CYBER THREAT DETECTION & ANALYSIS USING ML & DL

ST. TERESA'S COLLEGE(AUTONOMOUS)

AFFILIATED TO MAHATMA GANDHI UNIVERSITY

PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree of

BCA (CLOUD TECHNOLOGY & INFORMATION SECURITY MANAGEMENT)

By

MICHELLE CELINE PEREIRA – SB21BCA025

&

SONA N SHOJAN – SB21BCA037

III DC BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY MANAGEMENT)

Under the guidance of

SHREELAKSMI I.J

DEPARTMENT OF BCA (CLOUD TECHNOLOGY & INFORMATION SECURITY MANAGEMENT)

MARCH 2024

CYBER THREAT DETECTION & ANALYSIS USING ML & DL

ST. TERESA'S COLLEGE(AUTONOMOUS)

AFFILIATED TO MAHATMA GANDHI UNIVERSITY

PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree of

BCA (CLOUD TECHNOLOGY & INFORMATION SECURITY MANAGEMENT)

By

MICHELLE CELINE PEREIRA – SB21BCA025

&

SONA N SHOJAN – SB21BCA037

III DC BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY MANAGEMENT)

Under the guidance of

SREELAKSMY I.J

DEPARTMENT OF BCA (CLOUD TECHNOLOGY & INFORMATION SECURITY MANAGEMENT)

MARCH 2024

DECLARATION

We, undersigned, hereby declare that the project report, **'CYBER THREAT DETECTION & ANALYSIS USING ML & DL'**, submitted for partial fulfillment of the requirements for the award of degree of BCA (Cloud Technology and Information Security Management)at St. Teresa's College (Autonomous),Ernakulam (Affiliated to Mahatma Gandhi University), Kerala, is a bonafide work done by us under the supervision of Sreelakshmy I.J. This submission represents our ideas in our own words and where ideas or words of others have not been included. We have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Ernakulam — MICHELLE CELINE PEREIRA – SB21BCA025

March 2024 — SONA N SHOJAN – SB21BCA037

ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM

BCA (CLOUD TECHNOLOGY AND INFORMATION SECURITY MANAGEMENT)

DEPARTMENT OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the report entitled "**CYBER THREAT DETECTION & ANALYSIS USING ML & DL**", submitted by Michelle Celine Pereira & Sona N Shojan to the Mahatma Gandhi University in partial fulfillment of the requirements for the award of the Degree of BCA (Cloud Technology and Information Security Management) is a bonafide record of the project work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

[illegible]

ARCHANA MENON P

Head of the department

[illegible] 20/03/2024

SREELAKSHMY I.J

Internal Supervisor

ST. TERESA'S COLLEGE
DEPT. OF COMPUTERS
ERNAKULAM

[illegible]

External Supervisor

ACKNOWLEDGEMENT

First and foremost, we thank God Almighty for his blessings. We take this opportunity to express our gratitude to all those who helped us in completing this project successfully. I wish to express our sincere gratitude to the **Manager Rev. Dr. Sr. Vinitha CSST** and the Principal **Dr. Alphonsa Vijaya Joseph** for providing all the facilities.

We express our sincere gratitude towards the Head of the department **Ms. Archana Menon P** for the support. We deeply express sincere thanks to our project guide **Ms. Sreelakshmy I. J** for her proper guidance and support throughout the project work.

We are indebted to our beloved teachers whose cooperation and suggestion throughout the project which helped us a lot. We thank all our friends and classmates for their support.

We convey our hearty thanks to our parents for the moral support, suggestion and encouragement.

ABSTRACT

The use of cyberspace is rising with each passing day. People are spending more time on the Internet than ever before. As a result, the risks of cyber threats and cyber-crimes are increasing. Cybercriminals are changing their techniques with time to pass through the wall of protection. Machine Learning and Deep Learning are two advanced technologies to develop advanced tools. This project focuses on developing a cyber threat detection system leveraging Machine Learning algorithms like XGBoost which are renowned for their ability to discern complex patterns and anomalies in large datasets. The rationale behind this endeavor stems from the escalating sophistication and frequency of cyber-attacks, necessitating proactive defense mechanisms capable of adapting to evolving threats. While traditional algorithms like K-Nearest Neighbors (KNN) and XGBoost have shown efficacy in certain contexts, the rise of deep learning approaches, particularly Generative Adversarial Networks (GANs), presents a promising avenue for enhancing detection capabilities. By training various algorithms on network attack datasets and comparing their performance, this project aims to discern the strengths and limitations of each approach. Furthermore, the utilization of GANs to generate synthetic attack data introduces a novel dimension to the evaluation process, enabling the assessment of models' robustness against adversarial examples. Through this holistic approach, the project seeks to advance the field of cyber threat detection by harnessing the power of deep learning and fostering resilience against emerging cyber threats.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

DL - Deep Learning

ML - Machine Learning

DNN – Deep Neural Network

KNN – K Nearest Neighbor

GAN – Generative Adversarial Networks

IoT – Internet of Things

AI – Artificial Intelligence

SVM – Support Vector Machines

ANN – Artificial Neural Network

AML – Adverbial Machine Learning

EDA – Explanatory Data Analysis

Chapter 1
INTRODUCTION

Cybersecurity stands as one of the paramount concerns in our contemporary digital age. With the rapid expansion of technology and the pervasive integration of digital systems into every facet of modern life, the threat landscape facing individuals, organizations, and nations has become increasingly complex and dynamic. Cyber-attacks, ranging from data breaches and ransomware infections to sophisticated hacking operations and state-sponsored espionage, pose significant risks to the confidentiality, integrity, and availability of digital assets. In response to this escalating threat environment, the development of robust and adaptive cyber threat detection systems has emerged as a critical imperative. These systems play a pivotal role in proactively identifying, mitigating, and neutralizing cyber threats before they can inflict substantial harm. However, devising effective detection mechanisms that can keep pace with the rapidly evolving tactics and techniques employed by cyber adversaries presents a formidable challenge. Against this backdrop, our project endeavors to address this challenge by exploring the application of advanced machine learning and deep learning techniques for cyber threat detection. Leveraging the power of artificial intelligence (AI) and data analytics, our aim is to develop and evaluate sophisticated detection models capable of discerning subtle patterns and anomalies indicative of malicious activities within vast and complex datasets.

The objectives of our project are multifold. Firstly, we seek to investigate the efficacy of various machine learning algorithms, including traditional classifiers like K-Nearest Neighbors (KNN) and ensemble methods like XGBoost, in detecting and classifying different types of cyber-attacks. Additionally, we aim to explore the potential of deep learning architectures, such as Deep Neural Network (DNN), in capturing intricate patterns inherent in cyber threat data. Furthermore, our project seeks to push the boundaries of traditional evaluation methodologies by incorporating novel approaches such as the use of Generative Adversarial Networks (GANs) to generate synthetic attack data for model testing. By subjecting our detection models to both real-world and synthetic attack scenarios, we endeavor to assess their robustness, generalization capabilities, and susceptibility to adversarial examples. In pursuit of these objectives, our project follows a systematic methodology encompassing data collection, preprocessing, model development, training, and evaluation phases. We draw upon diverse datasets containing samples of various cyber-attacks, ensuring comprehensive coverage of the threat landscape. Through iterative experimentation and rigorous evaluation, we aim to gain insights into the strengths and limitations of different detection approaches, thereby informing the development of more effective and adaptive cyber threat detection systems.

1.1 History of Cyber Threats

The history of cyber threats is a narrative that spans the evolution of technology and the proliferation of digital systems, reflecting the continuous interplay between innovation and vulnerability. Since the inception of computing, the landscape of cyber threats has undergone significant transformations, shaped by technological advancements, societal changes, and the motivations of malicious actors. The earliest instances of cyber threats can be traced back to the nascent stages of computing, where vulnerabilities in primitive systems were exploited by curious hackers and early cybercriminals. These early threats primarily manifested as isolated incidents of unauthorized access, data manipulation, and system intrusions, often driven by curiosity or personal challenge rather than malicious intent. However, as computing technology advanced and became more widespread, the nature and scale of cyber threats began to escalate. The emergence of networking protocols and the proliferation of interconnected systems gave rise to new attack vectors, enabling malicious actors to launch coordinated attacks across distributed networks. In the 1980s and 1990s, notable cyber incidents such as the Morris Worm, the Michelangelo virus, and the Melissa virus underscored the growing threat posed by malware and the potential for widespread disruption. The turn of the millennium marked a pivotal moment in the history of cyber threats, characterized by the rapid expansion of the internet and the increasing reliance on digital infrastructure for essential services and commerce. This period witnessed the emergence of sophisticated cybercriminal organizations, state-sponsored hacking groups, and underground marketplaces for cyber exploits and malware. Notable incidents such as the Code Red worm, the SQL Slammer worm, and the Conficker botnet highlighted the scale and impact of cyber-attacks on a global scale. In the ensuing years, cyber threats continued to evolve in complexity and sophistication, fueled by advancements in technology and the growing interconnectedness of digital systems. The proliferation of mobile devices, the rise of cloud computing, and the advent of the Internet of Things (IoT) introduced new attack surfaces and vulnerabilities, further expanding the threat landscape. Cyber-attacks became increasingly targeted, stealthy, and financially motivated, with the proliferation of ransomware, banking Trojans, and advanced persistent threats (APTs) posing significant challenges to cybersecurity professionals. Today, the landscape of cyber threats is characterized by a diverse array of adversaries, tactics, and motivations, ranging from nation-state actors engaged in espionage and sabotage to cybercriminals seeking financial gain and hacktivists pursuing ideological agendas. The scale and impact of cyber-attacks have escalated exponentially, with high-profile incidents such as the WannaCry ransomware attack, the SolarWinds supply chain attack, and the Colonial Pipeline ransomware attack highlighting the vulnerabilities inherent in modern digital infrastructure.

1.2 Types of Cyber Attacks

The realm of cyber-attacks encompasses a diverse array of tactics, techniques, and strategies employed by malicious actors to compromise, disrupt, or exploit digital systems and assets. Understanding the various types of cyber-attacks is essential for cybersecurity professionals, as it enables them to anticipate threats, implement appropriate defenses, and respond effectively to incidents. From common, low-complexity attacks to sophisticated, nation-state-sponsored operations, the cyber threat landscape is dynamic and ever-evolving. One prevalent type of cyber-attack is malware, which refers to malicious software designed to infiltrate, damage, or control computer systems without the consent of the owner. Malware comes in various forms, including viruses, worms, Trojans, ransomware, and spyware, each with its own unique characteristics and objectives. Viruses, for example, attach themselves to legitimate files or programs and replicate themselves when executed, while ransomware encrypts files or systems and demands payment for their release. Phishing attacks represent another pervasive threat, involving the use of fraudulent emails, websites, or messages to deceive individuals into divulging sensitive information such as passwords, financial data, or personal details. Phishing attacks often exploit human psychology and social engineering techniques to elicit trust or fear, convincing unsuspecting victims to click on malicious links, download malicious attachments, or disclose confidential information. Distributed Denial of Service (DDoS) attacks constitute yet another significant threat vector, aimed at disrupting the availability of online services by overwhelming target systems with a flood of illegitimate traffic. DDoS attacks can range from simple volumetric attacks that flood network bandwidth to more sophisticated application-layer attacks that exploit vulnerabilities in web applications or server infrastructure.

1.3 Technologies and Methods in Cyber Threat Detection

In the ever-evolving landscape of cybersecurity, a wide array of technologies and methodologies are employed to detect, mitigate, and respond to cyber threats. These approaches encompass diverse techniques ranging from traditional signature-based methods to advanced machine learning and artificial intelligence (AI) algorithms. By leveraging a combination of these technologies and methods, organizations can enhance their ability to detect and thwart cyber-attacks effectively.

