Lebanese University, Faculty of Sciences

Hassan ABDALLAH

Thesis for the Acquisition of a Master's Degree in Data Science for Risk Analysis

Graph-based cyber-attacks detection in IoT networks

September 2022

Under the supervision of:

Prof. Ali Jaber

Jury:

Dr. Yasser Fadlallah

Dr. Houssein Alaeddine

Abstract

An IoT system is made up of smart devices that are linked together to collect and send data as well as act on it, due to resource constraints, IoT systems are vulnerable to a variety of vulnerabilities. Despite the effectiveness of the existing Intrusion Detection Systems, IoT environments restrict the use of some security features and systems. In this work we made two contributions, the first contribution was state-of-the-art, where we discussed various studies containing several techniques to detect attacks and anomalies in order to improve security in IoT networks, highlighted the challenges when building an intrusion detection approach in IoT, then we focused on the graph-based techniques which in general gave good results. The second contribution was an approach to detecting cyber-attacks in IoT networks in real-time, our approach has achieved 99% accuracy. After comparing with three existing approaches, we noticed that ours outperformed them.

Contents

List of Figures

Chapter 1

State of the Art

1.1 Introduction

More and more connected objects are present in our daily life. An IoT system is made up of smart devices that are linked together to collect and send data as well as act on it. IoT has emerged as one of the most important technologies in recent years. Now that everyday objects such as kitchen appliances, cars, thermostats, and baby monitors can be connected to the internet via embedded devices, seamless communication between people, processes, and things is possible. Physical things can share and collect data with minimal human intervention thanks to low-cost computing, the cloud, big data analytics, and mobile technologies. As a result, the physical and digital worlds collide and collaborate.

Due to device and technology heterogeneity, as well as resource constraints, IoT security fails to keep up with the rapid development of IoT technologies, leaving IoT systems vulnerable to a variety of vulnerabilities (Jia et al. 2018). Digital systems can record, monitor, and adjust each interaction between connected things in this hyper-connected world, these digital systems are called anti-intrusion systems, as shown in Fig 1.1 they are composed of three main categories: Intrusion detection systems (IDS), intrusion prevention systems (IPS), and Intrusion Response systems (IRS). An intrusion detection system (IDS) is a hardware or software system that monitors a network for malicious events, attacks, or policy violations. To put it another way, it scans a network or a system for malicious activity or policy violations, then notifies the administrator of any malicious activity found, or it collects the data centrally using a security information and event management (SIEM) system. When organizations want to install an intrusion detection system, it must first be properly set up, it means they need to fine-tune their IDS to be able to recognize the traffics that looks like a malicious event, to reduce the number of false alarms. As shown in Fig 1.2 there are two popular types of Intrusion detection

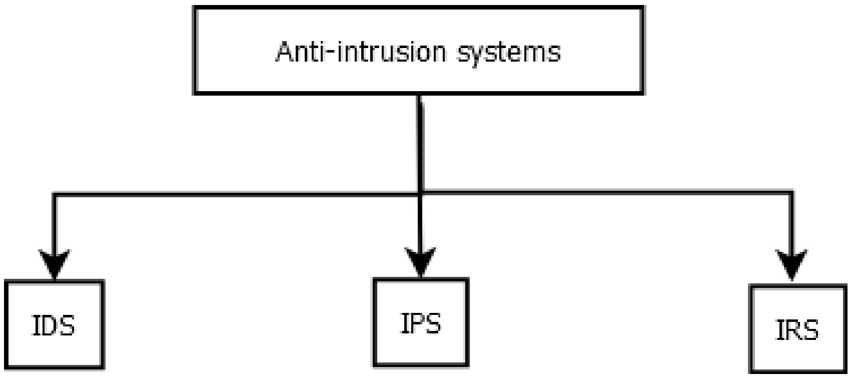


Figure 1.1: Types of anti-intrusion systems

systems and two popular detection methods. Firstly, the Network Intrusion Detection System (NIDS), NIDS is typically installed at a predetermined point in the network to analyze traffic from all devices on the network, it monitors the traffic of the entire subnet and compares it to a list of known attacks. The alert can be sent to the administrator once an attack has been detected or abnormal behavior has been observed. Secondly, the Host Intrusion Detection System (HIDS) is a network security system that monitors incoming and outgoing packets from a single device and alerts the administrator if suspicious or malicious activity is detected. Also, there are two popular detection methods of IDS. Firstly the signature-based Method, signature-based IDS detects attacks based on specific patterns in network traffic, such as sequences of bytes or numbers of 1s or 0s, and it also detects attacks based on previously known malicious instruction sequences used by malware. The patterns detected in IDS are known as signatures. Signature-based IDS can quickly detect attacks with a known pattern (signature), but new malware attacks are more difficult to detect because their pattern (signature) is unknown. Secondly the anomaly-based method. As new malware is generated at a rapid rate, anomaly-based IDS was designed to identify unknown malware threats. Machine learning is used in anomaly-based IDS to construct a trustworthy activity model, and anything that comes in is compared to that model, and it is considered suspicious or normal. In comparison to signature-based IDS, machine learning-based methods have a superior generic property because these models may be taught according to the applications.

Some intrusion detection systems can respond to intrusions as soon as they are discovered, intrusion prevention systems (IPS) are what they're called. An IPS is a hardware or software network security instrument that monitors a network for harmful activity and takes appropriate preventive action, such as reporting, blocking, or dropping it. It goes beyond an intrusion detection system (IDS), which can only detect malicious activity and notify an administrator. Manual monitoring and response aren't an option in a modern network, which has a lot of access points and deals with a lot of traffic, so an intrusion prevention system (IPS) is an important component of any security system. Furthermore, the threats to security systems are becoming increasingly numerous and sophisticated, in this case, an IPS's automated capabilities are critical, allowing the system to respond to threats swiftly. IPS, on the other hand, need high-performance systems and is difficult to manage in terms of analyzing and preventing intrusions at the same time, particularly in a distributed context. To efficiently identify and mitigate potential events, a security countermeasure that continuously monitors system performance is required. Intrusion response systems (IRS) are the countermeasures in question, the IRS is in charge of detecting intrusions and mitigating their effects (Inayat et al. 2016).

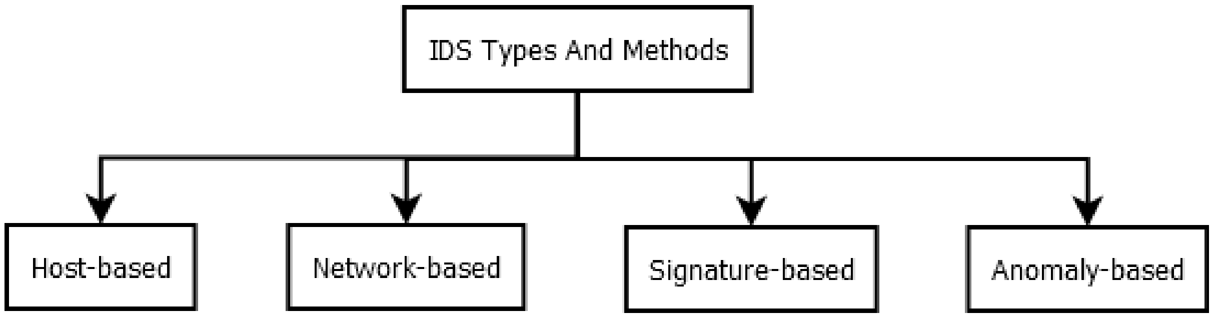


Figure 1.2: Types and methods of the intrusion detection systems

Despite the effectiveness of the aforementioned digital security systems, the unique characteristics of IoT environments, such as limited computing power and storage capacity, limit the application of some of the security systems and features (Alhajri et al. 2019). As a result of the computation and memory constraints of IoT devices, traditional security systems aren't up to the task of detecting attacks. Therefore, We can use log files or packet capture files captured by softwares such as Wireshark in order to improve system security by detecting attacks and malicious events. These files contain a massive amount of information about almost all of the system's events. As a result, it is the primary source of information for locating relevant security information. The traditional method of detecting security issues necessitated a manual examination of the files by an expert (Svacina et al. 2020). However, this method becomes infeasible due to the large amount of data stored, which leads to taking a lot of time in the data analysis. As a result, it's critical to develop a method for automating the file analysis process in order to detect cyber attacks. In this work, we discussed various studies containing several techniques to detect attacks and anomalies in order to improve security in IoT systems and networks, then we have focused on the graph_based techniques which in general gave a good result, then we explain our proposed approach, present the experiment, and evaluate the results.

