UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

TÍTULO: Zero-day Attacks Detection Using Machine Learning Techniques

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniera en Tecnologías de la Información

Autor:

Arianna Belen Armijos Inga

Tutor:

Ph.D. Cuenca Erick

Urcuquí, Noviembre de 2023

Autoría

Yo, **Arianna Belen Armijos Inga**, con cédula de identidad **0750167181**, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el autor del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Noviembre del 2023.

Firmado electrónicamente por:
ARIANNA BELEN
ARMIJOS INGA

Arianna Belen Armijos Inga
CI: 0750167181

Autorización de publicación

Yo, **Arianna Belen Armijos Inga**, con cédula de identidad **0750167181**, cedo a la Universidad de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Urcuquí, Noviembre del 2023.

Firmado electrónicamente por:
ARIANNA BELEN
ARMIJOS INGA

Arianna Belen Armijos Inga
CI: 0750167181

Dedication

"This work is dedicated, first and foremost, to my beloved family: my mother, my brother, my father, and my dear Aunt Nancy, who has always been like a second mother to me. I also extend my heartfelt dedication to my cherished grandmothers and all those who have walked alongside me on this arduous journey. Secondly, I dedicate this to myself, acknowledging my resilience on this challenging and sacrifice-filled path."

Acknowledgments

A heartfelt and special gratitude extends to my graduation project advisor, who, since our first meeting, has been a steadfast supporter, encouraging my growth as a professional through their dedicated guidance and boundless patience. My deepest thanks to my family for their unwavering support. I am immensely grateful to the Computer Science Club, a place where I not only flourished but also gained invaluable knowledge and experience.

I extend my appreciation to all those who have left their mark on this journey. I'd like to express my gratitude to Wacho, who, since our acquaintance, has instilled in me the values of discipline, perseverance, and becoming a better student.

I offer my thanks to José, who has played a significant role in my life in recent years, offering unconditional support during both challenging and joyous moments.

Finally, I wish to express my deep gratitude to everyone who believed in me throughout my career, from childhood to youth, as I have always harbored a deep passion for everything related to the world of computer science.

Abstract

Currently, we live in a world where technology is ubiquitous, and it has become an integral part of our daily lives. Technology continually advances to become more robust and to provide greater privileges to those who consume its services. However, as technology continues to grow, the Internet expands, and so do the vulnerabilities and cyberattacks to which we, as users, may be exposed within a network. It's a network where our data and personal information can be discovered by malicious users.

Among these attacks, there are zero-day attacks, which fall within the realm of cybersecurity. They are the most dangerous threats one can encounter within a network or software today. This is because the security measures in place to prevent such cyberattacks lack knowledge or records of them. Zero-day attacks rely on the injection of malicious code, exploiting vulnerabilities that have not yet been discovered by users or the creators of the said software or network.

The objective of this graduation project is to propose a new algorithm that, by employing various machine learning techniques, can detect zero-day attacks through anomaly recognition within a dataset containing benign and malicious network traffic.

Keywords: Zero-day attack, Cibersecurity, IDS, Isolation Forest, Machine Learning, UNSW-NB15.

Resumen

Actualmente, vivimos en un mundo donde la tecnología está en todas partes y es parte de nuestro día a día, cada vez avanza más para ser más robusta y darnos mayores privilegios a quienes consumimos estos servicios, sin embargo, mientras más crece la tecnología, La red de Internet se expande y mayores son las vulnerabilidades y ciberataques a los que como usuarios podemos estar expuestos dentro de una red, una red en la que nuestros datos e información personal pueden ser descubiertos por cualquier usuario malintencionado.

Dentro de estos ataques existen los ataques de día cero, estos están dentro del área de la ciberseguridad, son los más peligrosos que se pueden encontrar dentro de una red o software hoy en día, debido a que la seguridad encargada de prevenir este tipo de ciberataques no tiene conocimiento. o registro de aquellos. Los ataques de día cero se basan en la inyección de código malicioso gracias al conocimiento de vulnerabilidades aún no descubiertas por los usuarios o por los creadores de dicho software o red.