Signature-based detection methods represent one of the oldest and most widely used approaches in cybersecurity. These methods rely on predefined signatures or patterns of known threats to identify and block malicious activity. Signature-based detection is effective against well-established malware strains and known attack vectors but may struggle to detect novel or previously unseen threats. Anomaly detection techniques offer a complementary approach to signature-based methods by focusing on identifying deviations from normal behavior within a system or network. By establishing baseline behavior patterns and monitoring for deviations that may indicate potential threats, anomaly detection systems can detect previously unknown or zero-day attacks. However, anomaly detection methods may also suffer from high false-positive rates and require careful tuning to minimize false alarms. Behavioral analysis represents another powerful approach to cyber threat detection, focusing on analyzing the behavior of users, applications, and network traffic to identify suspicious or malicious activity. Behavioral analysis techniques leverage machine learning algorithms to analyze large volumes of data and detect patterns indicative of cyber threats. By monitoring for deviations from expected behavior and identifying anomalous activities, behavioral analysis systems can detect sophisticated attacks that evade traditional signature-based defenses. Machine learning and artificial intelligence (AI) have emerged as game-changing technologies in the field of cyber threat detection. These techniques enable the development of advanced algorithms capable of learning from data and making intelligent decisions without explicit programming. Machine learning algorithms such as decision trees, random forests, support vector machines (SVMs), and neural networks can analyze vast amounts of data to identify patterns and anomalies indicative of cyber threats. Deep learning, a subset of machine learning that utilizes neural networks with multiple layers of abstraction, has shown particular promise in cybersecurity applications. Deep learning algorithms, Deep Neural Network (DNN), excel at extracting complex features from unstructured data such as images, text, and network traffic. These algorithms can detect subtle patterns and correlations in cyber threat data, enabling more accurate and robust detection capabilities.

1.4 Current Challenges in Cybersecurity

As technology continues to advance at a rapid pace and cyber threats become increasingly sophisticated and pervasive, cybersecurity professionals must grapple with a range of complex challenges that demand innovative solutions and proactive approaches. One of the foremost challenges facing cybersecurity practitioners is the rapid evolution of cyber threats. Malicious actors constantly adapt their tactics, techniques, and procedures (TTPs) to exploit vulnerabilities in technology and human behavior. The proliferation of new attack vectors, such as ransomware, supply chain attacks, and zero-day exploits, presents a formidable challenge for defenders, who must continually update their defenses to keep pace with emerging threats. Another significant challenge is the growing complexity of modern IT environments. As organizations embrace digital transformation initiatives and adopt new technologies such as cloud computing, mobile devices, and Internet of Things (IoT) devices, the attack surface expands, creating new vulnerabilities and entry points for attackers. Securing these diverse and interconnected systems requires a holistic approach that addresses both technical and organizational challenges. The shortage of skilled cybersecurity professionals represents a critical challenge that exacerbates the cybersecurity skills gap. As the demand for cybersecurity expertise continues to outpace supply, organizations struggle to recruit and retain qualified professionals with the necessary skills and experience to effectively defend against cyber threats. This shortage of talent limits the capacity of organizations to implement robust cybersecurity programs and leaves them vulnerable to attacks. The rapid pace of technological innovation also poses challenges for cybersecurity practitioners. Emerging technologies such as artificial intelligence (AI), quantum computing, and blockchain introduce new opportunities for both defenders and attackers. While AI and machine learning hold promise for enhancing cyber threat detection and response capabilities, they also introduce new risks, such as adversarial attacks and AI-driven malware.

Chapter 2
LITERATURE SURVEY

The literature survey has been conducted on different papers such as technical papers and review papers in the domain published in leading publications, journals and conferences. Keywords such as cyber threat, attack detection, GAN, Machine Learning, etc. are used to filter out.

D. O. Won et al. laid the groundwork for addressing the pervasive threat posed by zero-day malicious software (malware), denoting previously unknown or newly discovered software vulnerabilities. With the primary aim of enhancing detection capabilities for analogous zero-day malware instances, this paper introduces an innovative approach centered on efficient learning from plausibly generated data. The proposed malware training framework leverages generative adversarial networks (GANs), specifically PlausMal-GAN, to generate analogous zero-day malware data efficiently. PlausMal-GAN operates by synthesizing high-quality and diverse zero-day malware images based on existing malware data, thus enabling the discriminator, acting as a detector, to learn various malware features from both real and generated malware images. The discriminator's role in distinguishing between real and generated malware images enhances the framework's ability to detect analogous zero-day malware instances effectively. Performance evaluations of the proposed framework demonstrate superior and more stable detection performances for analogous zero-day malware images, indicative of the framework's efficacy in handling analogous zero-day malware data. Notably, the PlausMal-GAN framework exhibits reliable accuracy performances across representative GAN models, including deep convolutional GAN, least-squares GAN, Wasserstein GAN with gradient penalty, and evolutionary GAN. These findings underscore the potential benefits of the proposed framework in enhancing the detection and prediction of numerous and analogous zero-day malware instances. By enabling the development and refinement of malware detection systems capable of effectively identifying and mitigating emerging threats, the PlausMal-GAN framework represents a significant advancement in cybersecurity research and practice. Its ability to generate high-quality analogous zero-day malware data holds promise for bolstering the resilience of malware detection systems and fortifying defenses against evolving cyber threats [3]. ML systems are increasingly trusted in cyber-physical systems [7], including factories, power plants, and the oil and gas industries. In such complex physical surroundings, a threat that manages to get through a weak system could be harmful [8]. Despite the dependence on and faith in ML systems, attackers who want to avoid ML-based system discovery processes may use the inherent nature of ML, learning to recognize patterns, as a possible attack component [9]. As a result, attackers craft malicious inputs called "adversarial samples." Adversarial samples are constructed by intentionally adding minor perturbations to initial inputs, which results in the misclassification of the ML/DL models. Adversarial machine learning (AML) based on the National Institute of Standards and Technology (NIST) is divided into four attacks, which are: evasion, extraction, poisoning, and inference [1]. Hence, the misclassification of ML initially appeared approximately two decades ago and has piqued researchers' interest. The researchers in [2] deceived spam classifiers into injecting some changes into an email. Moreover, it is even older than 38 years, according to the authors in [1], when they showed that false fingerprints might be made with plastic-like materials to deceive biometric identity recognition systems. Along with ML technology's significant advancement in network security, it exposes a new attack surface for attackers. Accordingly, the IDS is susceptible to adversarial attacks since it is built on ML, which could be compromised by crafting adversarial input against ML/DL models such as the artificial neural network (ANN), the deep neural network (DNN), and the support vector machine (SVM), affecting its accuracy and robustness. Furthermore, research has also demonstrated that adversarial samples could affect ML-based IDSs [10,14]. As a result, ML can also be fooled, necessitating some protection mechanisms. Additionally, the system becomes susceptible due to communication on the open network, which also gives enemies a massive attack surface [5].

Many surveys present AML in various domains, such as computer vision, which recently received much attention, and network security. Major previous studies focused on adversarial attacks against ML and DL in various domains or the computer vision domain, such as in [4, 3]. Additionally, other surveys take this topic from a game perspective, making it more straightforward for the reader, such as [1], which presents a general view of the arms race between adversarial attacks and defense methods and how they constantly try to defeat each other. In addition, [7] presented more details about adversarial attacks and defense methods from a cybersecurity perspective. Furthermore, ML security has received much attention, with many researchers mentioning the dangers of adversarial attacks on ML and the defense methods described in [3]. This survey clarified the various types of adversarial attacks and the defense methods to protect ML. However, this study highlighted the ML adversaries and primary defenses; it was not specialized in specific ML methods in cybersecurity, such as malware detection. On the other hand, the authors of [5] had to dig deeper into the network security domain. This study has more than the original view. It presents detailed information for network security applications in ML and adversarial attacks against them, in addition to defense methods against these attacks. However, it is not connected to something special such as phishing or spam detection. The research in [6] presented adversarial attacks in cybersecurity, such as intrusion detection, which provided a more detailed perspective, discussed attacks, and offered some defense strategies. In general, the researchers found this study helpful in providing the basis for the issue of adversaries and defenses against ML-based network applications. Despite this study's insightful ideas, its main focus is on keeping adversarial attacks functioning so they can continue avoiding ML classifiers. IDS is a type of computer security software that seeks to identify a wide range of security breaches, from attempted break-ins by outsiders to system penetrations by insiders [2]. Furthermore, the essential functions of IDSs are to monitor hosts and networks, evaluate computer system activity, produce warnings, and react to abnormal behavior [5]. Moreover, one of the significant constraints of typical intrusion detection systems (IDS) is filtering and decreasing false alarms [4]. In addition, many IDSs improve their performance by utilizing neural networks (NN) for deep learning. Furthermore, deep neural network (DNN)-based IDS systems have been created to improve tremendous data learning, processing, and a range of assaults for future prediction [6].

Alotaibi A et al. highlighted the escalating concerns surrounding cybersecurity and the surge in attack methods prevalent in the information age. Among the techniques deployed to combat these threats, intrusion detection systems (IDSs) play a pivotal role, striving to detect and classify malici ous activities before they infiltrate systems. However, conventional IDS approaches exhibit limitations, particularly in accurately classifying novel attacks and adapting to dynamic environments, resulting in decreased accuracy and elevated false alarm rates. Consequently, researchers have advocated for the integration of machine learning techniques within IDSs to bolster their efficacy. Machine learning models offer the promise of autonomously discerning between normal and malicious data, including novel attack types, with heightened precision. Nonetheless, the susceptibility of these models to adversarial input perturbations during training or testing poses a significant challenge, potentially undermining their predictive capabilities and classifications. Adversarial machine learning (AML) poses a formidable cybersecurity threat across various sectors reliant on machine learning-based classification systems. In particular, AML techniques have the potential to deceive IDSs, leading to misclassification of network packets and compromising system security. In response to these challenges, this paper presents a comprehensive survey of adversarial machine-learning strategies and corresponding defense mechanisms. It commences by elucidating diverse types of adversarial attacks capable of undermining IDS functionality, thereby highlighting the critical need for robust defense strategies.

Subsequently, the paper delineates various defense mechanisms designed to mitigate or eliminate the influence of adversarial attacks on machine learning models and IDSs. These defense strategies encompass a spectrum of approaches, ranging from adversarial training and robust optimization to ensemble methods and anomaly detection techniques [7].

A. Shi et al. illuminated the profound impact of modern technologies, particularly the Internet of Things (IoT), on society, concurrently highlighting the escalating cybersecurity challenges stemming from these advancements. With the proliferation of network attacks, including botnet and distributed denial of service (DDoS) attacks, detecting and mitigating cyber threats has become increasingly complex and critical for maintaining the normal functioning of social infrastructure. Consequently, solutions leveraging artificial intelligence (AI) techniques have emerged as promising approaches to bolster cybersecurity defenses and thwart malicious activities effectively. Central to this endeavor is the application of Generative Adversarial Networks (GANs) in the domain of network security. GANs have garnered considerable attention as one of the most influential deep learning models, distinguished by their dynamic game-theoretic framework comprising a generative model and a discriminative model. This unique architecture enables GANs to generate synthetic data samples that closely resemble real data distributions, facilitating diverse applications across various domains, including image generation, speech processing, data augmentation, and cyberattack detection. This paper elucidates the fundamental working principles and underlying infrastructure of GANs, shedding light on their potential utility in augmenting cyber intrusion detection capabilities. By harnessing the discriminative power of GANs, cybersecurity practitioners can enhance their ability to detect and mitigate cyber threats, particularly those characterized by stealthy and sophisticated attack vectors. Furthermore, the integration of GANs into cyber defense frameworks holds promise for improving the resilience and adaptability of intrusion detection systems (IDSs) in dynamic and evolving threat landscapes. Through a comprehensive examination of GANs' applications in cyber intrusion detection, this paper underscores the significance of leveraging AI-driven approaches to bolster network security defenses. By harnessing the capabilities of GANs to generate realistic synthetic data and discern subtle patterns indicative of malicious activities, cybersecurity professionals can enhance their ability to safeguard critical infrastructure and mitigate the detrimental impacts of cyber threats on society. Moving forward, continued research and development efforts in this area are crucial for advancing the state-of-the-art in cyber defense and effectively combating emerging cyber threats [8]. X. Hao et al underscored the efficacy of machine learning techniques in network attack detection systems for identifying malicious network behaviors. However, they highlighted the challenges posed by the concealment of network attack traffic within the vast volume of everyday communication traffic in real-world environments. To address these challenges, the authors proposed a novel data augmentation approach based on generative adversarial networks (GANs). In their approach, the features of flow-based network traffic are preprocessed to align with the requirements of GANs. Subsequently, the authors enhanced the original GANs by incorporating Earth-Mover (EM) distance to capture the distribution of low-dimensional subspace data. Additionally, an encoder structure was introduced to facilitate the learning of latent space representation. Unlike traditional data augmentation methods that rely on numerical calculations on existing data, their approach generates data by learning the underlying data distribution. The authors conducted experiments using an imbalanced dataset derived from real-world data and compared the performance of their method with alternative approaches. Their method demonstrated superior performance in terms of recall, F1-score, and area under the receiver operating characteristic curve (AUC), indicating its effectiveness in addressing the challenges of data augmentation in network attack detection systems.[9].