1.2 Existing surveys

As shown in table 1.1, there are many studies on the topic of anomaly detection, but few focus on or contain a graph_based technique in order to improve the security of an IoT system. A study focused on machine learning and anomaly detection for log file analysis (Svacina et al. 2020), in this study, they analyzed papers and identified relevant characteristics as well as common techniques such as machine learning, data mining, and text analysis. They then identified significant challenges in real-time analysis, universal log formatting, and multisource analysis. Articles on web systems, cloud computing, and the Internet of Things are among their findings. In another paper, Deorankar and Thakare

Table 1.1: Previous survey and reviews

Ref	Title	years	datasets	contains graph based techniques	contains file analysis techniques	In IoT networks
Svacina et al. 2020	On Vulnerability and Security Log analysis: A Systematic Literature Review on Recent Trends	2020	not mentioned	✓	✓	✗
Alhajri et al. 2019	Survey for Anomaly Detection of IoT Botnets Using Machine Learning Auto-Encoders	2019	not mentioned	✗	✗	✓
Deorankar and Thakare 2020	Survey on Anomaly Detection of (IoT)- Internet of Things Cyberattacks Using Machine Learning	2020	NSL-KDD & UNSW-NB15	✗	✓	✓
Haji and Ameen 2021	Attack and Anomaly Detection in IoT Networks using Machine Learning Techniques: A Review	2021	IoT-23 & BoT-IoT & NSL-KDD & DS2OS & CICIDS-2017 & ICS & UNSW-NB15 & KDDCUP99	✗	✗	✓
Zhou et al. 2021	A Review of Machine Learning and Deep Learning Techniques for Anomaly Detection in IoT Data	2021	KDD99 & NAB & UCSD & CUHK	✗	✓	✓
Hasan et al. 2019	Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches	2019	NSL-KDD dataset	✗	✗	✓
Ngo et al. 2020	A survey of IoT malware and detection methods based on static features	2020	not mentioned	✓	✗	✓
Erhan et al. 2021	Smart anomaly detection in sensor systems: A multi-perspective review	2021	Bot-IoT, ODDS, NAB	✓	✗	✓
Moustafa, Hu, et al. 2019	A holistic review of Network Anomaly Detection Systems: A comprehensive survey	2019	KDD99, NSL-KDD, Kyoto, DARPA, ADFA, UNSW-NB15, NGIDS-DS	✓	✗	✗
Martins et al. 2022	Host-based IDS: A review and open issues of an anomaly detection system in IoT	2022	NLS KDD, HIDS	✓	✗	✓

2020 proposed a study of different anomaly detection techniques to prevent harmful activities, improve the efficiency of attack detection, and detect the malicious behavior of data packets over a network, using machine learning based on the anomaly detection system. In addition, Haji and Ameen 2021 proposed a study to aid in the resolution of IoT security challenges by comparing different machine learning algorithms in terms of attack detection and anomaly detection in IoT networks, as well, possible ML-based IoT security technologies have been discussed. The findings of a study on anomaly detection techniques were presented by Fahim and Sillitti 2019, they concentrated on research in the areas of intelligent inhabitant environments, transportation systems, and other IoT-related examples. They discovered a number of research gaps in data collection, analysis of large imbalanced datasets, and limitations of statistical methods for analyzing large amounts of sensory data. Also, Ngo et al. 2020 conducted a comprehensive survey of the static-based method for detecting IoT malware, where they summarized, compared, and analyzed the existing methods. Furthermore Erhan et al. 2021 reviewed the latest methods that can be used to detect anomalies in a specific area of sensor systems, the review points to the most promising intelligent sensing methods and identifies a set of interesting open issues and challenges. Moreover, a study has talked about the issues encountered in the process of anomaly detection in IoT systems (Martins et al. 2022). Although the use of machine learning algorithms is well, the use of deep learning algorithms is likely to lead to more accurate results in some cases, especially when we have a big amount of data, some studies have discussed the use of deep learning algorithms to detect anomalies (Alhajri et al. 2019, Ma et al. 2021, Pang et al. 2021).

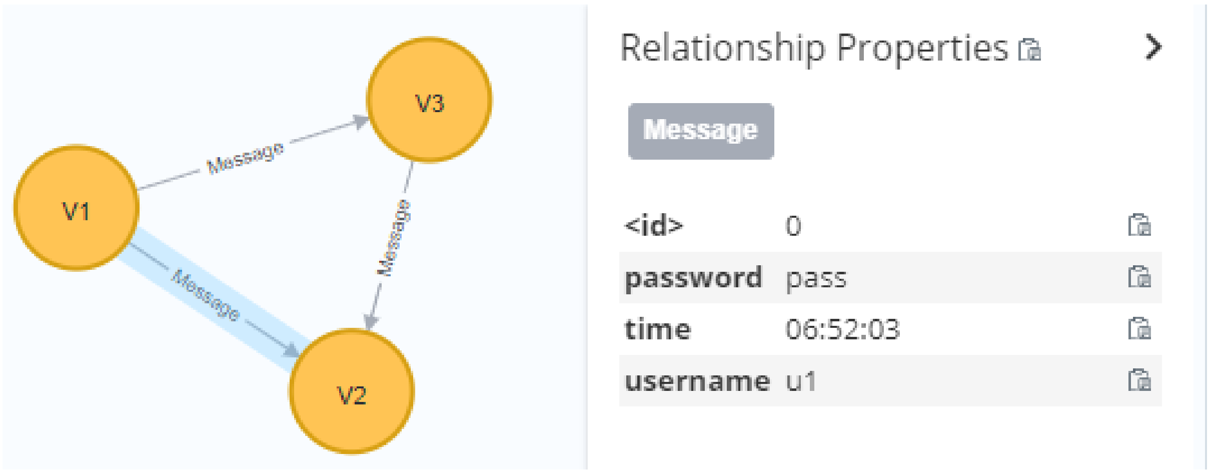
1.3 Graph-based approach

As mentioned before, a graph-based approach may be good in order to detect cyber attacks, because a graph is designed to express data that is highly connected and interdependent, and this is especially true in IoT networks, where there are numerous connections between devices, as well, as shown in fig 1.3, in an IoT network, devices may be thought of as nodes that can carry multiple attributes such as a device IP, name, etc.., and communication can be thought of as edges that can also carry multiple attributes such as connection duration, connection start time, etc... or it can simply carry a weight attribute only, this weight is increased every time a connection is created between the same two devices (Paudel, Muncy, et al. 2019).

Device A
connection
Device B

Figure 1.3: An example of graph representation for IoT devices

Another example is as shown in fig1.4, like Jia et al. 2018, given a set of target messages, we can enumerate all entities, such as smart IoT devices, servers, and smartphones, in the traffics saved in one or more files, and parse them as a set of nodes $V = \{v_1, v_2, ..., v_n\}$, if entity v_i sends a message to v_j and this message was captured and recorded at time t, we add a directed edge e^t_ij from v_i to v_j to the set of edges E, each e^t_ij belongs E is labeled by C^t_ij which is the set of all attributes and their values. Graph representation helps to store data while maintaining relationships between devices by using a graph database, the data can then be retrieved and analyzed to detect attacks and malicious actions using machine learning algorithms by detecting anomalies in the data.



<id>	0
password	pass
time	06:52:03
username	u1

Figure 1.4: An example of graph representation for IoT devices

As shown in Fig.1.5, an offline or online graph-based approach is possible. An online approach analyzes data and detects attacks in real-time, allowing for proactive decisions to protect the system before it suffers serious damage. This is accomplished by collecting data on a regular basis, modeling the data in a graph database, and retrieving the data for rapid analysis using a machine learning model that has already been trained. As well, to analyze data, detect passive attacks, and make reactive decisions, an offline approach is used.

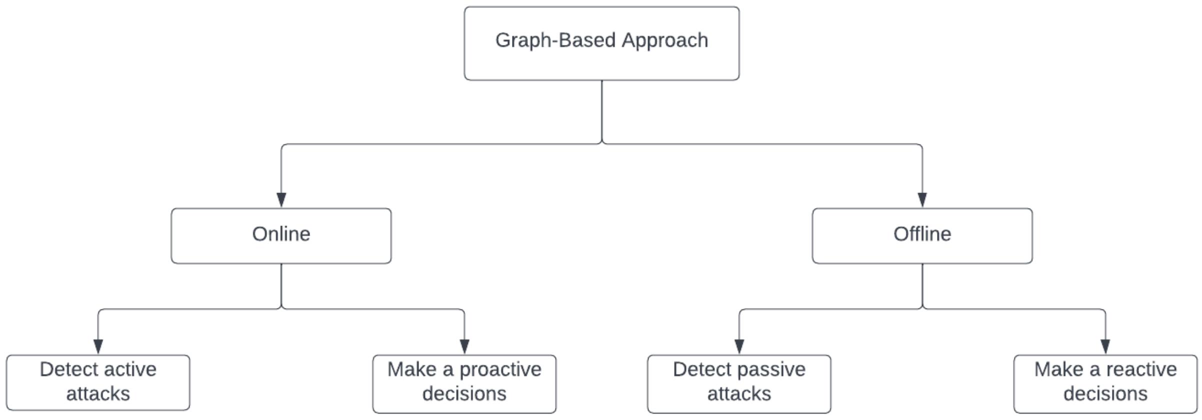


Figure 1.5: Types of the graph-based approaches for attacks detection

So, the graph-based approach has two main tasks: the first is to collect all data from IoT devices and model it in a graph database. The second task is to devise a method for extracting the key features from the stored graph and feeding them into a machine learning algorithm for anomaly detection purposes. In the remainder of this section, we will explain these two tasks in detail and talk about the different research papers out there that have discussed these two problems.