El objetivo de este proyecto de graduación es proponer un nuevo algoritmo en el que, utilizando diferentes técnicas de aprendizaje automático, sea posible crear un algoritmo capaz de detectar ataques de día cero bajo el análisis de reconocimiento de anomalías dentro de un conjunto de datos con tráfico de red benigno y maligno.

Palabras Clave: Ataques de día cero, Ciberseguridad, IDS, Isolation Forest, Aprendizaje automático, UNSW-NB15.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The rise of technology has enabled businesses and organizations to become more efficient and connected than ever before [4]. However, this increased connectivity also comes with a higher risk of cyberattacks, which can compromise sensitive data and disrupt normal operations [5]. As such, it is crucial for organizations to have effective prevention techniques in place to protect against these attacks [6]. Two commonly used methods for preventing cyberattacks are Intrusion Prevention Systems (IPS) and Intrusion Detection Systems (IDS), but these methods are not always effective [5]. As such, there is a growing interest in the use of machine learning techniques for improving cybersecurity [4].

The aim of this thesis is to explore the potential of machine learning algorithms for preventing cyberattacks, particularly in the context of IDS systems. The thesis will begin by providing an overview of the different types of cyberattacks and the methods used to prevent them. It will then discuss the limitations of traditional prevention techniques, such as IPS and IDS, and the potential of machine learning algorithms for improving these techniques.

One of the main challenges of using machine learning for cybersecurity is the need for high-quality data to train the algorithms [4]. The thesis will explore different sources of data, such as network traffic data and historical attack data, and discuss the best methods for collecting and cleaning this data. It will also examine different machine learning algorithms, such as decision trees, neural networks, and clustering algorithms, and discuss their strengths and weaknesses for preventing cyberattacks.

In conclusion, this thesis will provide valuable insights into the potential of machine learning algorithms for preventing cyberattacks. It will help to identify the best methods for collecting and cleaning data and highlight the most effective machine learning algorithms for preventing different types of cyberattacks. By developing a better understanding of these techniques, organizations can improve their cybersecurity strategies and better protect against the growing threat of cyberattacks.

1.1 Problem statement

As the use of the Internet and advanced technology expands, cybersecurity risks have significantly increased. The growing utilization of artificial intelligence and other technolo-

gies further heightens the risk of malicious attacks, making it crucial to create effective measures for the prevention of zero-day attacks. These attacks pose a looming threat to information security worldwide due to their ability to exploit unknown vulnerabilities in software and online systems. Many organizations employ conventional security solutions to protect themselves against known malware, but these approaches can be ineffective against unidentified threats [7]. As cybercriminals seek more sophisticated ways to evade defenses, the industry must find ways to anticipate and prevent such attacks [8]. The rise in zero-day attacks is jeopardizing corporate networks worldwide, and today the need for safeguarding critical data online is more important than ever. Preventing or mitigating these increasing economic, legal, and privacy risks presents significant security challenges for decision-makers [9]. Moreover, according to [10] more than three billion zero-day attacks were reported in 2016, and the volume and intensity of the zero-day attacks were substantially greater than previously. However, the new generation of malware has become more ambitious and is targeting the banks themselves, sometimes trying to take millions of dollars in one attack. For that reason, the detection of zero-day attacks has become the highest priority.

1.2 Objectives

1.2.1 General Objective

The overall objective of this final degree project is to pioneer an innovative approach that leverages a spectrum of machine learning techniques, and statistical and qualitative analysis to develop an algorithm capable of detecting zero-day attacks by analyzing and discussing anomalous behavior within existing network traffic.

1.2.2 Specific Objectives

- Exploring cutting-edge machine learning algorithms and anomaly detection methodologies, evaluating their suitability for predicting zero-day attacks based on identified behavioral patterns.
- Developing a robust algorithm that combines machine learning techniques so that it can detect anomalies within network traffic.
- Validating the effectiveness of the proposed algorithm through rigorous, qualitative, detailed, and graphic analysis of each of the properties that are part of an example of network traffic.

