwd_URG_Flags	73 - Active_Min
35 - Fwd_Header_Length	74 - Idle_Mean
36 - Bwd_Header_Length	75 - Idle_Std
37 - Fwd_Packets_Sec	76 - Idle_Max
38 - Bwd_Packets_Sec	77 - Idle_Min
39 - Min_Packet_Length	78 - Label (target)

Table A.1: Columns of the dataset used for the use case

Appendix B

Python code of the pipeline from Section 4.3 of Chapter 4

```
import pandas as pd

# Read CSV File
df = pd.read_csv('logs/test_mosaic.csv')

# Groups rows by the value of the 'Label' column
grouped = df.groupby('Label')

# Saves each group to a separate CSV file
for name, group in grouped:
    group.to_csv(f'{name}.csv', index=False)

df = pd.read_csv("BENIGN.csv")
df

# If want to search only the first 50 lines of the CSV
df = df.head(50)
```

```
# Points to the column where the VG algorithm will be applied and turns
it into a list
ts = df["Total_Fwd_Packets"].tolist()

from matplotlib import pyplot as plt
import numpy as np

from visibility_graph import visibility_graph

result = visibility_graph(ts)

fig = plt.figure()
N = 50
ax = fig.add_axes([0,0,1,1])
ind = np.arange(N)

ax.bar(ind,ts, width=0.1)
ax.set_xticklabels([])
ax.set_yticklabels([])
ax.set_xticks([])
ax.set_yticks([])
array_length = len(ind)
last_element = ind[array_length - 1]

edgelisteth = list(result.edges)

for i,j in edgelisteth:
  x1, y1 = [i, j], [ts[i], ts[j]]
  plt.plot(x1, y1, color='red')

plt.title("Visibility Graph", fontsize = 15)
```

```
plt.savefig('images/test.png')

plt.show()
```

Appendix C

Python code of the pipeline from Section 5.3 of Chapter 5

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Read CSV File
df = pd.read_csv('../logs.csv')
df

# Read only the lines with the IPs we want (according to the dataset
xml document)
df = df[(df['Source'] == '192.168.1.68') | (df['Source'] == '192.168.2.83')
| (df['Source'] == '192.168.2.183') | (df['Source'] == '192.168.10.124')]

# Export csv for analysis
df.to_csv("df_to_check.csv", index=False)

# Read only logs associated with an IP (according to the dataset xml
document)
df = df[(df['Source'] == '192.168.1.68')]
```

```
# Turn categorical values into numeric
def Encoder(df):
          columnsToEncode =
            list(df.select_dtypes(include=['category','object']))
          le = LabelEncoder()
          for feature in columnsToEncode:
              try:
                  df[feature] = le.fit_transform(df[feature])
              except:
                  print('Error encoding '+feature)
          return df
df = Encoder(df)

#Points to the column where the VG algorithm will be applied and turns
it into a list
ts = df["Protocol"].tolist()

from matplotlib import pyplot as plt
import numpy as np

from visibility_graph import visibility_graph

result = visibility_graph(ts)

fig = plt.figure()
N = 50
ax = fig.add_axes([0,0,1,1])
ind = np.arange(N)

ax.bar(ind,ts, width=0.1)
```

```
ax.set_xticklabels([])
ax.set_yticklabels([])
ax.set_xticks([])
ax.set_yticks([])
ax.set_xlabel('Number of Packets')
ax.set_ylabel('Protocol')
array_length = len(ind)
last_element = ind[array_length - 1]


edgelisteth = list(result.edges)


for i,j in edgelisteth:
  x1, y1 = [i, j], [ts[i], ts[j]]
  plt.plot(x1, y1, color='red')


plt.title("Visibility Graph", fontsize = 15)


plt.xticks(list(ax.get_xticks()) + [min(ind), max(ind)])
plt.yticks(list(ax.get_yticks()) + [min(ts), max(ts)])


plt.tight_layout()
plt.savefig('images/protocol.png', bbox_inches='tight')
plt.show()
```