Y. Xiong et al. odel accuracy or computational overhead. In this context, the authors aimed to mitigate privacy issues in a non-intrusive manner by proactively detecting GAN-based attackers at the onset of training. Their proposed detection mechanism leverages only the gradient updates uploaded by participants during training, rendering it transparent to participants and obviating the need for protocol changes. Through extensive experiments conducted across various settings and attack scenarios, the authors demonstrated the effectiveness of their detection approach. Overall, their work represents a significant contribution to enhancing the security and privacy of distributed learning frameworks. By adopting a proactive approach to detect potential attackers, the proposed method offers a transparent and efficient means of safeguarding against privacy breaches without compromising training performance. This research sets the stage for further exploration and development of robust security mechanisms in distributed learning environments, ensuring the integrity and privacy of participants' data in collaborative training efforts.[10].

Chapter 3
EXISTING SYSTEM

Existing cyber threat detection systems encompass a variety of approaches include :

3.1 Conventional Systems

Conventional systems have long served as foundational pillars in the cybersecurity landscape, employing established techniques and methodologies to detect and mitigate cyber threats. These systems, characterized by their reliance on predefined rules, signatures, and patterns, have played a crucial role in safeguarding digital assets and infrastructure from malicious activities. Within the realm of conventional systems, several key approaches and methodologies are commonly employed, each addressing distinct aspects of cyber threat detection and prevention. One of the fundamental methods utilized by conventional systems is request packet examination, which involves inspecting incoming network packets to identify potentially malicious traffic. This process typically entails examining packet headers and payloads for known signatures or indicators of compromise. By scrutinizing network traffic at the packet level, these systems can identify suspicious patterns or anomalies indicative of cyber threats, such as port scanning, denial-of-service attacks, or malware propagation. Another prevalent technique employed by conventional systems is rule-based blocking, where security policies are enforced through the application of predefined rules or conditions. These rules dictate the permissible behavior of network traffic based on criteria such as source IP addresses, destination ports, protocol types, or payload content. When incoming traffic matches a rule's criteria, the system takes action to either allow, block, or redirect the traffic according to the specified policy. Rule-based blocking provides a flexible and configurable framework for enforcing security policies and protecting against known attack vectors. Pattern matching and blocking represent another core capability of conventional systems, particularly in the context of intrusion detection and prevention. These systems utilize pattern matching algorithms to identify known attack signatures or patterns within network traffic. By comparing incoming data against a database of predefined signatures, the system can detect and block malicious activity in real-time. Pattern matching techniques are effective against well-established threats and known attack vectors but may struggle to detect novel or previously unseen attacks that lack predefined signatures.

3.2 Based on request packet examination

Based on request packet examination, conventional cybersecurity systems employ a fundamental technique to scrutinize incoming network packets, aiming to identify and mitigate potential cyber threats. This method involves the meticulous inspection of packet headers and payloads, allowing for the detection of anomalies, malicious patterns, or indicators of compromise. By examining network traffic at the packet level, these systems can gain insights into the nature and intent of data transmissions, enabling proactive defense measures to be implemented. At its core, request packet examination relies on the analysis of various attributes within network packets, including source and destination IP addresses, port numbers, protocol types, packet size, and payload content. These attributes provide valuable metadata that can be leveraged to discern legitimate traffic from potentially malicious activity. For example, anomalies such as unusually large packet sizes or unexpected protocol types may indicate attempts to exploit vulnerabilities or launch denial-of-service attacks. One of the primary objectives of request packet examination is to detect and thwart common cyber threats, such as port scanning, reconnaissance activities, and network-based attacks. Port scanning, for instance, involves probing target systems to identify open ports and potential vulnerabilities that can be exploited. By monitoring incoming packets for sequential port scans or unusual connection patterns, cybersecurity systems can flag suspicious behavior and block further access attempts.

3.3 Rule based blocking

Pattern matching and blocking represent a core methodology within conventional cybersecurity systems, providing a mechanism for identifying known attack signatures or patterns within network traffic and taking proactive measures to block or mitigate potential threats. This approach relies on the comparison of incoming data against a database of predefined signatures or patterns, enabling the system to detect and respond to malicious activity in real-time. At its essence, pattern matching and blocking involve the creation and maintenance of a signature database that catalogs known indicators of compromise, malware signatures, or anomalous behavior associated with cyber threats. These signatures may encompass various attributes of network traffic, including packet headers, payload contents, protocol behaviors, and communication patterns. By continuously updating and refining this signature database, cybersecurity systems can stay abreast of emerging threats and adapt to evolving attack techniques. One of the primary benefits of pattern matching and blocking is its effectiveness in detecting well-established cyber threats and known attack vectors. By comparing incoming data against a comprehensive library of signatures, cybersecurity systems can rapidly identify and block malicious activity, preventing unauthorized access, data exfiltration, or system compromise. Common examples of signatures used in pattern matching include virus definitions, intrusion detection rules, and network behavior anomalies. However, while pattern matching and blocking are effective against known threats and established attack techniques, they also exhibit certain limitations and challenges. One notable challenge is the reliance on static, predefined signatures, which may fail to detect novel or previously unseen attacks that lack known signatures. As cyber threats continue to evolve in sophistication and complexity, signature-based detection methods may struggle to keep pace with emerging threats, leading to potential gaps in cybersecurity defenses.

3.4 Pattern matching and blocking.

Pattern matching and blocking serve as fundamental components within conventional cybersecurity systems, providing essential capabilities for identifying and mitigating known cyber threats. This approach involves comparing incoming data against a database of predefined signatures or patterns to detect malicious activity and take proactive measures to block or mitigate potential threats. By leveraging pattern matching techniques, cybersecurity systems can rapidly identify and respond to known attack vectors, enhancing the overall security posture of organizations and safeguarding critical assets and infrastructure from cyber-attacks. At its core, pattern matching relies on the creation and maintenance of a signature database that catalogs known indicators of compromise, malware signatures, or anomalous behavior associated with cyber threats. These signatures encapsulate various attributes of network traffic, including packet headers, payload contents, protocol behaviors, and communication patterns. By continuously updating and refining this signature database, cybersecurity systems can stay abreast of emerging threats and adapt to evolving attack techniques, bolstering their ability to detect and mitigate cyber threats effectively. Pattern matching and blocking are particularly effective in detecting well-established cyber threats and known attack vectors. By comparing incoming data against a comprehensive library of signatures, cybersecurity systems can rapidly identify and block malicious activity, preventing unauthorized access, data exfiltration, or system compromise. For example, antivirus software relies on signature-based detection to identify and quarantine malicious files or processes based on known malware signatures. Similarly, intrusion detection and prevention systems (IDPS) use signature-based rules to detect and block network-based attacks, such as port scans, SQL injections, or buffer overflows.

3.5 Learning-based systems

Learning-based systems represent a paradigm shift in cybersecurity, harnessing the power of machine learning and artificial intelligence to detect and mitigate cyber threats more effectively. Unlike conventional signature-based approaches that rely on predefined rules or patterns, learning-based systems leverage algorithms that can analyze vast amounts of data, learn from it, and make intelligent decisions without explicit programming. This approach enables cybersecurity systems to adapt to evolving threats, identify previously unknown attack vectors, and enhance overall defense capabilities. One of the key advantages of learning-based systems is their ability to detect novel or previously unseen threats that may evade traditional signature-based detection methods. By analyzing large datasets of network traffic, system logs, and security events, machine learning algorithms can identify patterns, trends, and anomalies indicative of malicious activity. This enables cybersecurity systems to detect emerging threats, zero-day exploits, and sophisticated attack techniques that may not be captured by static signature-based approaches. Machine learning algorithms used in cybersecurity include a variety of techniques such as supervised learning, unsupervised learning, and reinforcement learning. Supervised learning algorithms learn from labeled training data, where each example is associated with a known outcome (e.g., benign or malicious). These algorithms can then classify new instances based on patterns learned from the training data. Unsupervised learning algorithms, on the other hand, analyze unlabeled data to identify hidden patterns or structures within the data. These algorithms are particularly useful for anomaly detection, where the goal is to identify deviations from normal behavior without explicit labels. Reinforcement learning algorithms learn through trial and error, receiving feedback from the environment based on their actions and adjusting their behavior to maximize rewards.

Chapter 4
PROPOSED SYSTEM

The proposed system is a Deep Learning and Machine Learning-based attack detection system that is trained well on the attack data. Initially the KDD NSL Dataset was collected from Kaggle. The NSL-KDD dataset, short for "NSL-KDD Network Intrusion Detection Dataset," is a benchmark dataset widely used in the field of network intrusion detection and cybersecurity research. It was developed as an extension of the original KDD Cup 1999 dataset to address certain limitations and challenges present in the earlier version. The dataset was introduced in the paper titled "Feature Selection for Intrusion Detection Using NSL-KDD Dataset" by M. Tavallaee et al. in 2009. This dataset is well prepared and is trained on algorithms like Random Forest and XGBoost.

Dataset 1:

Dataset Characteristics:

- The NSL-KDD dataset consists of network traffic data collected from a simulated environment, specifically designed to emulate real-world network traffic patterns and cyber-attacks.
- It contains a total of 41 features, including both categorical and numerical attributes, that describe various aspects of network connections and activities.
- The dataset is divided into two subsets: the training set and the test set. The training set contains approximately 125,000 instances, while the test set contains around 22,500 instances.
- Each instance in the dataset represents a network connection, and the goal is to classify these connections as either normal or malicious (intrusive).

Dataset 2:

The malware dataset

- Dataset collected from https://www.kaggle.com/datasets/dscclass/malware
- Basic Data analysis, and preprocessing done.
- Total number of samples = 138047
- Number of columns = 57

These two datasets will be first used to train Machine Learning and Deep Learning Models. Machine Learning models like Random Forest and XGBoost are trained to detect the attacks and Deep Neural Network is also trained.

The testing of the best performing model will be conducted in a different way. The same two datasets will be used to train a GAN model which will generate plausible new attack data that trained on. The Machine Learning model will be tested to detect these newly generated attack data.

GAN Model:

Generative Adversarial Networks (GANs) represent a revolutionary breakthrough in the field of artificial intelligence, particularly in the realm of generative modeling. Introduced by Ian Goodfellow and his colleagues in 2014, GANs have since become one of the most prominent and widely studied architectures in machine learning. At its core, a GAN consists of two neural networks: a generator and a discriminator. The generator is tasked with generating synthetic data samples, such as images or text, that resemble real data from a given distribution. On the other hand, the discriminator acts as a critic, distinguishing between real and fake samples. The two networks engage in an adversarial game, where the generator aims to produce increasingly realistic samples to fool the discriminator, while the discriminator strives to correctly classify real and fake samples. The beauty of GANs lies in their ability to generate highly realistic and diverse data samples, capturing complex patterns and structures present in the training data. GANs have been successfully applied in various domains, including image generation, text generation, video synthesis, and even drug discovery. They have enabled breakthroughs in computer vision, natural language processing, and many other fields, opening up new avenues for creativity and innovation. The GAN model will be trained on both the datasets and will be used as a test data generator for testing the Machine Learning models.

Chapter 5
SYSTEM DESIGN AND ARCHITECTURE

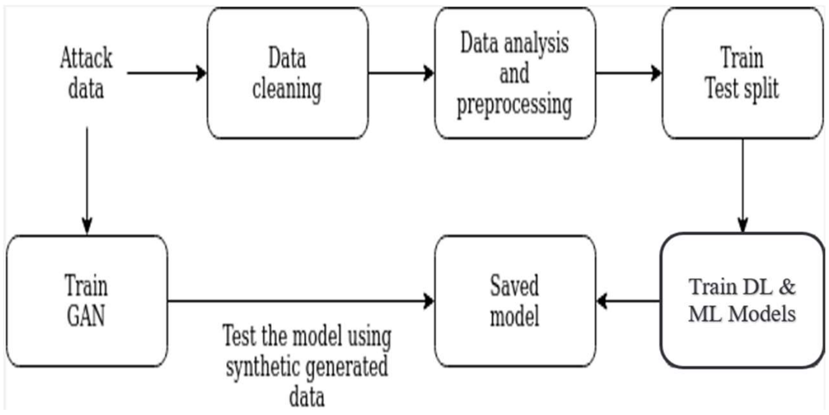


Figure 5.1: System Architecture.

The system architecture clearly shows how we use the GAN generator model to generate new attack data to test and evaluate the trained ML and DL models. The attack data will be used for training the ML and DL models as well as the GAN model.

5.1 Data Analysis and Preprocessing:

In the context of machine learning, analyzing and preprocessing this dataset involves several crucial steps to ensure data quality, feature extraction, and model readiness. Below, we delve into the technical details of these steps:

Data Understanding and Exploration:

Before diving into preprocessing, it's essential to gain a comprehensive understanding of the dataset's structure, attributes, and distribution. The KDD Cup '99 dataset consists of several features representing different aspects of network traffic, such as protocol type, service, source and destination IP addresses, and more. Exploratory data analysis (EDA) techniques, including statistical summaries, visualization, and correlation analysis, can provide insights into the dataset's characteristics, identify anomalies, and inform preprocessing decisions.