1.3.1 Graph from files

Nowadays, all systems and applications produce and receive a large amount of data stored in files, so we can benefit from those files in several areas, including cybersecurity by analyzing them in order to detect attacks, defects, and vulnerabilities. Anomaly behavior and signatures in data can be identified as a system attack or intrusion and mitigated by tracking key events in data (Svacina et al. 2020). Because the majority of the system generates a large number of files, human analysis of those files is extremely difficult. Furthermore, threats are frequently detected through a pattern of entries spread throughout the file, rather than individual entries. In order to achieve this, it is necessary to automate the analysis of data (Svacina et al. 2020). Structuring files in graph databases brings flexibility and extensibility to data representation and management. For this reason, several graph_based techniques have been made to automate the process of analyzing files in order to improve systems security where all data is analyzed on a continuous basis, for example, to create a profile that represents normal behavior when there isn't a threat, one of the techniques to do this task is the applying of kill chain model using a graph database, to identify various attack vectors and improve detection rate precision, where all of an adversary's and their targets' actions are described by a series of events. In their approach Schindler 2018 implemented the kill chain model as a directed graph with three main sections where the log events were sorted in chronological order representing the first section connected with the event sequences section which in turn is connected to the killing chain mapping section. The graph analysis successfully identifies the adversary after simulation using two datasets. In another paper, Jia et al. 2018 built a graph_based techniques with the target of vulnerability detection in IoT traffic. In another paper, the log file is modeled in a graph database but not for security purposes, it's for answering the question of whether it is a good idea to employ graph databases as combined storage for all systems logs (Hofer et al. 2020), where we can benefit from this paper by taking a look at the different modeling methods of log files. In addition, Paudel, Muncy, et al. 2019 proposed a lightweight graph-based outlier detection technique in the real-time graph stream of IoT traffic by taking advantage of graph characteristics, so after presenting smart home IoT traffic as a real-time graph stream, they processed graph data, and finally detect DOS attack in real-time.

As we can see, there are various techniques for modeling IoT communication in the graph database in order to analyze the data and detect attacks and malicious events. The whole technique and the modeling structure always leave an impact on the result, but in general, the graph database-based methods for analyzing data, especially in the anomaly detection target showed a good result, so the research portal in this field remains open.

1.3.2 Data analysis for anomaly detection

Chandola et al. 2009 defined anomaly detection as the problem of detecting patterns in data that do not follow expected patterns. Anomalies, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities, and contaminants are all terms used to describe nonconforming patterns in different application domains. In the context of anomaly detection, the terms "anomalies" and "outliers" are frequently used interchangeably. Credit card, insurance, and health-care fraud detection, cyber-security intrusion detection, fault detection in safety-critical systems, and military surveillance of enemy activities are just a few of the applications where anomaly detection is used. We can take advantage of the anomaly detection techniques in the context of IoT networks, an attack indicates abnormal behavior, indicating an anomaly. For example, we can detect a DOS attack by analyzing the data and discovering a large amount of communication from multiple devices to a server, which is an anomaly. After modeling the log data in the graph database, we need to find a technique to extract all the key features from the graphs to be input to the machine learning algorithm to detect anomalies. To put it another way, the main challenge is extracting informative features that can represent a graph structure and feed them into a machine learning model. Graph kernel, graph embeddings, and graph sketching are three popular graph representation techniques. Sketching is a graph representation technique in which a higher-dimensional object, such as a graph, is projected into a lower-dimensional feature vector. Sketching, in other words, summarizes large graphs while preserving their original properties. These methods are capable of handling a graph (Paudel and Eberle 2020). In a paper, Paudel, Muncy, et al. 2019 proposed a graph technique where they created a graph in which IoT devices are represented as nodes and the connection between them is represented as edges, then they extracted a vector that holds informative features of the graph, and finally, the extracted vector is input into the machine learning algorithm to detect whether or not there is an anomaly.

1.4 Graph-based anomaly detection approaches in IoT

In this section, we will talk about the different approaches that have contributed to protecting IoT networks by detecting malicious activities, using a graph-based technique, and using the anomaly detection-based method. Listing the principal of each approach, advantages, and disadvantages.

In a paper, Jia et al. 2018 built an approach to examining common IoT systems' potential vulnerabilities. They used a simple smart home example system consisting of a Google Home smart speaker, a smart light bulb, and a smartphone to demonstrate their work, which focuses on the detection of vulnerabilities in IoT communications. Their approach is composed of three main steps, firstly they constructed a directed graph using the collected IoT data, then they isolated correlated subgraphs, a correlated subgraph is a subgraph in which all entities and edges have a meaningful relationship, implying that all messages corresponding to the edges of the subgraph perform a set of coordinated actions, such as turning on/off the light. And lastly, they have identified vulnerable subgraphs by weighing correlated subgraphs, and based on the assigned weight they inferred if this subgraph is vulnerable or not. The dataset used in this approach is a packet capture file, they collected it in a continuous 15 minutes containing 58,714 messages in total, they got 4,102 messages after removing all the unnecessary ones like SYN, SYN-ACK, ACK, and FIN. After applying their approach, they found that six correlated subgraphs can be exploited to create six different attack scenarios. Finally, they created six attack cases based on the six identified vulnerable correlated subgraphs to test the effectiveness of their approach, three of the six attacks require the attacker to obtain some weak credentials from the victim, to begin with, while the other three do not. Although their approach does not directly detect the attack, they have detected vulnerable points in an IoT network, so we can say that they have detected a possibility of an attack occurring. One of the disadvantages of their approach is that cannot be used for real-time attack detection since it doesn't use either signature-based method or anomaly-based method, which prevents the detection of new and emerging attacks.

In the paper Paudel, Muncy, et al. 2019, they have built an approach to detect the DOS attack in real-time in an IoT network without the need for full packet access such as protocol, and packet size port numbers. In this paper, the smart home IoT dataset was created with 28 different IoT devices from a real-life smart home environment, including healthcare devices, cameras, light bulbs, switches and triggers, hubs, air quality sensors, and electronics. The graph presentation in this technique is very simple, each device in the IoT network is presented as a node in the graph and each connection between two devices is presented as an edge between these two nodes, if there is multiple communication between two nodes, the edge weight is increased between these two nodes. The principle of their approach is to build a new version of the graph every minute, each version representing the traffic collected at the last minute, then they extracted a vector from each version of the graph that holds its informative features, to be used as an input to a machine learning model, who has responsibilities to detect whether there is a DOS attack or not. To do that, they extracted many random paths from each graph, and based on the sum of the weights in the path, as well as on the frequency of that path (the number of times the path occurs in the graph) they extract a graph vector that holds the cost for the path in the graph, based on the sum of the weights and frequency. Finally, the extracted graph vector is fed into a machine learning algorithm to determine whether or not there is an anomaly. This approach has multiple advantages, for instance, their technique can detect DOS attacks in real-time without requiring much information about packets, as well as it's a lightweight technique because it does not require high computation power from devices. On the other hand, this approach has some drawbacks, because it only detects a DOS attack and turns a blind eye to other types of attacks. Also, their approach does nothing to stop or prevent the attack, it only detects attacks.

In another paper Mills et al. 2021, they devised an approach in which they built a dataset containing labeled attacks and then trained a machine learning model using their dataset in order to detect emerging cyber attacks in real-time. They used Honeypots to collect data, Honeypots are surveillance systems with components that imitate legitimate enterprise infrastructure to catch unsuspecting attackers using previously unknown exploits, attack tactics, and infiltration patterns. In order to label the collected data, they used Cyber Threat Intelligence Services, organisations can use Cyber Threat Intelligence Services to learn more about the threat actors attempting to compromise infrastructure. The vast majority of services are tailored to the needs of the client and provide a variety of intelligence. These include services that systematically scan and profile the entire internet address range, as well as services that maintain historical records of identified attackers in blacklists. Graph-based metrics and a variety of machine learning (ML) algorithms are used to classify malicious behavior. The labeled dataset of this paper is available for research purposes. Their approach has multiple advantages such as it detects attacks in real-time in a continuous manner using anomaly detection methods, and cyber threat intelligence which leads to detecting emerging cyber threats. But the use of cyber threat intelligence also has a drawback, because the labeling of their dataset is related to the outcome of the cyber threat intelligence services, so training and typical performance of the machine learning model is also associated with these services, and the true nature of how these services work is unknown to the general public for commercial reasons, so any fault in the outcome of those services, leads to a decrease in the accuracy of their technique.

In the paper Nguyen et al. 2020, they have built a technique to detect IoT botnet, in their approach, they aimed to demonstrate the advantages of static methods for IoT malware botnet and propose a new graph-based method to detect IoT botnets. Function–call graph generation, printable string information graph construction, preprocessing, and classification, are the four steps in their method. First, binary programs must be unpacked and disassembled. They then use the disassembled codes and caller–callee relationships to create function–call graphs and printable string information graphs. Using a graph embedding technique, the printable string information graphs are then processed and converted into numeric vector data. Finally, using a convolutional neural network, the samples are divided into two categories: botnet and non-botnet. Despite the fact that their method is effective and efficient for detecting IoT botnet malware files with a 98.7%

accuracy, it has a major drawback in that it uses a static approach which is different from the dynamic approach, it is not able to detect complex and polymorphic malware.

In the paper Zhu et al. 2022, they proposed an anomaly detection algorithm based on graphs. By constructing the covariance matrix and the statistical test quantity using a graph model of the WSN, so the anomaly detection of the sensor network can be realized. In addition, to optimize the topology of the sensor network graph, this paper selects a subset of edges with high correlation to construct a new graph. Although the goal of their method is to ensure network data accuracy and reliability by detecting outliers that show abnormal collected data, such as a sensor node affected by a battery power shortage, they have developed a graph-based anomaly detection technique that could be useful in cyber security targets.

Chapter 2

Our approach to detect attacks in IoT networks

2.1 Motivation

IoT can boost productivity and efficiency through remote administration that is intelligent, but it also raises the risk of cyberattacks. IoT application security issues and potential attacks have recently gained attention in the cyber security community. Mission-critical operations, such as those involving industrial control and infrastructure systems, are involved in several IoT-based applications, and they call for a high level of security. According to reports, many power substations in Ukraine were infiltrated in the most recent attack on IoT applications, causing a power outage that affects about 225,000 customers (Falco et al. 2018). An attacker successfully gained access to this system through an IT network, compromised a Supervisory Control and Data Acquisition system, which controls and monitors IoT devices on the smart grid, and disconnected the power (Falco et al. 2018). The Mirai botnet, which in late 2016 was mostly made up of compromised smart cameras, is another example of IoT attacks. It caused widespread Internet outages when it flooded several well-known organizations with intense distributed denial-of-service (DDoS) attacks (Antonakakis et al. 2017).