1.3 Contributions

This work proposes a zero-day attack detection algorithm, through the analysis and detection of anomalies within network traffic, whose data represents benign and malignant cases. For this algorithm, machine learning techniques were used, especially unsupervised

learning based on anomalies, statistical analysis, and data dimensionality reduction techniques. The algorithm concludes by training the Isolation Forest model with the data previously processed with the respective statistical analysis and after having gone through both linear and non-linear dimensionality reduction techniques. Furthermore, a part of this work was carried out as a review article under the name "Zero-day attacks: a review of the methods used based on intrusion detection and prevention systems" and was sent to the IEEE Colombia-Caribbean Conference - C3, which was accepted to be published within the IEEE magazine.

1.4 Document Organization

This work is made up of six main sections. These are detailed below:

- Section 1: Introduction. This section begins with a concise overview of the topic, followed by an elaboration of the problem statement central to the development of this project and its associated contributions. Also, outlines the objectives and provides an overview of the structure of this document.
- Section 2: This section establishes the theoretical bases on which this project is built and provides definitions to facilitate a comprehensive understanding of the topic.
- Section 3: State of the Art. This chapter explores a series of works related to the topic of cybersecurity and the detection of various attacks, including zero-day attacks. Its techniques, methodology, and the results obtained are analyzed.
- Section 4: Methodology. This section presents the methodology used in this work to create the proposed algorithm. The process involves the development of statistical, qualitative, graphical analysis and the machine learning model.
- Section 5: Results and Discussion. This section presents the results obtained from this work, in addition to comparing and discussing several examples.
- Section 6: Conclusions and Future Work. This section summarizes the achievements of this work and provides future ideas that can be developed, using this research as a basis.

Chapter 2

Theoretical Framework

This section will explain the fundamentals that revolve around the theme of this thesis, from what security is its types to the existing and most common types of Machine Learning. The different types of cyber attacks will be detailed, examples will be given, and how they have impacted the world and its economy, especially zero-day attacks, will be mentioned, whose attacks are increasingly dangerous within the network. In addition, it will explain how, currently, the different researchers or experts in the area use the different tools available to create detection or prevention methods for the different types of attacks.

2.1 Security Types

Security is a critical concern permeating every domain, and the digital realm is no exception. Various types of security have emerged within this vast landscape to shield information systems against threats and prevent unauthorized access [11]. Each of the types of security mentioned in the paragraph is detailed below:

- **Information Security:** is about safeguarding information and information systems against unauthorized access, use, disclosure, interruption, modification, or destruction. According to [12], information security is a critical issue for organizations as the value of information continues to increase and threats to information security become more sophisticated. Information security measures include access controls, data encryption, backup and recovery procedures, and security awareness training.
- **Computer Security:** This field, sometimes called cybersecurity or IT security, is focused on safeguarding computer systems and networks against unauthorized access, use, disclosure, disruption, alteration, or destruction. As [13] point out, computer security plays a vital role in our interconnected society, as cyber threats constantly evolve, presenting substantial risks to individuals, organizations, and even governments. To ensure computer security, various measures are employed, such as the implementation of firewalls, antivirus software, intrusion detection systems, and the regular performance of security audits, among other practices.
- **Cibersecurity:** is the set of practices, measures, and technologies to protect computer systems and networks from cyber threats. According to [14], It implies the

implementation of strategies and controls designed to safeguard the confidentiality, integrity, and availability of the information stored and transmitted through digital systems and networks. This includes protection against hacks, malware, data theft, unauthorized intrusions, and other malicious activity online. Cybersecurity has become increasingly important as reliance on technology and global interconnection continue to grow, and a robust defense is required to protect digital assets and user privacy.

2.1.1 Confidentiality, Integrity and Availability (CIA)

All the types of security and the CIA triad are closely intertwined concepts within the realm of information security. The CIA triad is a fundamental model used to ensure the comprehensive security of information, and it is highly relevant to cybersecurity. This relationship can be seen in Fig. 2.1. By leveraging the principles of the CIA triad, organizations can identify and prioritize their security objectives, enabling them to implement suitable measures to safeguard their valuable information assets [15].