Data Cleaning:

Data cleaning is a crucial preprocessing step aimed at addressing missing values, outliers, and inconsistencies in the dataset. For the KDD Cup '99 dataset, common cleaning tasks may include:

Handling missing values: Identify and impute missing values using techniques such as mean imputation, median imputation, or predictive modeling.

Outlier detection: Identify outliers using statistical methods or domain knowledge and decide whether to remove or transform them.

Data normalization: Normalize numerical features to a common scale to prevent certain features from dominating others during model training.

Feature Engineering: Feature engineering involves transforming raw data into meaningful features that enhance model performance. In the context of the KDD Cup '99 dataset, feature engineering techniques may include:

One-hot encoding: Convert categorical variables into binary vectors to represent different categories.

Feature scaling: Scale numerical features to a specific range to ensure uniformity and improve model convergence.

Dimensionality reduction: Apply techniques such as principal component analysis (PCA) or feature selection to reduce the dimensionality of the dataset while preserving relevant information.

Handling Imbalanced Data: Imbalanced class distribution is common in intrusion detection datasets, where normal instances significantly outnumber attack instances.

To address this imbalance, various techniques can be employed, such as:

Resampling: Oversampling minority classes or under sampling majority classes to balance class distribution.

Synthetic data generation: Generate synthetic instances of minority classes using techniques like Synthetic Minority Over-sampling Technique (SMOTE) to augment the dataset.

Data Splitting:

Before training machine learning models, the dataset is typically divided into training, validation, and test sets. The training set is used to train the model, the validation set is used to tune hyperparameters and evaluate model performance during training, and the test set is used to assess the final model's generalization performance.

5.2 Machine Learning Models:

Random Forest Algorithm

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression. One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.

Real-Life Analogy Let's dive into a real-life analogy to understand this concept further. A student named X wants to choose a course after his 10+2, and he is confused about the choice of course based on his skill set. So, he decides to consult various people like his cousins, teachers, parents, degree students, and working people. He asks them varied questions like why he should choose, job opportunities with that course, course fee, etc.
Finally, after consulting various people about the course he decides to take the course suggested by most of the people. An effective alternative is to use trees with fixed structures and random features. Tree collections are called forests, and classifiers built-in so-called random forests. The random water formation algorithm requires three arguments: the data, a desired depth of the decision trees, and a number K of the total decision trees to be built, i. The algorithm generates each of the K trees. independent, which makes it very easy to parallelize. For each tree, build a complete binary tree.

The characteristics used for the branches of this tree are selected randomly, usually with replacement, which means that the same characteristic can occur more than 20 times, even in a single branch. a. the leaves of this tree, where predictions are made, are completed based on training data. The last step is the only point at which the training data is used. The resulting classifier is just a K-lot vote, and random trees. The most amazing thing about this approach is that it actually works remarkably well. They tend to work best when all the features are at least, well, relevant, because the number of features selected for a particular tree is small. One intuitive reason that it works well is the following. Some trees will query necessary features. These trees will essentially make random predictions. But some of the trees will happen to question good characteristics and make good predictions (because the leaves are estimated based on training data).

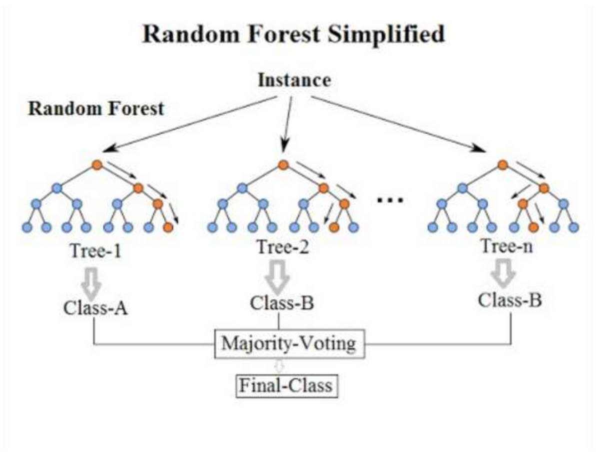


Figure 5.2: Random forest architecture

Steps involved in random forest algorithm:

Step 1: In Random Forest n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

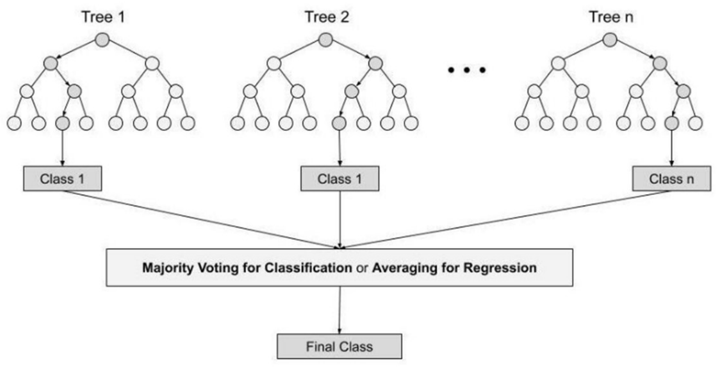

Tree 1
Tree 2
Tree n
Class 1
Class 1
Class n
Majority Voting for Classification or Averaging for Regression
Final Class

Figure 5.3: Random forest classifier procedures.

Important Hyperparameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster. Following hyperparameters increases the predictive power:

1. n_estimators– number of trees the algorithm builds before averaging the predictions.
2. max_features– maximum number of features random forest considers splitting a node.
3. mini_sample_leaf– determines the minimum number of leaves required to split an internal node.

Following hyperparameters increases the speed:

1. n_jobs– it tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor but if the value is -1 there is no limit.
2. random_state– controls randomness of the sample. The model will always produce the same results if it has a definite value of random state and if it has been given the same hyperparameters and the same training data.
3. oob_score – OOB means out of the bag. It is a random forest\

XGBoost Classifier

Ever since its introduction in 2014, XGBoost has been lauded as the holy grail of machine learning hackathons and competitions. From predicting ad click-through rates to classifying high energy physics events, XGBoost has proved its mettle in terms of performance – and speed. The beauty of this powerful algorithm lies in its scalability, which drives fast learning through parallel and distributed computing and offers efficient memory usage.

XGBoost is an ensemble learning method. Sometimes, it may not be sufficient to rely upon the results of just one machine learning model. Ensemble learning offers a systematic solution to combine the predictive power of multiple learners. The resultant is a single model which gives the aggregated output from several models. The models that form the ensemble, also known as base learners, could be either from the same learning algorithm or different learning algorithms. Bagging and boosting are two widely used ensemble learners. Though these two techniques can be used with several statistical models, the most predominant usage has been with decision trees.

Unique features of XGBoost

● Regularization: XGBoost has an option to penalize complex models through both L1 and L2 regularization. Regularization helps in preventing overfitting

● Handling sparse data: Missing values or data processing steps like one-hot encoding make data sparse. XGBoost incorporates a sparsity-aware split finding algorithm to handle different types of sparsity patterns in the data

● Weighted quantile sketch: Most existing tree-based algorithms can find the split points when the data points are of equal weights (using quantile sketch algorithm). However, they are not equipped to handle weighted data. XGBoost has a distributed weighted quantile sketch algorithm to effectively handle weighted data

● Block structure for parallel learning: For faster computing, XGBoost can make use of multiple cores on the CPU. This is possible because of a block structure in its system design. Data is sorted and stored in in-memory units called blocks. Unlike other algorithms, this enables the data layout to be reused by subsequent iterations, instead of computing it again. This feature also serves useful for steps like split finding and column sub-sampling
● Cache awareness: In XGBoost, non-contiguous memory access is required to get the gradient statistics by row index. Hence, XGBoost has been designed to make optimal use of hardware. This is done by allocating internal buffers in each thread, where the gradient statistics can be stored
● Out-of-core computing: This feature optimizes the available disk space and maximizes its usage when handling huge datasets that do not fit into memory

5.3 Deep Learning Models:

Deep Neural Network (DNN)

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship. The network moves through the layers calculating the probability of each output. For example, a DNN that is trained to recognize dog breeds will go over the given image and calculate the probability that the dog in the image is a certain breed.

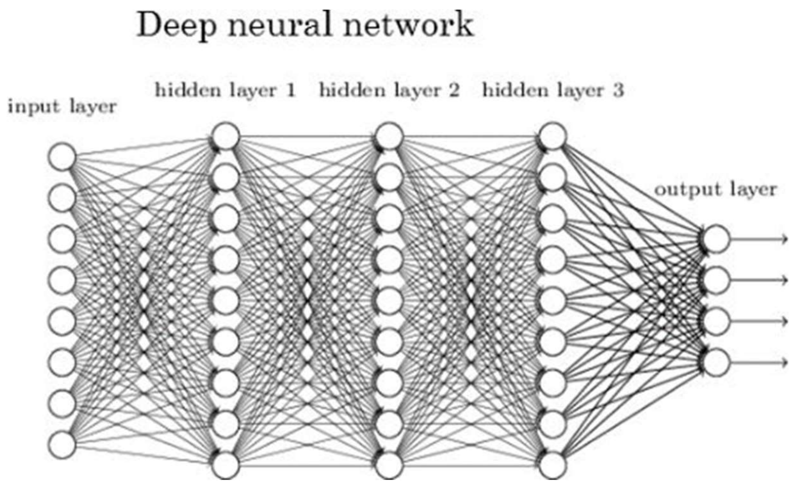


Figure 5.4: DNN

DNNs are typically feedforward networks in which data flows from the input layer to the output layer without looping back. At first, the DNN creates a map of virtual neurons and assigns random numerical values, or “weights”, to connections between them. The weights and inputs are multiplied and return an output between 0 and 1. If the network didn’t accurately recognize a particular pattern, an algorithm would adjust the weights. That way the algorithm can make certain parameters more influential, until it determines the correct mathematical manipulation to fully process the data.

5.4 Generative Adversarial Network:

Generative Adversarial Networks (GANs) represent a revolutionary class of deep learning models that have gained widespread attention and application across various domains, including computer vision, natural language processing, and cybersecurity. Introduced by Ian Goodfellow and his colleagues in 2014, GANs offer a powerful framework for generating realistic synthetic data samples that closely resemble real data distributions.

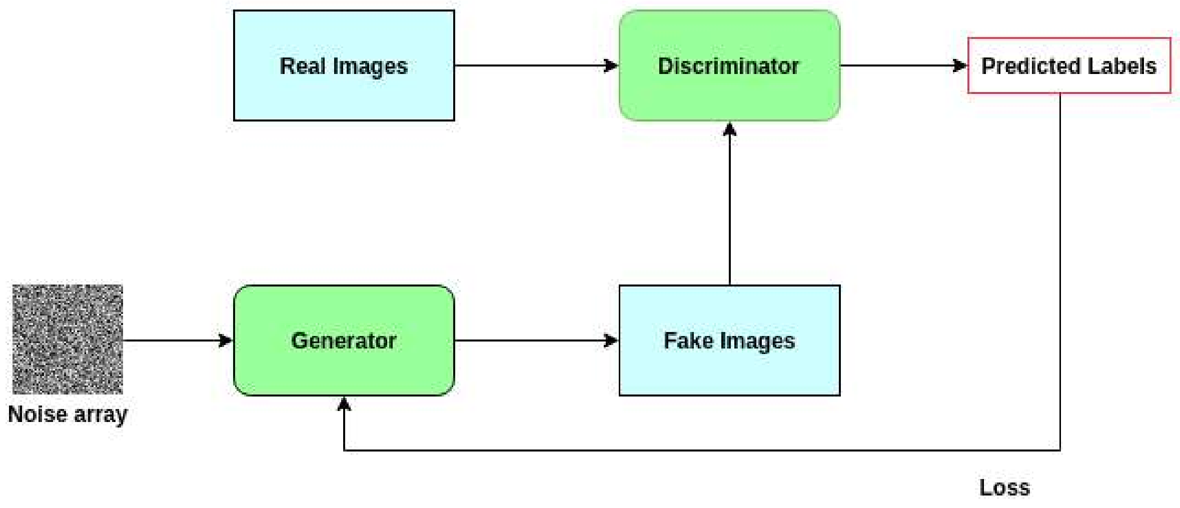


Figure 5.5: GAN model

At the heart of GANs lies a novel adversarial training scheme involving two neural networks: the generator and the discriminator. The generator network learns to generate synthetic data samples from random noise, aiming to produce outputs that are indistinguishable from real data. Concurrently, the discriminator network learns to differentiate between real and synthetic data samples, effectively acting as a critic that evaluates the authenticity of the generated samples. During training, the generator and discriminator engage in a minimax game, wherein the generator seeks to minimize the discriminator's ability to distinguish between real and synthetic data, while the discriminator strives to maximize its accuracy in discriminating between the two types of data. This adversarial interplay drives the generator to progressively improve its ability to generate realistic samples, ultimately converging to a point where the generated samples are indistinguishable from real data. The training process of GANs involves iteratively updating the parameters of both the generator and discriminator networks based on the feedback received from each other. This iterative optimization process typically involves gradient descent-based optimization algorithms such as stochastic gradient descent (SGD) or Adam optimizer.