Many connected IoT devices and sensors are not as secure as machines in traditional networks. By thinking about it, we can ask some questions, such as whether our smart device has an antivirus installed? Or is our smart security camera protected against malware? This is why many hackers are turning their sights to these susceptible IoT devices, when only one IoT device in the network is hacked, it can disrupt the entire network and expose confidential data, for that, IoT security is becoming a top priority for every organization or individual considering launching an IoT initiative, in order to prevent cybersecurity threats and protect their data. The Internet of Things is different from standard Internet networks, it contains physical devices that are connected to each other, since most devices are battery-powered, there are limitations in memory and computing power of these devices, in addition, the Internet of Things network has a large amount of communication between devices, so it generates a large amount of data describing these connections with their attributes, therefore, an approach for attack detection in IoT networks requires some techniques to handle big data in order to be able to find anomalies, while taking into account the characteristics of the IoT network and the limitations of the connected things, for example by not relying on them to do some extra work. While there are several methods to improve security in networks, IoT provides a unique case. Not only does the Internet of Things have a lot of attack surfaces, but it also has its own set of security problems that must be taken into account in any security approach. For example, in IoT, there is a large assault surface with intricate interconnection, because of their ability to connect a variety of software and hardware solutions. The more ways these devices and software may communicate with one another, the more opportunities cybercriminals have to expose these relationships. This poses a unique problem for IoT security because we want the IoT network to be as flexible and accessible as possible for our users, then we want more devices and more connectivity choices. But, these possibilities will provide fraudsters with a new opportunity to attack the IoT network. The second main difficulty in IoT security, as previously stated, is that many IoT devices and sensors lack the necessary resources to provide comprehensive protection, many small IoT devices, for example, lack the processing power and storage space required to run effectively antivirus and firewall software. On the other hand, many IoT devices are designed to be as small as possible, and security is often a secondary consideration. In conclusion, an IoT security approach must consider all possible attack surfaces and entry points. Even a single device with a slight defect can be hazardous. Which makes IoT security approaches more difficult and expensive to implement than traditional security approaches. As a result, Iot cannot use security applications or cryptographic techniques that demand advanced processing skills, and it is essential to develop cyber-security tools like intrusion detection systems (IDS) that are specifically created to fit the needs of IoT applications in order to detect attacks against IoT services.

2.2 Background

2.2.1 What is Neo4j?

The most popular open source Graph Database in the world, Neo4j, was created utilizing Java technology. It is very scalable and has no schema (NoSQL). A graph is a visual representation of a collection of things where some object pairs are linked together. Relationships (edges) and nodes (vertices) make up two components. A database called a "graph database" is used to model data as a graph. Here, the entities are represented by the graph's nodes, while their connections between them are shown by their relationships.

2.2.2 Why Graph Database?

The majority of data today comes in the form of relationships between various objects, and most of the time, these relationships are more useful than individual pieces of data. Relational databases can be used to store highly structured data since they contain several records of the same type of data, but they do not store the relationships between the data. Graph databases hold connections and relationships as first-class entities, in contrast to conventional databases. In addition, when compared to other databases, Neo4j allows us to more quickly represent and readily access (traverse/navigate) related data, also Neo4j provides a declarative query language that is very easy to learn. In addition, Neo4j offers graph algorithms, which are among the most effective methods for evaluating related data due to the fact that their mathematical operations are created expressly to work on relationships. They outline the procedures to take in order to analyze a graph and determine its fundamental characteristics or precise values. Where we can call an algorithm using the cypher query language by a simple command and straightforward syntax, Neo4j Graph Data Science is a library that offers quickly implemented parallel versions of basic graph algorithms.

2.2.3 Graph Embedding

As written by Tong 2020, a strategy called ”graph embedding” is used to convert nodes, edges, and their attributes into a lower dimension vector while maintaining the most amount of the graph's information and structure. Graphs can differ in scale, specificity, and subject, which makes them challenging to interpret. We have seen that graph embedding has gained significance in a number of machine learning techniques during the past few years. We can carry out a number of operations including clustering, classification, etc. Using the nodes, edges, and other graph embedding components. In contrast to an IoT network, which might be represented by a huge, dense, and dynamic graph, a small social network can be shown as a small, sparse, and static graph. This makes it challenging to identify a perfect embedding technique. The embedding techniques itself are not a kind of neural network model if we think of embedding as a change to a lower dimension. Instead, they are a type of method used in graph pre-processing to convert a graph into a format that can be processed computationally. The following algorithms all perform differently on various datasets, but they are the ones that Deep Learning practitioners utilize the most: DeepWalk (Perozzi et al. 2014), node2vec (Grover and Leskovec 2016), graph2vec (Narayanan et al. 2017), SDNE (D. Wang et al. 2016), LINE (Tang et al. 2015), GraphSAGE (Hamilton et al. 2017). In our approach, we used GraphSAGE, so in the next subsection, we will give an overview of how this algorithm works.

2.2.4 GraphSAGE

As written by Özçelik 2019, transductive algorithms, which include existing methods like DeepWalk, require access to the entire graph in order to learn a node's embedding. Therefore, it must be restarted to produce an embedding for the new node whenever one is added to the old ones. For example, in our case, where we considered each IoT device as a node in the graph, if the owner decides to add one more IoT device to the existing network, we have to restart and retrain graphSAGE for our intrusion detection approach to work properly. GraphSAGE, on the other hand, is a representation learning method appropriate for dynamic graphs. Without the need for retraining, GraphSAGE can forecast the embedding of a new node. To do this, GraphSAGE learns aggregator functions that, given a new node's properties and neighborhood, can initiate the embedding of that node. We refer to this as inductive learning. The three major components of GraphSAGE are context construction, information aggregation, and loss function. As a result, GraphSAGE is a learning technique for inductive representations that is particularly effective for graphs that expand over time. GraphSAGE is substantially faster than transductive methods at creating embeddings for new nodes. GraphSAGE does not sacrifice performance for speed. It outperformed the previous solutions in tests on three separate datasets involving node classification, node clustering, and across-graph generalization. For more information about the graphSAGE, the reader can read the following article: Hamilton et al. 2017.

2.2.5 Overview

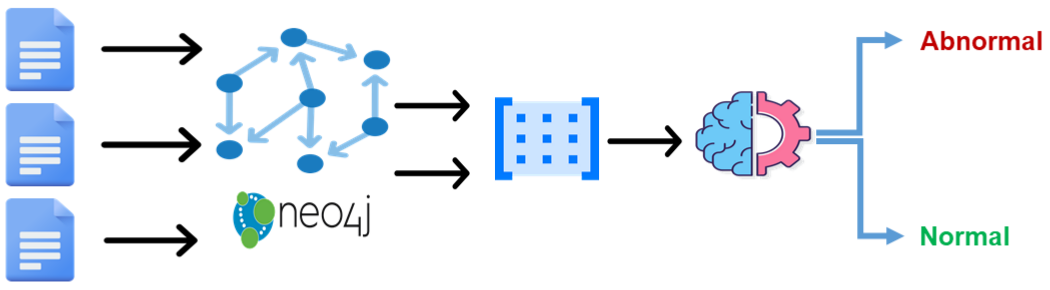


Figure 2.1: The main steps of our approach

In order to detect attacks in IoT networks in real-time, as shown in Fig2.1 we proposed an approach that generally consists of three main steps that lead us to achieve our goal. The first step is to model the data generated by the network by a graph database such as neo4j. To achieve real-time attack detection, we need to collect data every x seconds, it is recommended that x be small to detect attacks early, which helps decision makers to make their own decision early and make a proactive decision, for example, we can collect data every 30 seconds and then modeling the collected data in a graph database. The second step is to extract a multidimensional vector that preserves the topology of the graph while preserving the properties and showing the main features of the graph, in other words, the extracted vector must realize a numerical description of the data collected in the last x seconds that were used to create the graph. In the third and final step, each extracted vector is an input to a machine learning or deep learning algorithm, which has the responsibility to detect whether the last x seconds contain an attack or not.

2.3 Used dataset:TON_IoT

The Ton_IoT dataset is a recent, representative dataset that can be used to precisely design and assess IoT defense solutions. They suggested creating a brand-new data-driven IoT-based dataset based on data gathered from each IoT device's representative scale-down testbed. For use by the research community, the proposed datasets are made available to the public. The data was collected using the testbed's seven IoT devices, which include weather and Modbus sensors (Alsaedi et al. 2020).

2.3.1 Why Ton IoT?

A survey of cybersecurity studies using data mining and machine learning techniques for intrusion detection systems was done by Buczak and Guven 2015. They confirmed that a crucial gap in the literature that needs to be filled in order to create a promising anomaly-based intrusion detection method is the absence of a labelled dataset. Different network datasets, such as KDDCUP99, NSLKDD (Tavallaee et al. 2009), UNSW NB15

(Moustafa and Slay 2015), and ISCX (Sharafaldin et al. 2018), were created for evaluating IDSs. However, because these datasets do not contain either sensor reading data or IoT network traffic, they do not contain any particular characteristics of IoT applications. The LWSNDR (Suthaharan et al. 2010) dataset excludes any attack scenarios and only includes homogeneous data gathered from single-hop and multi-hop Wireless Sensor Networks (WSNs).