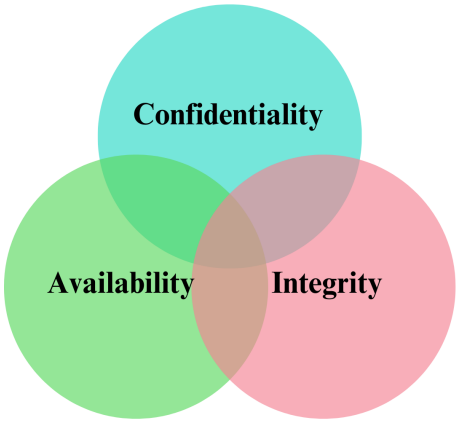


Figure 2.1: CIA Triad, model used to develop cybersecurity policies and systems in an organization

The CIA triad, an acronym for confidentiality, integrity, and availability, encompasses critical components of information security. Confidentiality involves protecting sensitive information from unauthorized access or disclosure. Integrity focuses on maintaining the accuracy, consistency, and reliability of information. Availability ensures authorized users can access information when required [16].

In cybersecurity, the CIA triad is pivotal in helping organizations safeguard their sensitive information and systems against a myriad of cyber threats and attacks. For instance, to uphold confidentiality, organizations may employ robust encryption algorithms, access controls, and data classification techniques. Upholding the integrity of information involves using measures such as digital signatures, data backups, and audit trails. Ensuring availability entails implementing redundancy mechanisms, disaster recovery plans, and load balancing techniques [17].

In essence, the CIA triad serves as a cornerstone of cybersecurity, guiding organizations to prioritize protecting their information by implementing appropriate security measures aligned with confidentiality, integrity, and availability. By leveraging the insights the CIA triad provides, organizations can establish a robust security posture to mitigate risks and safeguard their critical information assets effectively [18].

2.2 Cibersecurity

Cybersecurity is essential in today's digital era, tasked with protecting systems, networks, and data from a wide range of malicious attacks. It involves implementing comprehensive measures and practices to safeguard the confidentiality, integrity, and availability of information in cyberspace. With the increasing sophistication of cyber threats, cybersecurity plays a crucial role in ensuring the resilience of digital infrastructure and the preservation of trust in our interconnected world [19].

First and foremost, cybersecurity encompasses a multidimensional approach to defending the digital landscape against many threats. This includes protecting the intricate infrastructure web from insidious malware, deceptive phishing attacks, and intrusive hacking attempts. Cybersecurity efforts focus on deploying advanced systems for detecting and preventing intrusions while effectively managing vulnerabilities and implementing timely security patches. The primary objective is to maintain the robustness and integrity of systems while mitigating the risk of unauthorized access and data breaches [20].

Moreover, in our data-driven society, cybersecurity plays a vital role in preserving the privacy of individuals and safeguarding their personal data. This involves establishing clear and comprehensive privacy policies and guidelines, implementing robust encryption mechanisms to protect sensitive information, and fostering a culture of cyber awareness among users. Educating individuals about best practices for online security empowers them to make informed decisions, fortify their digital footprint, and guard against privacy infringements and identity theft [21]. This wants us to understand that the primary goal of cybersecurity is to maintain the confidentiality, integrity, and availability (CIA) of information. This involves implementing various security measures, including firewalls, encryption, access controls, and other technologies, to prevent unauthorized access, data breaches, and cyber-attacks.

In addition to preventive measures, cybersecurity emphasizes the importance of incident preparedness and swift response. Organizations and individuals must proactively develop incident response plans, conduct thorough penetration testing and realistic attack simulations, and implement real-time monitoring and early detection systems. By promptly identifying and containing cyber threats, swift responses can minimize the impact of attacks, limit the scope of potential damages, and expedite the recovery of affected systems [22].