Chapter 6
SYSTEM REQUIREMENTS

The system requirement is not that much for this project as the training has been carried out in Google Colab free version. So, no specific hardware is required. The design and development of the whole system of image processing and deep learning have been carried out in Google Colab Cloud platform.

Hardware requirement:

Basic system with intel i3 or above processor.

Software requirement:

IDE used for ML development and training - Google Colab.

Language used for ML development and training - Python 3.7 - 3.11

In addition to this various python libraries like TensorFlow for deep learning are also used.

Chapter 7
MODULE DESCRIPTION

This project can easily be divided into multiple modules as it involves multiple models and datasets. So, the first module, the Data collection and preprocessing module, involves in the data collection and extensive preprocessing of the data.

7.1 Module 1: Data Collection and Preprocessing

In the first module of the project, the focus is on gathering the necessary data and preparing it for subsequent analysis and modeling. This module involves several key steps, each of which contributes to ensuring the quality and suitability of the data for the intended machine learning tasks. Here's a breakdown of the technical content involved in this module:

Data Collection:

Identify relevant sources of data, which may include public datasets, proprietary databases, or data scraped from online sources.

Implement data collection mechanisms such as web scraping, API calls, or direct downloads to acquire the required data.

Ensure proper data governance and compliance with privacy regulations to protect sensitive information.

Data Cleaning:

Perform initial data cleaning to address missing values, inconsistencies, and outliers in the dataset.

Utilize techniques such as imputation, filtering, and outlier detection to enhance the quality of the data.

Verify data integrity and consistency to prevent errors or biases in subsequent analysis.

Data Preprocessing:

Transform raw data into a format suitable for machine learning algorithms by encoding categorical variables, scaling numerical features, and handling text or image data appropriately.

Apply techniques such as one-hot encoding, feature scaling, and text tokenization to preprocess the data effectively.

Explore dimensionality reduction methods such as principal component analysis (PCA) or feature selection to reduce the complexity of the dataset while preserving relevant information.

Exploratory Data Analysis (EDA):

Conduct exploratory data analysis to gain insights into the dataset's characteristics, distributions, and correlations.

Visualize key features and relationships using statistical plots, histograms, scatter plots, and heatmaps.

Identify patterns, trends, and anomalies in the data that may inform subsequent modeling decisions.

Data Splitting:

Divide the preprocessed dataset into training, validation, and test sets to facilitate model training, evaluation, and validation.

Ensure proper stratification and randomization to maintain the representative nature of each dataset split.

Determine appropriate proportions for each dataset split based on the specific requirements of the machine learning task. By diligently executing these steps in the first module, the project lays a solid foundation for subsequent machine learning modeling and analysis. The data collection and preprocessing module play a critical role in ensuring the quality, integrity, and suitability of the data for training and evaluating machine learning algorithms.

7.2 Module 2: ML and DL Training.

This segment encompasses a myriad of technical components including algorithm selection, feature engineering, model training, and evaluation. For the KDD dataset, a variety of ML algorithms like XGBoost, Random Forest are contemplated for intrusion detection, while for the Malware dataset, DL models like Deep Neural Network (DNN) are considered for malware classification alongside traditional ML algorithms. Feature engineering becomes pivotal during this phase to extract pertinent features from the datasets, ensuring they encapsulate meaningful information for the classification tasks. Techniques such as dimensionality reduction, feature scaling, and transformation are applied to refine the feature set, enhancing its predictive efficacy. The subsequent step involves training ML models like XGBoost and Random Forest on the KDD dataset utilizing the Python scikit-learn library. Hyperparameters such as learning rate, maximum depth, and number of estimators are fine-tuned to optimize model performance, while cross-validation techniques like k-fold cross-validation are employed to assess model generalization and mitigate overfitting. Meanwhile, deep learning models are developed using frameworks like TensorFlow or PyTorch for training on the Malware dataset. This involves designing neural network architectures, incorporating techniques such as dropout regularization and batch normalization to enhance model robustness, and defining hyperparameters like activation functions and optimization algorithms. Model evaluation ensues, wherein the performance of trained models is scrutinized using evaluation metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC). Comparative analysis between different algorithms and models aids in identifying the most effective approach for each dataset, while insights gleaned from model predictions and misclassifications inform potential areas for further refinement. By meticulously executing these technical steps, the project endeavors to develop resilient ML and DL models adept at accurately classifying network intrusions and malware instances, thereby bolstering cybersecurity defenses.

7.3 GAN Module

In the final module of the project, attention is directed towards the implementation of Generative Adversarial Networks (GANs) to generate synthetic attack data for both the KDD and malware datasets. This module plays a crucial role in augmenting the available data and enhancing the robustness of the trained machine learning (ML) and deep learning (DL) models by introducing synthetic samples for testing and evaluation purposes.

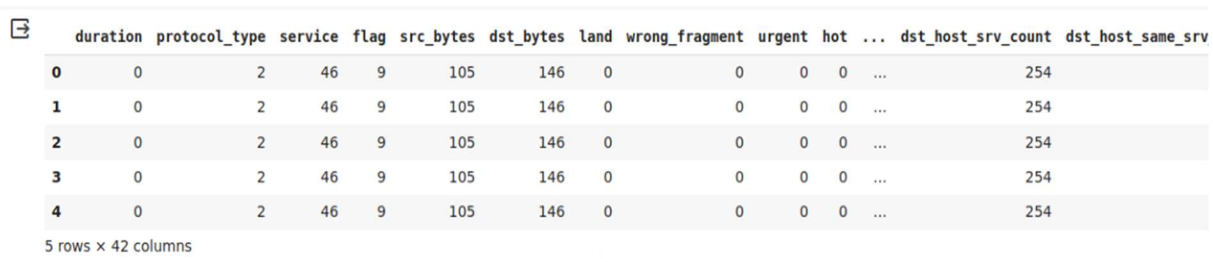
The implementation of Generative Adversarial Networks (GANs) to produce synthetic attack data for both the KDD and malware datasets, thereby augmenting the available data and fortifying the trained machine learning (ML) and deep learning (DL) models for robust testing and evaluation. This pivotal module unfolds through a series of meticulously orchestrated technical steps, each crucial in leveraging the power of GANs to generate synthetic attack samples for comprehensive model assessment and refinement. At the outset, careful consideration is given to the selection of an appropriate GAN architecture tailored to the specific characteristics of the datasets, encompassing factors like dataset complexity, output format requirements, and computational constraints. With the architectural foundation laid, meticulous dataset preparation ensues, wherein the KDD and malware datasets undergo preprocessing to align with the chosen GAN framework and training regimen, ensuring seamless integration and optimal performance. Subsequently, the GAN training process commences, orchestrated through the deployment of leading-edge deep learning frameworks such as TensorFlow or PyTorch, meticulously configured to optimize training stability and efficiency. Here, the fine-tuning of GAN hyperparameters, including learning rate, batch size, and optimization algorithm, assumes paramount importance in steering the training process towards convergence and facilitating the generation of high-quality synthetic attack samples. Throughout the training regimen, sophisticated techniques such as mini-batch discrimination, gradient penalty, and spectral normalization are harnessed to bolster training stability and enhance the diversity and realism of the generated samples. With the GAN models effectively trained, the focus shifts towards synthetic data generation, where the GANs' prowess is harnessed to produce synthetic attack samples representative of real-world scenarios. Through meticulous monitoring of the training process and judicious evaluation of generated samples using metrics such as Inception Score (IS) or Fréchet Inception Distance (FID), the quality and fidelity of the synthetic data are systematically refined, ensuring its efficacy in bolstering the testing and evaluation phase. As the generated synthetic attack data takes shape, the stage is set for comprehensive testing and evaluation of the trained ML and DL models, wherein the synthetic samples are seamlessly integrated into the evaluation pipeline alongside real-world data. Leveraging the synthesized attack data, the performance of the models is meticulously scrutinized across various metrics, encompassing accuracy, precision, recall, and F1-score, enabling a comprehensive assessment of model generalization and robustness. Through iterative refinement cycles, insights gleaned from model performance on synthetic data are meticulously assimilated to refine and enhance the efficacy of the ML and DL models, thereby culminating in a robust and resilient cybersecurity defense framework poised to tackle emerging threats with unwavering efficacy. By executing these technical steps in the GAN module, the project aims to leverage synthetic attack data generated by GANs to enhance the testing and evaluation process of the trained ML and DL models. This approach not only augments the available data but also provides a means to assess model performance under diverse attack scenarios, thereby strengthening cybersecurity defenses and mitigating the risk of false negatives in intrusion detection and malware classification tasks.

Chapter 8
IMPLEMENTATION

8.1 Data Preparation.

8.1.1 Dataset - 1: NSL KDD Dataset

- Train Test split → 70:30
- Final dataset: 217720 samples for training (X_train).
- 93309 samples for testing (X_test).
- 41 features and 1 label. 2 classes (attack:1, normal: 0).
- 38 Numerical and 3 Categorical and one label in the data.



	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv
0	0	2	46	9	105	146	0	0	0	0	...	254	
1	0	2	46	9	105	146	0	0	0	0	...	254	
2	0	2	46	9	105	146	0	0	0	0	...	254	
3	0	2	46	9	105	146	0	0	0	0	...	254	
4	0	2	46	9	105	146	0	0	0	0	...	254	

5 rows × 42 columns

Figure 8.1: NSL KDD Dataset

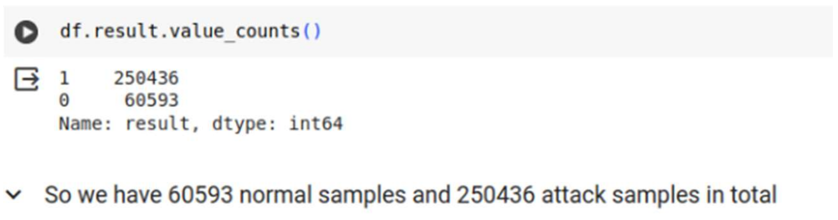
1. Class Imbalance

The types of attacks in this dataset are distributed unevenly. In order to remove class imbalance mainly two approaches are chosen:

Approach 1: delete minority classes (data diversity loss)

Approach 2: combine attack classes as one (Binary classification)

So as to resolve this concern we convert the attack classes into 1 and normal classes into 1.



```
df.result.value_counts()
```

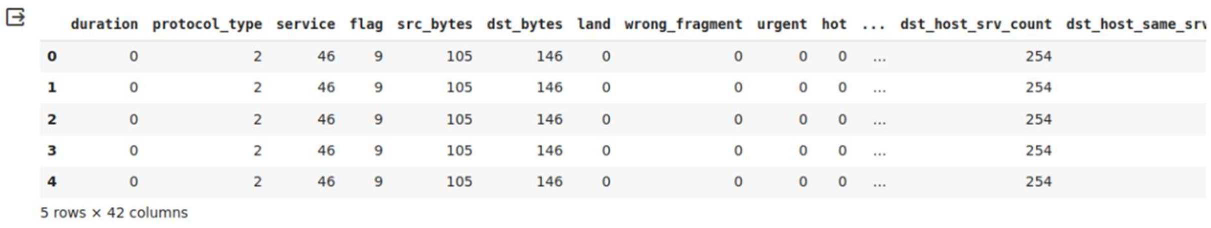
```
1    250436
0     60593
Name: result, dtype: int64
```

So we have 60593 normal samples and 250436 attack samples in total

Figure 8.2: Dataset after conversion

2. Label Encoding

Label encoding is a technique used in machine learning to convert categorical variables (data with labels or names) into numerical representations. This is necessary because most machine learning algorithms can only work with numerical data. In this dataset the protocol type, service, and flag are categorical feature so they must be converted to numerical values. Sk-learn label encoder is used. It will assign numbers from 0 to n in alphabetical order in each column entries.