2.3.2 Overview of the dataset

As mentioned by Alsaedi et al. 2020, The ToN-IoT datasets contain information taken from the Telemetry data of IoT services, as well as the Operating Systems logs and Network traffic of IoT network. These datasets were created at the UNSW Canberra Cyber Range and IoT Labs from a realistic representation of a medium-scale network. Furthermore, a label feature was added to the suggested datasets (indicating whether an observation is normal or attack) There were nine (9) different forms of cyberattacks launched against different IoT sensors throughout the IoT network, including scanning, DoS, DDoS, ransomware, backdoor, data injection, Cross-site Scripting (XSS), password cracking assault, and Man-in-the-Middle (MITM). Seven (7) IoT sensors, two smartphones, and a smart TV were utilized to collect data. The obtained data were stored in log and CSV files. The "Train Test datasets" folder contains the bulk of the IoT datasets. Samples of the datasets used as train-test datasets in a CSV format are contained in the "Train Test datasets" folder and are used to assess the efficacy and accuracy of new cybersecurity apps and machine learning techniques. All seven (7) IoT devices—a Fridge, a GPS tracker, a motion light, a garage door, a thermostat, and a weather monitoring device—are included in the Train-Test IoT datasets. To create the dataset, various IoT scenarios were simulated on the testbed. The popular IoT applications of smart homes, smart cities, and smart manufacturing may all use these scenarios. For instance, most smart homes have sensors for the Smart Fridge, garage doors, and motion lighting. In smart cities, GPS sensors can be discovered. The majority of industrial and smart manufacturing applications use the Modbus, thermostat, and weather monitoring. Now we will explain the details and role of each IoT device:

- **Smart fridge** It measures the temperature and lowers it below a threshold when necessary.
- A **garage door** that is remotely operated can be opened or closed in response to an input.
- Global Positioning System, **GPS**, remotely keeps track of an object's latitude and longitude coordinates.
- **Smart motion light** sensor that activates or deactivates lights depending on a pseudo-randomly generated signal.
- The **Modbus service** mimics the functionality of the Modbus devices used in many industrial applications, which transfer register types like input, discrete, holding, and coil across serial lines utilizing master-slave communication.
- A **smart thermostat** controls a heating or cooling system (such as the central heating, air conditioning, or water heaters system) to control the temperature of a physical system.
- Data about temperature, humidity, and air pressure are produced by **weather monitoring systems**.

2.3.3 Attack scenarios

As mentioned by Alsaedi et al. 2020, in order to categorize the data as either normal or an attack, several IoT sensors were subjected to a variety of cyber-security incidents, including DDoS and ransomware, XSS - Cross-Site Scripting, backdoor, and injection, to tag each vector in the dataset, they used the timestamp field of each well-known attack that happened. Below are further details on the hacking scenarios and various tools used to launch these attacks.

- **Scanning** is regarded as the first phase of an assault, during which an attacker gathers data about a target system, such as opening ports and available services about a victim device or sensor, before launching the real attack. The attacker often scans ports using scanning tools like Nmap (Lyon 2008) or Nessus (Wendlandt 2010). They conducted scanning attacks using the offensive Kali systems' Nmap and Nessus tools.

- **Denial of Service (DoS)** is a well-known flooding attack in which an attacker often makes a series of fraudulent attempts to prevent a genuine user from accessing services. The IoT dataset includes attacks such as Distributed Denial of Service (DDoS) and Denial of Service (DoS). Typically, a huge number of infected machines known as bots or botnets start DDoS attacks. This attack can be carried out by overloading the resources of the target IoT devices with a large number of connections (e.g., CPU and memory). As mentioned before, IoT devices have a limited computation power and storage capacity which makes them easily vulnerable to DoS attacks. A Python script called "dos.py" was created to carry out DoS attacks using the Scapy package against such vulnerable IoT devices in their scenarios, and to launch DDoS attacks utilizing the UFONet toolkit, various Python scripts (such as ddos.py, ddos2.py...) were built.
- **Ransomware** is a sophisticated form of malware that seeks to sell the decryption key that would allow a user to regain access to the system after denying them access by encrypting the system or services in question. The same is true of IoT ransomware, where IoT devices cannot be accessed. Due to the fact that many IoT devices and applications perform mission-critical functions, blocking access to or locking down these applications could have disastrous effects such as financial losses. They carried out the ransomware attack after taking advantage of flaws in the target devices using the Metasploitable3 framework.
- **Backdoor** is a passive attack that enables an attacker to obtain unauthorized remote access to the afflicted IoT devices through backdoor malware. The attacker makes infected IoT devices a part of botnets to attempt DDoS attacks and uses the backdoor to control those devices. To conduct backdoor attacks, they used the

Metasploitable 3 framework.

- **Injection Attack** frequently attempts to run malicious code or introduce malicious data into IoT applications. Additionally, the injection attack has the ability to alter data and control commands in the IoT system, which would interfere with normal operation. Two bash scripts called injection-1.sh and injection-2.sh were created to perform injection attacks against Security Shepherd VMs, IoT service websites, and vulnerable public PHP web applications.

- **Cross-Site Scripting (XSS)** In IoT applications, there are frequent attempts to execute malicious commands on a Web server. The XSS allows an attacker to remotely insert any Web scripts, such as malicious HTTP or JavaScript instructions. The data and authentication processes between Internet of Things devices and a remote Web server may be compromised by this attack. In order to conduct XSS attacks against the webpages of IoT services, they created two bash scripts (such as xss1-sh and xss2.sh) utilizing the Cross-Site Scripter (XSSer) toolkit (Cui et al. 2020).

- **Password Cracking Attack** When a hacker tries to guess an IoT device's password using brute force attacks or other password cracking techniques. As a result, the attacker may be able to avoid authentication processes and compromise IoT devices. They created two bash scripts, such as password-1.sh and password-2.sh, which employ the Hydra toolkit (Papadogiannaki et al. n.d.) for brute force assaults and the CeWL toolkit (Moustafa, Ahmed, et al. 2020), respectively, for dictionary attacks. These scripts were created to conduct password attack scenarios simultaneously against testbed susceptible IoT devices.

- **Man-In-The-Middle (MITM) attack** is a well-known network attack that can modify data by intercepting the communication route between two machines. Common MITM attacks include ICMP redirection, port stealing, and ARP cache poisoning. They carried out ICMP redirection, port-stealing, and ARP Cache poisoning attacks using the Ettercap tool (Choi et al. 2021).

2.4 Implementation

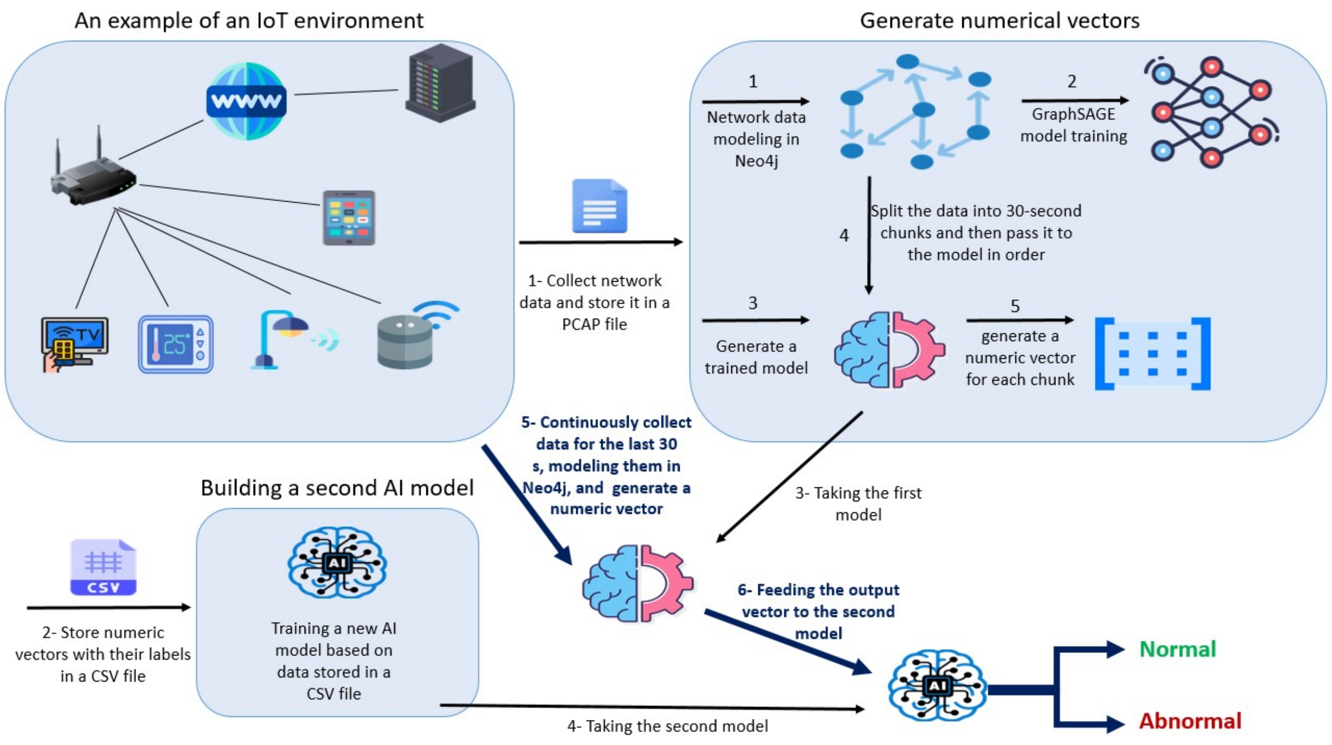

An example of an IoT environment
Generate numerical vectors
1 Network data modeling in Neo4j
2 GraphSAGE model training
4 Split the data into 30-second chunks and then pass it to the model in order
3 Generate a trained model
5 generate a numeric vector for each chunk
1- Collect network data and store it in a PCAP file
5- Continuously collect data for the last 30 s, modeling them in Neo4j, and generate a numeric vector
3- Taking the first model
Building a second AI model
2- Store numeric vectors with their labels in a CSV file
Training a new AI model based on data stored in a CSV file
6- Feeding the output vector to the second model
4- Taking the second model
Normal
Abnormal