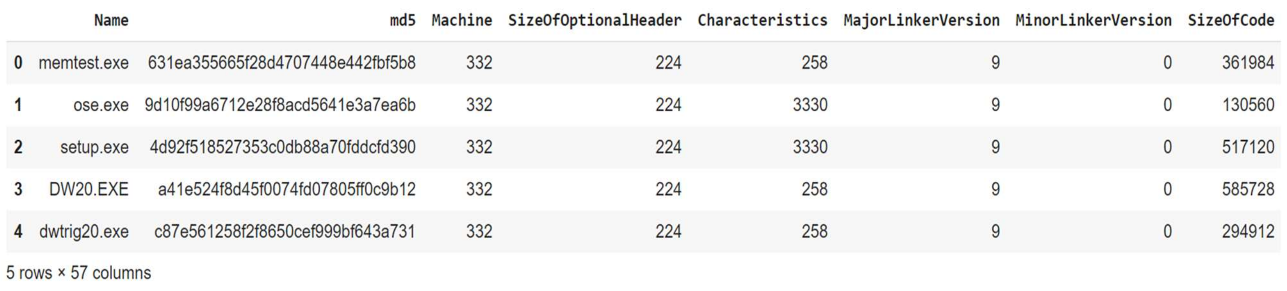
	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv
0	0	2	46	9	105	146	0	0	0	0	...	254	
1	0	2	46	9	105	146	0	0	0	0	...	254	
2	0	2	46	9	105	146	0	0	0	0	...	254	
3	0	2	46	9	105	146	0	0	0	0	...	254	
4	0	2	46	9	105	146	0	0	0	0	...	254	

5 rows × 42 columns

Figure 8.3: Dataset after Label encoding

8.1.2 Dataset - 2: MALWARE Dataset

- Dataset collected from https://www.kaggle.com/datasets/dscclass/malware
- Basic Data analysis, and preprocessing done.
- Total number of samples = 138047
- Number of columns = 57



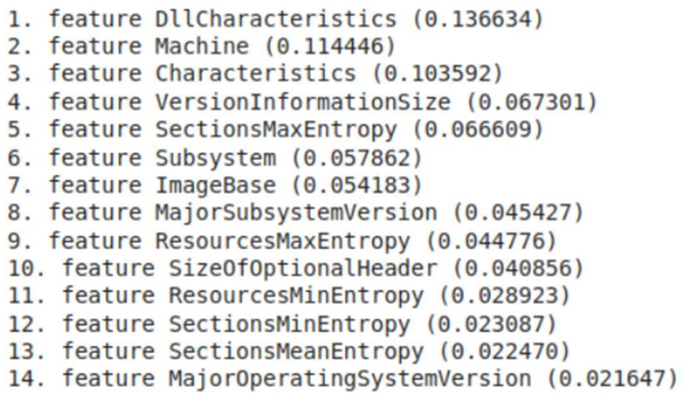
	Name	md5	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	SizeOfCode
0	memtest.exe	631ea355665f28d4707448e442fbf5b8	332	224	258	9	0	361984
1	ose.exe	9d10f99a6712e28f8acd5641e3a7ea6b	332	224	3330	9	0	130560
2	setup.exe	4d92f518527353c0db88a70fddcfd390	332	224	3330	9	0	517120
3	DW20.EXE	a41e524f8d45f0074fd07805ff0c9b12	332	224	258	9	0	585728
4	dwtrig20.exe	c87e561258f2f8650cef999bf643a731	332	224	258	9	0	294912

5 rows × 57 columns

Figure 8.4: MALWARE Dataset

1. Feature Selection

Feature selection is the process of identifying and selecting the most significant features from the dataset. This dataset have 57 features but we do not need all of them for training. Hence we perform Feature Selection in order to drop the less important features. In order to perform that we use SelectFromModel method. We train a model (Decision tree) using the all the 50 features of the dataset. After training, find the important feature learnt in order and we choose best n from them. We finally get 14 best features out of 57 based on their importance score.



```
1. feature DllCharacteristics (0.136634)
2. feature Machine (0.114446)
3. feature Characteristics (0.103592)
4. feature VersionInformationSize (0.067301)
5. feature SectionsMaxEntropy (0.066609)
6. feature Subsystem (0.057862)
7. feature ImageBase (0.054183)
8. feature MajorSubsystemVersion (0.045427)
9. feature ResourcesMaxEntropy (0.044776)
10. feature SizeOfOptionalHeader (0.040856)
11. feature ResourcesMinEntropy (0.028923)
12. feature SectionsMinEntropy (0.023087)
13. feature SectionsMeanEntropy (0.022470)
14. feature MajorOperatingSystemVersion (0.021647)
```

Figure 8.5 : Dataset after Feature Selection

8.2 Model development on dataset 1

Random forest we trained:

- Number of trees = 3
- Training data = 217720.
- Testing data = 93309.

XGboost forest we trained:

- Number of trees = 125
- Training data = 217720.
- Testing data = 93309.
-

DNN Training :

- Training data = 217720.
- Testing data = 93309.
- Input layer = 41 neurons.
- Layer 2 = 1024 neurons.
- Layer 3 = 512 neurons.
- Layer 4 = 256 neurons.
- Layer 5 = 128 neurons.
- Output layer = 2 neuron.
- Number of epochs trained = 10

8.3 Model development on dataset 2

Random forest we trained:

- Number of trees = 3
- Training data = 98013.
- Testing data = 40034.

XGboost forest we trained:

- Number of trees = 125
- Training data = 98013.
- Testing data = 40034.

DNN Training:

- Training data = 98013.
- Testing data = 40034.
- Input layer = 41 neurons.
- Layer 2 = 1024 neurons.
- Layer 3 = 512 neurons.
- Layer 4 = 256 neurons.
- Layer 5 = 128 neurons.
- Output layer = 2 neuron.
- Number of epochs trained = 10

8.4 GAN Training for Dataset 1

- Number of epochs trained = 200
- Time taken for training the final GAN model = 2.5
- Hrs. on Google Colab GPU paid version
- Data dimension = 41 (columns/features of the data)

8.5 GAN Training for Dataset 2

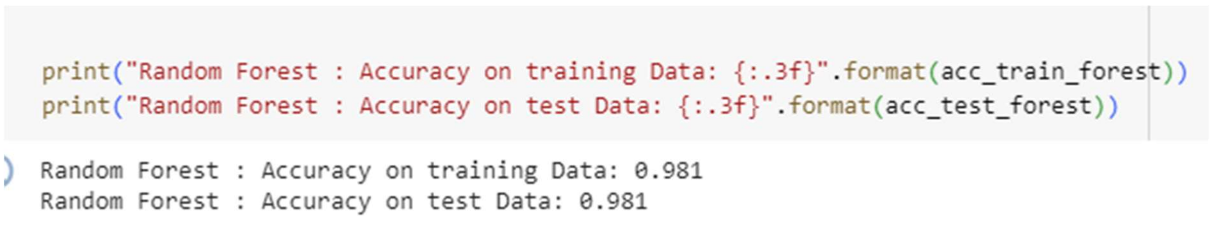
- Number of epochs trained = 200
- Time taken for training the final GAN model = 25 Mins on Google Colab GPU paid version
- Data dimension = 14 (columns/features of the data)

Chapter 9
RESULT & ANALYSIS

9.1 Results Of Dataset 1

1. Random Forest Result

- Training accuracy = 98.1%
- Testing accuracy = 98.1%
- Precision = 98.79 %
- Recall = 98.86%
- F1 score = 98.82%



```
print("Random Forest : Accuracy on training Data: {:.3f}".format(acc_train_forest))
print("Random Forest : Accuracy on test Data: {:.3f}".format(acc_test_forest))
```

```
Random Forest : Accuracy on training Data: 0.981
Random Forest : Accuracy on test Data: 0.981
```

Figure 9.1: Random forest Result 1

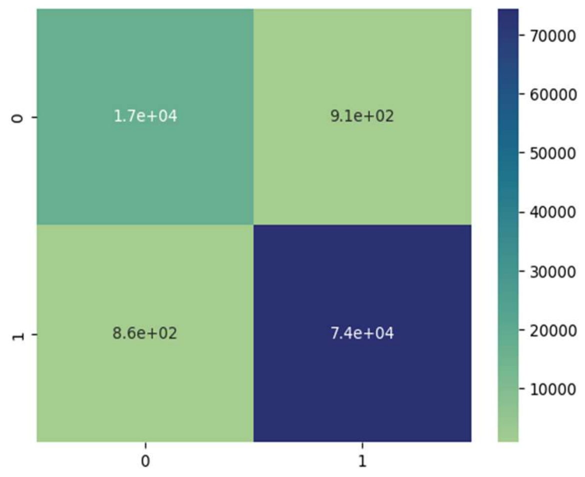
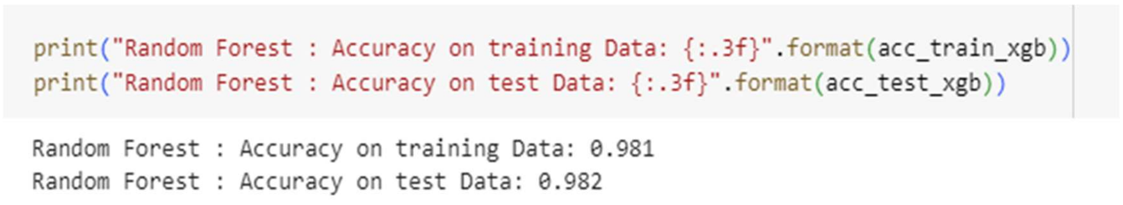


Figure 9.2: Random forest confusion matrix 1

2. XGBoost Result

- Training accuracy = 98.1%
- Testing accuracy = 98.2%
- Precision = 98.81 %
- Recall = 98.89%

- F1 score = 98.85%



```
print("Random Forest : Accuracy on training Data: {:.3f}".format(acc_train_xgb))
print("Random Forest : Accuracy on test Data: {:.3f}".format(acc_test_xgb))
```

```
Random Forest : Accuracy on training Data: 0.981
Random Forest : Accuracy on test Data: 0.982
```

Figure 9.3 : XGBoost Results 1

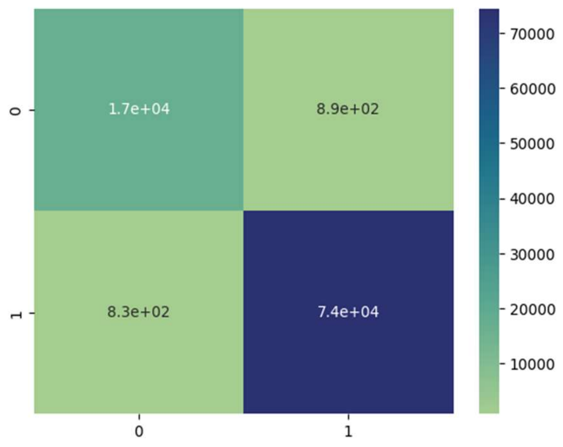
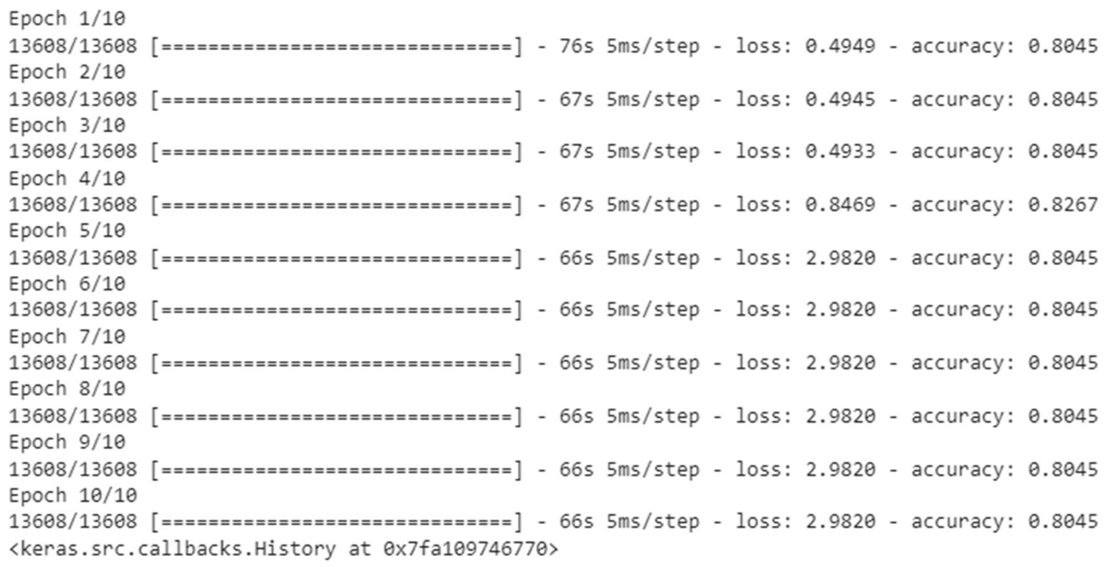