Figure 2.2: Our approach

As mentioned before in section 2.2.5 we have three main steps in our approach, in this section, we will detail our implementation of these steps, fig 2.2 illustrates our approach in detail. The first step is modeling the IoT network data in the graph database, we chose Neo4j as a graph database. From the Ton_IoT dataset, we used the file named Train_Test_Network.csv (for more information about this file, the reader can refer to section 2.3.2). This file contains 45 features extracted from packet captured files, some with numeric and continuous values, some with discrete or categorical values, or even they contain no values (noise). Since GraphSAGE only works with numeric and continuous node and edge properties, we needed to extract only features containing such kinds of values, we also needed to remove all noise. So, after feature extraction, we have left 12 features, in the following, we will explain the mean and the role of each of these features:

- **Time stamps** each packet that is captured has a time stamp on it as it arrives. These time stamps will be recorded in the capture file and made available for study at a later time. The time stamps are obtained during capture from the libpcap

(Npcap) library, which in turn obtains them from the kernel of the operating system.

- **Source IP address** This represents the IP address of the IoT device that launched the connection, for example, a connection between two devices.
- **Destination IP address** Which represents the IP address of the destination IoT device in a connection.
- **Duration** This represents a connection's duration.
- **Source bytes** represents the count of bytes sent from the source to the destination in a transaction.
- **Destination bytes** represents the count of bytes sent from the destination to the source in a transaction.
- **Missed bytes** which represents the number of bytes lost during the transaction.
- **Source packets count** represents the packet count sent during a transaction from the source to the destination.
- **Destination packets count** which represents the packet count sent during a transaction from the destination to the source.
- **HTTP request body length** it serves as an entity header and displays the entity-size body's in decimal. In essence, it is the quantity of data in the request's body, measured in bytes.
- **HTTP response body length** like the previous feature, but it represents the quantity of data in the response's body.
- **Label** which represents the type of the transaction, either normal or an attack, 0 means that this transaction is normal, and 1 means that it's an attack.

After getting our main features, we need to model this feature in the Neo4j database, our modulation is relatively simple, in which we have created only one type of node and only one type of relations, the node type represents an entity in the network and holds one property from the dataset which is the IP, and we added another property to the node type which is the node degree. On the other hand, each transaction between two entities in the network is represented by a relation (edge) between two nodes in the database, and each relation contains as properties all the rest of the features presented in the dataset after features extraction. We have implemented a python script that proceeds through each line of the csv file, creates new nodes for source IP and destination IP if one of these nodes does not exist before, and finally creates the relationship between these two nodes, we noticed that this process is very slow, so we decided to use the built-in 'load csv' command in Neo4j which is faster. After that we have our data modeled in the Neo4j graph database, we need to start training our models to use them to detect attacks in real-time. So, The second step to achieving our goal is to train the GraphSAGE model using data from the graph database. The good news is that GraphSAGE is implemented in Neo4j's built-in graph data science library. To train the GraphSAGE model, we need firstly to project our graph, a user-defined name can be used to save a projected graph in the catalog (main memory). Any algorithm in the library can refer to the graph by that name. This eliminates the need to project the graph before each algorithm run, allowing numerous algorithms to use the same graph. After storing our graph in the main memory, we proceed to train the GraphSAGE model by calling the 'train' function implemented in the graph data science library in Neo4j using the cypher language, once the training is complete, the GraphSAGE model is ready to take a new subgraph and generate and return the numeric vector that preserves the topology of this subgraph (preserves the properties of the graph). In our dataset, we have a feature which is called timeStamp which represents the time this transaction was captured, in order to train our machine and deep learning models to detect attacks in real-time we need to generate a significant amount of numerical vectors using our pre-trained GraphSAGE model, and these vectors will be a newly labeled dataset to train the new AI models. So we split our data sets into 30-second fragments, to put it another way, first, we took the network transactions that are captured in the first 30 seconds by querying them using the cypher query language, second, we stored the graph that represents these network transactions in the Neo4j model (main memory), and the final step was to generate the numerical multidimensional vector for these first 30 seconds by applying our pre-trained model GraphSAGE on the projected graph for the last 30 seconds, and so on for the next 30 seconds and so on. Until we have finished processing the entire dataset. GraphSAGE returns a numeric vector for each node in the graph, so the entire multidimensional vector of the graph is returned as a matrix, in order to transform this matrix into a simple numeric vector (1-row vector), we have implemented a small algorithm that sums each column of the matrix and store the result in a new one-dimensional vector, so that the new vector contains the same number of columns of the matrix, but it contains only one row. Afterward, We have a new dataset that contains in each row a numeric vector that represents the data captured every 30 seconds, along with a label that describes whether there is an attack within those 30 seconds or not. So, the last step of our whole approach is to train some machine and deep learning models using the new dataset and then detect cyberattacks in near real-time using our two pre-trained models (GraphSAGE model and AI models), by generating numeric vectors using the GraphSAGE model every new 30 seconds, then decide whether that vector contains an attack or not using our AI models.

2.5 Results and evaluation

When we got our new dataset, which contains the set of numeric vectors that describe the topology of graphs created from data captured every 30-second period, we noticed that this dataset contains an unbalanced classification, where there are more vectors that describe normal traffic than vectors that describe attack traffic. This problem arose due to the execution of DOS and DDOS attacks in a short period of time, resulting in a small number of numerical vectors describing this type of attack. The unbalanced dataset problem negatively affects some of the metrics we used in our assessment like precision and recall and therefore affects the F1 score. In order to make the samples more balanced, several procedures must be applied to alter the data. The conventional approaches consist of random undersampling and random oversampling. Undersampling is a technique used to make a sample appear more balanced by systematically reducing the number of majority individuals sampled. In contrast, the minority samples in the oversampling approach are given a higher weight to counteract the unequal representation of the data acquired. SMOTE (Synthetic Minority Oversampling Technique) is ultimately selected in our experiment since it offers the best performance. This technique is an enhanced oversampling-based technique. When using the algorithm SMOTE, the minority class is oversampled while adding synthetic examples and the line segments that connect the minority class's closest neighbors. SMOTE is used to generate artificial examples, which results in the classifier having a new region that is thought to be more expansive and less particular. Instead of being obscured by the surrounding majority class samples, a more generic region is now given as a representation of the minority class samples. Using the SMOTE technique enables us to more evenly alter the data sample. By using this approach, we are able to balance the data and provide a much more logical outcome for our model. Consequently, the SMOTE algorithm is used in light of these factors. After using the SMOTE algorithm, we got 1000 rows of numeric vectors, each row containing a numeric vector of 5 values and a label value indicating whether there is an attack or not. We used 600 rows to train our AI models and 400 to test them, and to evaluate them, we used 4 metrics: accuracy, precision, recall, and F1 score.