Figure 9.4: XGBoost confusion matrix 1

3. DNN Training result:

Highest accuracy achieved: 85.54%



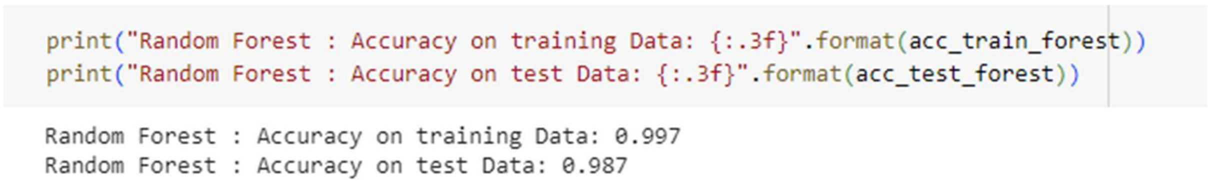
```
Epoch 1/10
13608/13608 [==============================] - 76s 5ms/step - loss: 0.4949 - accuracy: 0.8045
Epoch 2/10
13608/13608 [==============================] - 67s 5ms/step - loss: 0.4945 - accuracy: 0.8045
Epoch 3/10
13608/13608 [==============================] - 67s 5ms/step - loss: 0.4933 - accuracy: 0.8045
Epoch 4/10
13608/13608 [==============================] - 67s 5ms/step - loss: 0.8469 - accuracy: 0.8267
Epoch 5/10
13608/13608 [==============================] - 66s 5ms/step - loss: 2.9820 - accuracy: 0.8045
Epoch 6/10
13608/13608 [==============================] - 66s 5ms/step - loss: 2.9820 - accuracy: 0.8045
Epoch 7/10
13608/13608 [==============================] - 66s 5ms/step - loss: 2.9820 - accuracy: 0.8045
Epoch 8/10
13608/13608 [==============================] - 66s 5ms/step - loss: 2.9820 - accuracy: 0.8045
Epoch 9/10
13608/13608 [==============================] - 66s 5ms/step - loss: 2.9820 - accuracy: 0.8045
Epoch 10/10
13608/13608 [==============================] - 66s 5ms/step - loss: 2.9820 - accuracy: 0.8045
<keras.src.callbacks.History at 0x7fa109746770>
```

Figure 9.5 : GAN Results 1

9.2 Results of dataset 2

1. Random forest we trained:

- Training accuracy = 99.7%
- Testing accuracy = 98.7%
- Precision = 97.50 %
- Recall = 98.03%
- F1 score = 97.77%



```
print("Random Forest : Accuracy on training Data: {:.3f}".format(acc_train_forest))
print("Random Forest : Accuracy on test Data: {:.3f}".format(acc_test_forest))
```

```
Random Forest : Accuracy on training Data: 0.997
Random Forest : Accuracy on test Data: 0.987
```

Figure 9.6 : Random Forest Results 2

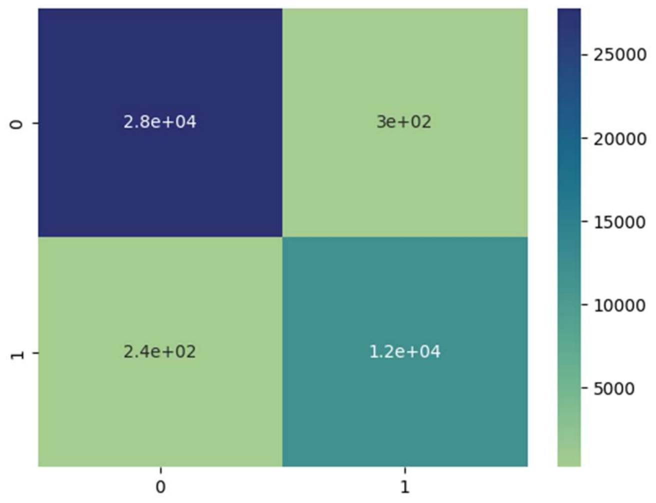
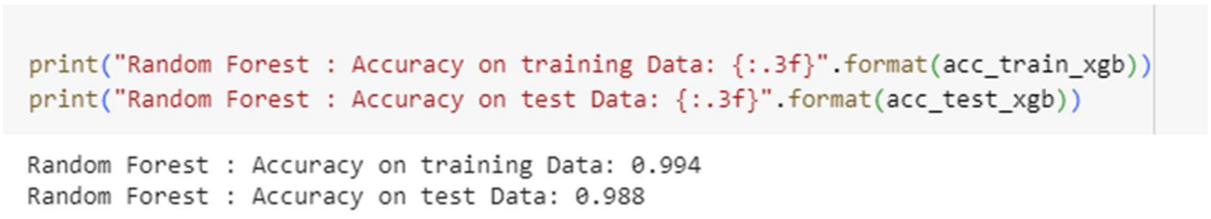


Figure 9.7: Random Forest confusion metrix 2

2. XGboost forest we trained:

- Training accuracy = 99.4%
- Testing accuracy = 98.8%
- Precision = 97.94 %
- Recall = 98.04%
- F1 score = 97.99%



```
print("Random Forest : Accuracy on training Data: {:.3f}".format(acc_train_xgb))
print("Random Forest : Accuracy on test Data: {:.3f}".format(acc_test_xgb))
```

```
Random Forest : Accuracy on training Data: 0.994
Random Forest : Accuracy on test Data: 0.988
```

Figure 9.8 : XGBoost Results 2

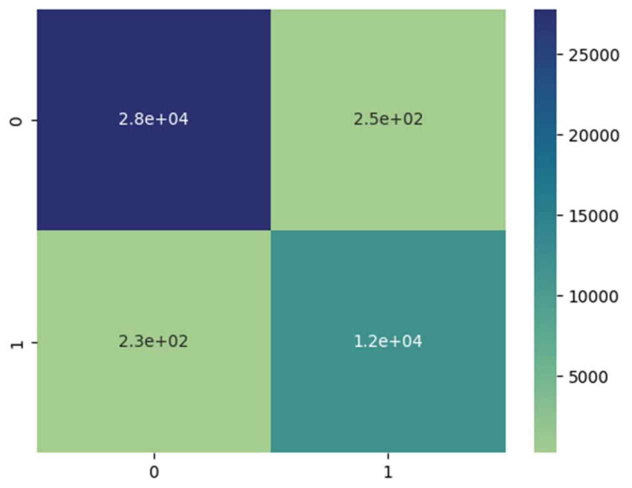
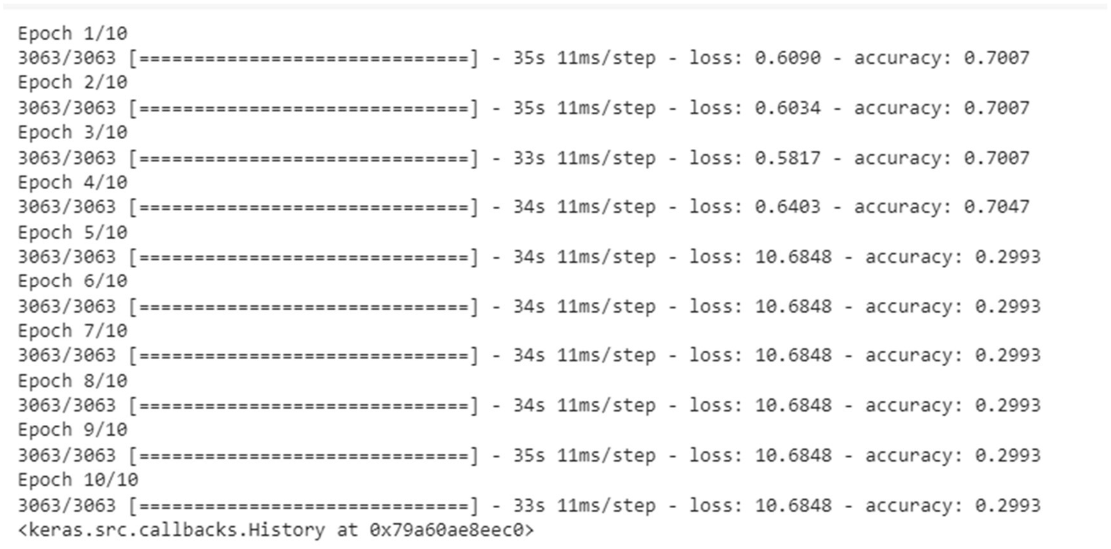


Figure 9.9: XGboost confusion matrix 2

3. DNN Training result:

- Highest accuracy achieved = 70.07%



```
Epoch 1/10
3063/3063 [==============================] - 35s 11ms/step - loss: 0.6090 - accuracy: 0.7007
Epoch 2/10
3063/3063 [==============================] - 35s 11ms/step - loss: 0.6034 - accuracy: 0.7007
Epoch 3/10
3063/3063 [==============================] - 33s 11ms/step - loss: 0.5817 - accuracy: 0.7007
Epoch 4/10
3063/3063 [==============================] - 34s 11ms/step - loss: 0.6403 - accuracy: 0.7047
Epoch 5/10
3063/3063 [==============================] - 34s 11ms/step - loss: 10.6848 - accuracy: 0.2993
Epoch 6/10
3063/3063 [==============================] - 34s 11ms/step - loss: 10.6848 - accuracy: 0.2993
Epoch 7/10
3063/3063 [==============================] - 34s 11ms/step - loss: 10.6848 - accuracy: 0.2993
Epoch 8/10
3063/3063 [==============================] - 34s 11ms/step - loss: 10.6848 - accuracy: 0.2993
Epoch 9/10
3063/3063 [==============================] - 35s 11ms/step - loss: 10.6848 - accuracy: 0.2993
Epoch 10/10
3063/3063 [==============================] - 33s 11ms/step - loss: 10.6848 - accuracy: 0.2993
<keras.src.callbacks.History at 0x79a60ae8eec0>
```

Figure 9.10 : GAN Results 2

8.4 Analysis

Dataset 1:

	Model Name	**Accuracy**
1	Random Forest	98.1
2	XGBoost	98.2
3	DNN	84.5

Figure 9.11: Analysis of KDD data

Dataset 2 :

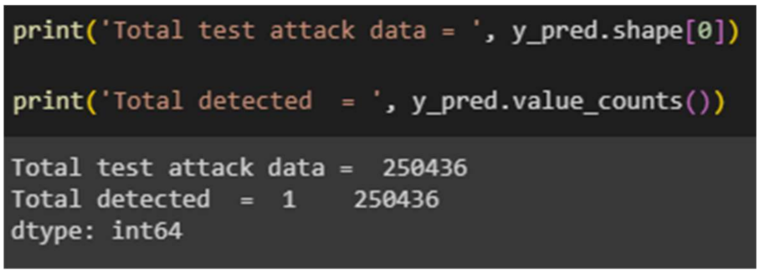
	Model Name	**Accuracy**
1	Random Forest	98.7
2	XGBoost	98.8
3	DNN	70.07

Figure 9.12: Analysis of MALWARE data

During our evaluation we concluded that XGBoost could achieve the highest accuracy in comparison to the other models we trained. Hence, we chose XGBoost for further testing.

8.5 Result of GAN 1

- Discriminator loss = 0.242
- Generator loss = 1.136



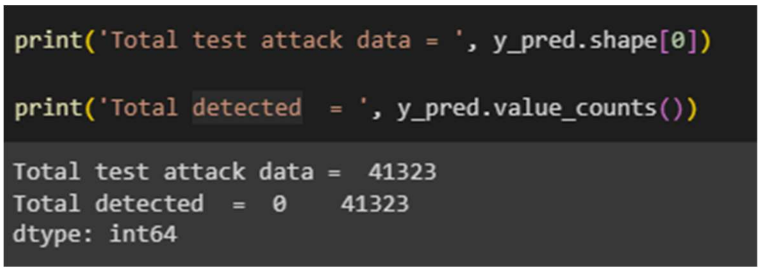
```
print('Total test attack data = ', y_pred.shape[0])

print('Total detected  = ', y_pred.value_counts())
```

```
Total test attack data =  250436
Total detected  =  1    250436
dtype: int64
```

Figure 9.13: Accurate detection of KDD data

8.6 Result of GAN 2

- Discriminator loss = 3.059
- Generator loss = 0.012



```
print('Total test attack data = ', y_pred.shape[0])

print('Total detected  = ', y_pred.value_counts())

Total test attack data =  41323
Total detected  =  0    41323
dtype: int64
```

Figure 9.14: Accurate detection of Malware data

Chapter 9
CONCLUSION

This project undertook the task of training multiple machine learning (ML) models to detect two prevalent types of cyber-attacks: web attacks and malwares. Through rigorous experimentation, it was observed that XGBoost emerged as one of the most effective classifiers among the ML models considered, surpassing even deep learning (DL) models in terms of performance. This finding underscores the robustness and versatility of XGBoost in handling complex cyber threat detection tasks. Moreover, a novel evaluation approach was adopted to assess the efficacy of the XGBoost models, which involved training a Generative Adversarial Network (GAN) to generate synthetic attack data. Remarkably, when subjected to this synthetic data, the XGBoost model demonstrated flawless performance, achieving 100% accuracy in predicting the generated attacks. This outcome highlights the adaptability and generalization capabilities of the XGBoost model, showcasing its potential for real-world deployment.

Looking ahead, there are opportunities to further enhance the effectiveness of the XGBoost model by incorporating additional attack data to augment its training dataset. This augmentation can contribute to making the model more robust and resilient against a broader spectrum of cyber threats. However, it is important to acknowledge that advancing the GAN training process to generate more diverse and sophisticated attack data poses computational challenges, requiring significant hardware resources and computational power.

REFERENCES

1. Shaukat, K., Luo, S., Chen, S. and Liu, D., 2020, October. Cyber threat detection using machine learning techniques: A performance evaluation perspective. In *2020 international conference on cyber warfare and security (ICCWS)* (pp. 1-6). IEEE.

2. Lee, J., Kim, J., Kim, I. and Han, K., 2019. Cyber threat detection based on artificial neural networks using event profiles. *Ieee Access*, *7*, pp.165607-165626.

3. Won, D.O., Jang, Y.N. and Lee, S.W., 2022. PlausMal-GAN: Plausible malware training based on generative adversarial networks for analogous zero-day malware detection. *IEEE Transactions on Emerging Topics in Computing*, *11*(1), pp.82-94.

4. Ahmed, N., Ngadi, A.B., Sharif, J.M., Hussain, S., Uddin, M., Rathore, M.S., Iqbal, J., Abdelhaq, M., Alsaqour, R., Ullah, S.S. and Zuhra, F.T., 2022. Network threat detection using machine/deep learning in sdn-based platforms: a comprehensive analysis of state-of-the-art solutions, discussion, challenges, and future research direction. *Sensors*, *22*(20), p.7896.

5. Ke, Q., 2021, November. Research on threat detection in cyber security based on machine learning. In *Journal of Physics: Conference Series* (Vol. 2113, No. 1, p. 012074). IOP Publishing.

6. Jullian, O., Otero, B., Rodriguez, E., Gutierrez, N., Antona, H. and Canal, R., 2023. Deep-learning based detection for cyber-attacks in iot networks: A distributed attack detection framework. *Journal of Network and Systems Management*, *31*(2), p.33.

7. Alotaibi, A. and Rassam, M.A., 2023. Adversarial machine learning attacks against intrusion detection systems: A survey on strategies and defense. *Future Internet*, *15*(2), p.62.

8. Shi, A., 2021, September. Cyber attacks detection based on generative adversarial networks. In *2021 2nd Asia Conference on Computers and Communications (ACCC)* (pp. 111-114). IEEE.

9. Hao, X., Jiang, Z., Xiao, Q., Wang, Q., Yao, Y., Liu, B. and Liu, J., 2021, May. Producing more with less: a GAN-based network attack detection approach for imbalanced data. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (pp. 384-390). IEEE.

10. Xiong, Y., Xu, F. and Zhong, S., 2020, June. Detecting GAN-based privacy attack in distributed learning. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)* (pp. 1-6). IEEE.

APPENDIX

```
#  XGBoost Classifier Model
from xgboost import XGBClassifier

# instantiate the model
xgb = XGBClassifier()

# Convert the label into XGBoost required format
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_train = le.fit_transform(y_train)

# fit/Train the model
xgb.fit(X_train,y_train)

# Ask the trained model to predict X_train and X_test
y_train_xgb = xgb.predict(X_train)
y_test_xgb = xgb.predict(X_test)

# Calculate the training accuracy and testing accuracy
acc_train_xgb = metrics.accuracy_score(y_train,y_train_xgb)
acc_test_xgb = metrics.accuracy_score(y_test,y_test_xgb)
print("Random Forest : Accuracy on training Data: {:.3f}".format(acc_train_xgb))
print("Random Forest : Accuracy on test Data: {:.3f}".format(acc_test_xgb))

# Print and see the confusion matrix
# print the confusion matrix
cf_matrix = confusion_matrix(y_test, y_test_xgb)
sns.heatmap(cf_matrix, annot=True, cmap="crest")

cf_matrix
print(accuracy_score(y_test, y_test_xgb))
print(precision_score(y_test, y_test_xgb))
print(recall_score(y_test, y_test_xgb))
print(f1_score(y_test, y_test_xgb))

# Save the model
filename = "XGBoost_model.pickle"
# save model
pickle.dump(xgb, open(filename, "wb"))

# Import librraies

import pandas as pd # For dealing with datasets and dataframes
import matplotlib.pyplot as plt # For plotting and visualizations
import numpy as np # For numerical operations
import seaborn as sns # For advanced plotting and visualization
from sklearn.preprocessing import LabelEncoder # The label encoder
from sklearn import metrics   # For model evaluation metrics
from sklearn.model_selection import train_test_split # For train test split of the dataset
import pickle # To save the trained models in pickle format
# To calculate the performance evaluation metrics
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score,
confusion_matrix # Evaluation metrics

# Read the KDD dataset  collected from http://www.kaggle.com
df = pd.read_csv('Datasets/kdd_dataset.csv')

# Remove nwanted columns
df = df.drop(['Unnamed: 0'], axis=1)

lenc=LabelEncoder()   # Get the label encoder
df.protocol_type = lenc.fit_transform(df.protocol_type) # Label encode the "Protocol type column"
print(lenc.classes_)  # Before label encoding
print(df.protocol_type.unique())  # After label encoding
```

```
# Next "service " column
lenc=LabelEncoder()   # Get the label encoder
df.service = lenc.fit_transform(df.service) # Label encode the "Protocol type column"
print(lenc.classes_)  # Before label encoding
print(df.service.unique())  # After label encoding

# Next "flag " column
lenc=LabelEncoder()   # Get the label encoder
df.flag = lenc.fit_transform(df.flag) # Label encode the "Protocol type column"
print(lenc.classes_)  # Before label encoding
print(df.flag.unique())  # After label encoding

# Label encoding
df['result'] = df['result'].map({'normal.':0, 'snmpgetattack.':1, 'named.':1, 'xlock.':1, 'smurf.':1, 'ipsweep.':1, 'multihop.':1, 'xsnoop.':1, 'sendmail.':1, 'guess_passwd.':1, 'saint.':1, 'buffer_overflow.':1, 'portsweep.':1, 'pod.':1, 'apache2.':1, 'phf.':1, 'udpstorm.':1, 'warezmaster.':1, 'perl.':1, 'satan.':1, 'xterm.':1, 'mscan.':1, 'processtable.':1, 'ps.':1, 'nmap.':1, 'rootkit.':1, 'neptune.':1, 'loadmodule.':1, 'imap.':1, 'back.':1, 'httptunnel.':1, 'worm.':1, 'mailbomb.':1, 'ftp_write.':1, 'teardrop.':1, 'land.':1, 'sqlattack.':1, 'snmpguess.':1})

# Now check the result column
df.result.unique()

df.result.value_counts()

# Choose the attck samples only to train the GAN model to generate fake new smaples
attacks = df[df['result'] == 1]
attacks.shape

attacks = attacks.drop(['result'], axis = 1)
attacks.shape
data = attacks
data.shape
```

```
import os
import logging
import numpy as np
import pandas as pd
import tensorflow as tf

tf.get_logger().setLevel(logging.ERROR)

class Gan():                # Define the GAN model
    def __init__(self, data):
        self.data = data
        self.n_epochs = 200

    # Genereta random noise in a latent space
    def _noise(self):
        noise = np.random.normal(0, 1, self.data.shape)
        return noise

    def _generator(self):                              # The generator
        model = tf.keras.Sequential(name="Generator_model")
        model.add(tf.keras.layers.Dense(15, activation='relu',
                                kernel_initializer='he_uniform',
                                input_dim=self.data.shape[1]))
        model.add(tf.keras.layers.Dense(30, activation='relu'))
        model.add(tf.keras.layers.Dense(
            self.data.shape[1], activation='linear'))
        return model

    def _discriminator(self):                      # The descriminator
        model = tf.keras.Sequential(name="Discriminator_model")
        model.add(tf.keras.layers.Dense(25, activation='relu',
                                kernel_initializer='he_uniform',
                                input_dim=self.data.shape[1]))
        model.add(tf.keras.layers.Dense(50, activation='relu'))
        # sigmoid => real or fake
        model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
        model.compile(loss='binary_crossentropy',
                      optimizer='adam',
                      metrics=['accuracy'])

        return model

    # define the combined generator and discriminator model,
    # for updating the generator
    def _GAN(self, generator, discriminator):       # The combined GAN model
        discriminator.trainable = False
        generator.trainable = True
        model = tf.keras.Sequential(name="GAN")
        model.add(generator)
        model.add(discriminator)
        model.compile(loss='binary_crossentropy', optimizer='adam')
        return model

    # train the generator and discriminator
    def train(self, generator, discriminator, gan):      # Train function

        # determine half the size of one batch, for updating the  discriminator
        # manually enumerate epochs
        for epoch in range(self.n_epochs):

            # Train the discriminator
            generated_data = generator.predict(self._noise())
            labels = np.concatenate([np.ones(self.data.shape[0]), np.zeros(self.data.shape[0])])
            X = np.concatenate([self.data, generated_data])
            discriminator.trainable = True
            d_loss , _ = discriminator.train_on_batch(X, labels)
            # Train the generator
            noise = self._noise()
            g_loss = gan.train_on_batch(noise, np.ones(self.data.shape[0]))

            print('>%d, d1=%.3f, d2=%.3f' %(epoch+1, d_loss, g_loss))

        return generator

# Train the GAN model for 200 epochs
model = Gan(data=data)
generator = model._generator()
descriminator = model._discriminator()
gan_model = model._GAN(generator=generator, discriminator=descriminator)
trained_model = model.train(
    generator=generator, discriminator=descriminator, gan=gan_model)

# Save the model
trained_model.save('GAN_KDD_Model.h5')

# load teh saved  model to check predictions
from tensorflow.keras.models import load_model
savedModel=load_model('GAN_KDD_Model.h5')
savedModel.summary()

# Convert the genrated data into pandas df
def _df(data):
    df = pd.DataFrame(data)
    for c in range(df.shape[1]):
        mapping = {df.columns[c]: c}
        df = df.rename(columns=mapping)
    return df

# Genrate some new  data
noise = np.random.normal(0, 1, data.shape)
```

```
new_data = _df(data=savedModel.predict(noise))
new_data.shape

# Load the first GAN model
from tensorflow.keras.models import load_model
GAN2 =load_model('GAN_Malware_Model.h5')
GAN2.summary()

# Convert the genrated data into pandas df
def _df(data):
    df = pd.DataFrame(data)
    for c in range(df.shape[1]):
        mapping = {df.columns[c]: c}
        df = df.rename(columns=mapping)
    return df

# Genrate some new atatck data
noise = np.random.normal(0, 1, (41323, 14))
new_data = _df(data=GAN2.predict(noise))
new_data.shape

# Now load the attack detection model we trained for malware
import pickle
Detector = pickle.load(open("Malware_XGBoost_model.pickle", "rb"))

columns = ['Machine',
 'SizeOfOptionalHeader',
 'Characteristics',
 'MajorLinkerVersion',
 'MinorLinkerVersion',
 'SizeOfCode',
 'SizeOfInitializedData',
 'SizeOfUninitializedData',
 'AddressOfEntryPoint',
 'BaseOfCode',
 'BaseOfData',
 'ImageBase',
 'SectionAlignment',
 'FileAlignment']

len(columns)

new_data.columns = columns
new_data.head()

# Ask the model to predict/detect the generated data
y_pred = Detector .predict(new_data)

# Check how many predictions are 1
y_pred = pd.DataFrame(y_pred)  #convert into dataframe

print('Total test attack data = ', y_pred.shape[0])
print('Total detected  = ', y_pred.value_counts())
```