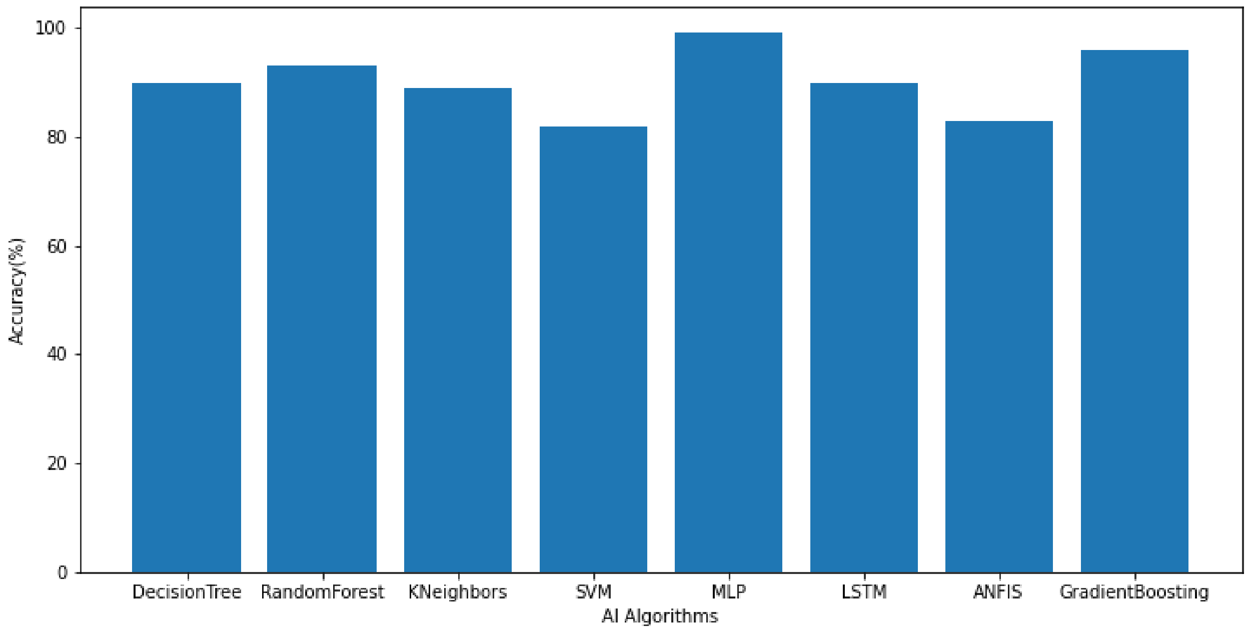


Figure 2.3: The measured accuracy of the AI algorithms used

One of the main metrics for evaluating our approach is accuracy, which is the proportion of all data points that were correctly predicted. Formally, it is the sum of the true positives and true negatives divided by the sum of the true positives and true negatives, as well as the number of false positives and false negatives. As shown in fig 2.3, among the many machine learning and deep learning algorithms used, MLP has occupied the top class with 99% accuracy, MLP is a deep learning technique that contains multiple layers of neurons. MLP is widely used for solving problems that require supervised learning. Although accuracy is the main metric for evaluating any artificial intelligence model, but it is not enough, other metrics are very important for evaluating the details of a model's performance before we choose the best model to use in our entire approach. We used three other metrics, the first is precision which is one indicator of a machine learning model's performance, and the quality of a positive prediction made by the model. Precision refers to the number of true positives divided by the total number of positive predictions (i.e., the number of true positives plus the number of false positives). The second metric is the recall which is how many of the true positives were recalled (found), i.e. how many of the correct hits were also found. The previous two metrics constitute a third metric which is the F1_score which is one of the most important evaluation metrics in machine learning. It elegantly sums up the predictive performance of a model by combining two otherwise competing metrics (precision and recall).

As shown in fig 2.4, the MLP deep learning model wins first place again, with F1_Score equal to 98%, so we will definitely choose this model to use in our approach because it gave the best results. We can see that other algorithms have good results like gradient boosting, random forest, and decision tree, with an accuracy of up to 90%. We can notice that all the models have good results in precision, recall, and F1_score, which is obtained using the SMOTE algorithm which helps us to obtain a more balanced data set.

As shown in table 2.1, there are several existing approaches on the subject of attack detection in IoT applications, we have taken 3 of these existing approaches to compare our approach with them, on one hand, there are techniques that use graph algorithms like node embedding algorithms, like the one built by Paudel, Muncy, et al. 2019 (GODIT),

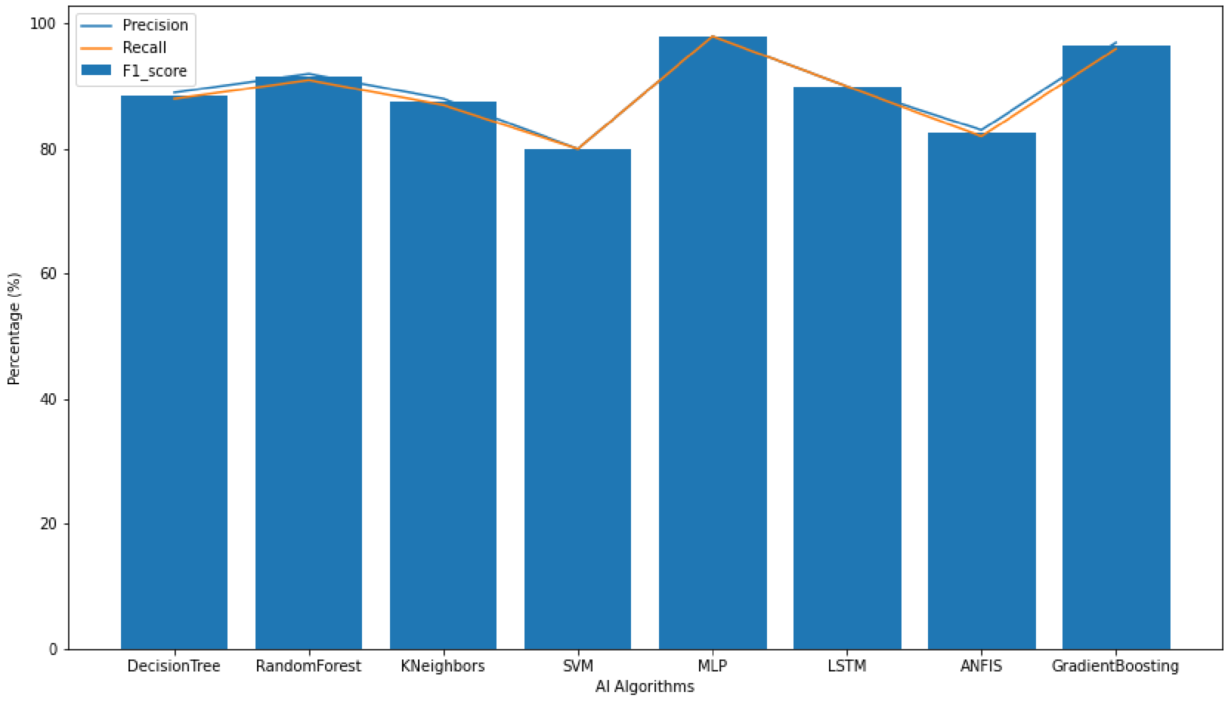


Figure 2.4: The measured precision, recall, and F1_score of the used AI algorithms

they used node2vec graph algorithm in order to detect attacks in smart home IoT applications, the second approach is built by Lo et al. 2022 (E-GraphSAGE), they built a new algorithm named E-GraphSAGE in order to detect intrusions in IoT applications. On the other hand, there are some techniques that do not use graphs, like the one built by Smys et al. 2020 (HCNN), where this approach also tries to detect IoT intrusions. Finally, there is our approach which uses the GraphSAGE algorithm to detect attacks in the IoT application, and we have noticed that our approach outperforms the existing ones.

As shown in table 2.2, where we compared the accuracy of our approach with the accuracy of two other approaches mentioned earlier, E-GraphSAGE and HCNN, we can see that our approach won first place in the accuracy metric with 99% accuracy. On the other hand, the accuracy of E-GraphSAGE is 97%, and for HCNN the accuracy is 98%. We did not compare the accuracy metric with GODIT approach, because in GODIT they did not use accuracy as an evaluation metric. However, when we compared our approach with E-GraphSAGE and GODIT with the precision metric, our approach lost the first class taken by E-GraphSAGE with 100% precision, where our approach was equivalent

Table 2.1: Existing techniques

Ref	Title	years	datasets	Contains graph based techniques	Graph algorithm	Use of Neo4j
Paudel, Muncy, et al. 2019	Detecting DoS Attack in Smart Home IoT Devices Using a Graph-Based Approach	2019	Their own dataset	✓	node2vec	✗
Smys et al. 2020	Hybrid Intrusion Detection System for Internet of Things (IoT)	2020	UNSW NB15	✗		✗
Lo et al. 2022	E-GraphSAGE: A Graph Neural Network based Intrusion Detection System for IoT	2022	TON_IoT	✓	E-GraphSAGE	✗
Our approach		2022	TON_IoT	✓	GraphSAGE	✓

to the GODIT approach in this evaluation metric with 98% accuracy as shown in table 2.2. The last evaluation metric in this comparison is the recall, where our approach won again with a recall of 98%, in contrast, GODIT has a 92% and E-GraphSAGE has 97%.

Table 2.2: Comparison of our approach with existing approaches

Approaches	Accuracy	Precision	Recall
GODIT	not mentioned	98%	92%
HCNN	98%	not mentioned	not mentioned
E-GraphSAGE	97%	**100%**	97%
Our approach	**99%**	98%	**98%**

Chapter 3

Conclusion

As a result of the internship, we made two contributions, the first contribution was state-of-the-art, where we introduced the domain of detecting cyberattacks in IoT networks, cited the latest surveys that talk about this field, then focused on methods that use graph-based techniques, present the results, summarize and cite the advantages and disadvantages of each, and finally highlight the challenges of constructing an intrusion detection approach in IoT. The second contribution was our approach to detecting cyberattacks in real-time, where we applied the GraphSAGE algorithm using Neo4j, then trained our AI model and achieved 99% accuracy in detecting intrusions in IoT applications. In our future work, we want to study the possibility of detecting attacks and specifying what type of attacks have been detected (scanning, DOS...), since our approach now detects the attack in a binary way (there has been an attack or not).

References

Alhajri, Reem, Zagrouba, Rachid, and Al-Haidari, Fahd (2019). "Survey for anomaly detection of IoT botnets using machine learning auto-encoders". In: *Int. J. Appl. Eng. Res* 14.10, pp. 2417–2421.

Alsaedi, Abdullah, Moustafa, Nour, Tari, Zahir, Mahmood, Abdun, and Anwar, Adnan (2020). "TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems". In: *IEEE Access* 8, pp. 165130–165150.

Antonakakis, Manos, April, Tim, Bailey, Michael, Bernhard, Matt, Bursztein, Elie, Cochran, Jaime, Durumeric, Zakir, Halderman, J Alex, Invernizzi, Luca, Kallitsis, Michalis, et al. (2017). "Understanding the mirai botnet". In: *26th USENIX security symposium (USENIX Security 17)*, pp. 1093–1110.

Buczak, Anna L and Guven, Erhan (2015). "A survey of data mining and machine learning methods for cyber security intrusion detection". In: *IEEE Communications surveys & tutorials* 18.2, pp. 1153–1176.

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

Choi, Jinchun, Narayanasamy, Deneesh, Ahn, Bohyun, Ahmad, Seerin, Zeng, Jianwu, and Kim, Taesic (2021). "A real-time hardware-in-the-loop (hil) cybersecurity testbed for power electronics devices and systems in cyber-physical environments". In: *2021*

IEEE 12th International Symposium on Power Electronics for Distributed Generation Systems (PEDG). IEEE, pp. 1–5.

Cui, Yanpeng, Cui, Junjie, and Hu, Jianwei (2020). “A survey on xss attack detection and prevention in web applications”. In: *Proceedings of the 2020 12th International Conference on Machine Learning and Computing*, pp. 443–449.

Deorankar, Anil V and Thakare, Shiwani S (2020). “Survey on anomaly detection of (iot)-internet of things cyberattacks using machine learning”. In: *2020 fourth international conference on computing methodologies and communication (ICCMC)*. IEEE, pp. 115–117.

Erhan, Laura, Ndubuaku, M, Di Mauro, Mario, Song, Wei, Chen, Min, Fortino, Giancarlo, Bagdasar, Ovidiu, and Liotta, Antonio (2021). “Smart anomaly detection in sensor systems: A multi-perspective review”. In: *Information Fusion* 67, pp. 64–79.

Fahim, Muhammad and Sillitti, Alberto (2019). “Anomaly detection, analysis and prediction techniques in iot environment: A systematic literature review”. In: *IEEE Access* 7, pp. 81664–81681.

Falco, Gregory, Caldera, Carlos, and Shrobe, Howard (2018). “IIoT cybersecurity risk modeling for SCADA systems”. In: *IEEE Internet of Things Journal* 5.6, pp. 4486–4495.

Grover, Aditya and Leskovec, Jure (2016). “node2vec: Scalable feature learning for networks”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864.

Haji, Saad Hikmat and Ameen, Siddeeq Y (2021). “Attack and anomaly detection in iot networks using machine learning techniques: A review”. In: *Asian journal of research in computer science* 9.2, pp. 30–46.

Hamilton, Will, Ying, Zhitao, and Leskovec, Jure (2017). “Inductive representation learning on large graphs”. In: *Advances in neural information processing systems* 30.

Hasan, Mahmudul, Islam, Md Milon, Zarif, Md Ishrak Islam, and Hashem, MMA (2019). “Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches”. In: *Internet of Things* 7, p. 100059.

Hofer, Daniel, Jäger, Markus, Mohamed, Aya, and Küng, Josef (2020). “On applying graph database time models for security log analysis”. In: *International Conference on Future Data and Security Engineering*. Springer, pp. 87–107.

Inayat, Zakira, Gani, Abdullah, Anuar, Nor Badrul, Khan, Muhammad Khurram, and Anwar, Shahid (2016). “Intrusion response systems: Foundations, design, and challenges”. In: *Journal of Network and Computer Applications* 62, pp. 53–74.

Jia, Yizhen, Xiao, Yinhao, Yu, Jiguo, Cheng, Xiuzhen, Liang, Zhenkai, and Wan, Zhiguo (2018). “A novel graph-based mechanism for identifying traffic vulnerabilities in smart home IoT”. In: *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, pp. 1493–1501.

Lo, Wai Weng, Layeghy, Siamak, Sarhan, Mohanad, Gallagher, Marcus, and Portmann, Marius (2022). “E-GraphSAGE: A Graph Neural Network based Intrusion Detection System for IoT”. In: *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, pp. 1–9.

Lyon, Gordon Fyodor (2008). *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure. Com LLC (US).

Ma, Xiaoxiao, Wu, Jia, Xue, Shan, Yang, Jian, Zhou, Chuan, Sheng, Quan Z, Xiong, Hui, and Akoglu, Leman (2021). “A comprehensive survey on graph anomaly detection with deep learning”. In: *IEEE Transactions on Knowledge and Data Engineering*.

Martins, Inês, Resende, João S, Sousa, Patrícia R, Silva, Simão, Antunes, Luís, and Gama, João (2022). “Host-based IDS: A review and open issues of an anomaly detection system in IoT”. In: *Future Generation Computer Systems.*

Mills, Ryan, Marnerides, Angelos K, Broadbent, Matthew, and Race, Nicholas (2021). “Practical intrusion detection of emerging threats”. In: *IEEE Transactions on Network and Service Management* 19.1, pp. 582–600.

Moustafa, Nour, Ahmed, Mohiuddin, and Ahmed, Sherif (2020). “Data analytics-enabled intrusion detection: Evaluations of ToN_IoT linux datasets”. In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom).* IEEE, pp. 727–735.

Moustafa, Nour, Hu, Jiankun, and Slay, Jill (2019). “A holistic review of network anomaly detection systems: A comprehensive survey”. In: *Journal of Network and Computer Applications* 128, pp. 33–55.

Moustafa, Nour and Slay, Jill (2015). “UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)”. In: *2015 military communications and information systems conference (MilCIS).* IEEE, pp. 1–6.

Narayanan, Annamalai, Chandramohan, Mahinthan, Venkatesan, Rajasekar, Chen, Lihui, Liu, Yang, and Jaiswal, Shantanu (2017). “graph2vec: Learning distributed representations of graphs”. In: *arXiv preprint arXiv:1707.05005.*

Ngo, Quoc-Dung, Nguyen, Huy-Trung, Le, Van-Hoang, and Nguyen, Doan-Hieu (2020). “A survey of IoT malware and detection methods based on static features”. In: *ICT Express* 6.4, pp. 280–286.

Nguyen, Huy-Trung, Ngo, Quoc-Dung, and Le, Van-Hoang (2020). “A novel graph-based approach for IoT botnet detection”. In: *International Journal of Information Security* 19.5, pp. 567–577.

Özçelik, Rıza (Oct. 2019). *An intuitive explanation of graphsage.* URL: `https://towardsdatascience.com/an-intuitive-explanation-of-graphsage-6df9437ee64f`.

Pang, Guansong, Shen, Chunhua, Cao, Longbing, and Hengel, Anton Van Den (2021). "Deep learning for anomaly detection: A review". In: *ACM Computing Surveys (CSUR)* 54.2, pp. 1–38.

Papadogiannaki, Eva, Tsirantonakis, Giorgos, and Ioannidis, Sotiris (n.d.). "Network Intrusion Detection in Encrypted Traffic". In: ().

Paudel, Ramesh and Eberle, William (2020). "Snapsketch: Graph representation approach for intrusion detection in a streaming graph". In: *Proceedings of the 16th International Workshop on Mining and Learning with Graphs (MLG).*

Paudel, Ramesh, Muncy, Timothy, and Eberle, William (2019). "Detecting DoS attack in smart home IoT devices using a graph-based approach". In: *2019 IEEE International Conference on Big Data (Big Data).* IEEE, pp. 5249–5258.

Perozzi, Bryan, Al-Rfou, Rami, and Skiena, Steven (2014). "Deepwalk: Online learning of social representations". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710.

Schindler, Timo (2018). "Anomaly detection in log data using graph databases and machine learning to defend advanced persistent threats". In: *arXiv preprint arXiv:1802.00259.*

Sharafaldin, Iman, Lashkari, Arash Habibi, and Ghorbani, Ali A (2018). "Toward generating a new intrusion detection dataset and intrusion traffic characterization." In: *ICISSp* 1, pp. 108–116.

Smys, S, Basar, Abul, Wang, Haoxiang, et al. (2020). "Hybrid intrusion detection system for internet of things (IoT)". In: *Journal of ISMAC* 2.04, pp. 190–199.

Suthaharan, Shan, Alzahrani, Mohammed, Rajasegarar, Sutharshan, Leckie, Christopher, and Palaniswami, Marimuthu (2010). "Labelled data collection for anomaly detection in wireless sensor networks". In: *2010 sixth international conference on intelligent sensors, sensor networks and information processing*. IEEE, pp. 269–274.

Svacina, Jan, Raffety, Jackson, Woodahl, Connor, Stone, Brooklynn, Cerny, Tomas, Bures, Miroslav, Shin, Dongwan, Frajtak, Karel, and Tisnovsky, Pavel (2020). "On vulnerability and security log analysis: A systematic literature review on recent trends". In: *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, pp. 175–180.

Tang, Jian, Qu, Meng, Wang, Mingzhe, Zhang, Ming, Yan, Jun, and Mei, Qiaozhu (2015). "Line: Large-scale information network embedding". In: *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077.

Tavallaee, Mahbod, Bagheri, Ebrahim, Lu, Wei, and Ghorbani, Ali A (2009). "A detailed analysis of the KDD CUP 99 data set". In: *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, pp. 1–6.

Tong, Flawnson (Nov. 2020). *Overview of deep learning on graph embeddings*. URL: `https://towardsdatascience.com/overview-of-deep-learning-on-graph-embeddings-4305c10ad4a4`.

Wang, Daixin, Cui, Peng, and Zhu, Wenwu (2016). "Structural deep network embedding". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1225–1234.

Wendlandt, D (2010). *Nessus: A security vulnerability scanning tool. Accessed: Jan. 21, 2020. [Online]. Available: https://www.cs.cmu.edu/ dwendlan/personal/nessus.html.*

Zhou, Wei, Gao, Yuan, Ji, Jianhang, Li, Shicheng, and Yi, Yugen (2021). "Unsupervised Anomaly Detection for Glaucoma Diagnosis". In: *Wireless Communications and Mobile Computing* 2021.

Zhu, Qianwen, Zhou, Jinyu, Zhao, Shiyu, and Wang, Wei (2022). “Graph-Based Anomaly Detection of Wireless Sensor Network”. In: *Artificial Intelligence in China*. Springer, pp. 144–150.