Finally, the collaborative nature of cybersecurity is critical for its effectiveness. The field thrives on the collective efforts of cybersecurity professionals and organizations who work hand in hand to share knowledge, expertise, and threat intelligence [23]. By fostering open communication channels and encouraging information sharing, stakeholders can collectively identify emerging threats, develop innovative defense strategies, and effectively respond to evolving cyber challenges. Furthermore, raising public awareness and provid-

ing comprehensive cybersecurity education is instrumental in cultivating a cyber-literate society that prioritizes digital security in all aspects of life [24].

2.2.1 Cibersecurity Types

Today, cybersecurity must be considered an indispensable aspect of our lives. As we navigate a world driven by ever-evolving technology, protecting our data, systems, and networks from malicious actors has become paramount [25]. Cybersecurity encompasses a broad field of practices, strategies, and technologies carefully designed to strengthen our digital connections against threats such as unauthorized access, data breaches, and cyberattacks [26]. Within cybersecurity, several types focus on different types of protection and defense. These include network security, application security, information security, cloud security, and many others [27]. Each type is very important in reinforcing our digital infrastructure ensuring our information's confidentiality, integrity, and availability. In this context, we will delve into the different types of cybersecurity, explaining each, when, or under what circumstances they are used or categorized under that type. Those security types are explained in detail below:

- **Network Security:** is a smaller subset that falls under the larger cybersecurity umbrella and refers to preventing unauthorized users from accessing computer networks and their associated devices [28]. This implies the creation of a secure infrastructure for devices, applications, users, and applications to work safely [29]. Network security has three main objectives: to prevent unauthorized access to network resources, detect and stop ongoing cyberattacks and security breaches, and ensure that authorized users have secure access to the network resources they need when they need them [30].

- **Application Security:** is making apps more secure by finding, fixing, and enhancing the security of apps [31]. It is important for several reasons. First, finding and fixing vulnerabilities reduces security risks, and doing so helps reduce an organization's overall attack surface. Second, software vulnerabilities are common. Taking a proactive approach to application security is better than reactive security measures. Third, as enterprises move more of their data, code, and operations into the cloud, attacks against those assets can increase [32]. This type of security aims to protect software application code and data against cyber threats. You can and should apply application security during all phases of development, including design, development, and deployment. Here are several ways to promote application security throughout the software development lifecycle (SDLC) [33]:

  - Conducting regular vulnerability assessments
  - Implementing secure coding practices
  - Using web application firewalls (WAFs)
  - Deploying runtime application self-protection (RASP) technology

  Application security incorporates steps taken to improve the safety of an application, often by discovering, correcting, and averting security flaws. Security scanning is

hugely important to protect crucial information and ourselves from cybercrime costs [34].

- **Cloud Security:** is the practice of protecting data and applications hosted in the cloud. It involves setting up security measures to protect data from unauthorized access, hacking attempts, and malicious activities. Cloud security includes measures such as access control, encryption, and vulnerability testing [27]. Access control ensures that only authorized users can access cloud data and applications. Encryption protects data in transit or at rest from unauthorized access. Vulnerability testing identifies and pats vulnerabilities in cloud-based applications and systems. Establishing policies and procedures for cloud security and training employees on cloud security awareness is also important. Cloud security is critical for protecting sensitive data and maintaining the trust of customers [35].

- **Endpoint Security:** is a type of cybersecurity that protects specific endpoints like laptops, smartphones, computers, and other network-connected devices. The objective is to safeguard data availability, confidentiality, and integrity while preserving these endpoints from potential attacks. Implementing numerous methods, including firewalls, intrusion detection and prevention systems, antivirus and anti-malware software, data encryption, and access control mechanisms, constitutes endpoint security. These steps protect endpoints from malware assaults, data breaches, unauthorized access, and other security threats. Since endpoints are frequently targeted by hackers looking to exploit weaknesses and obtain unauthorized access to sensitive information, endpoint security is essential to overall network security [36].

- **Data Security:** refers to the steps to secure digital data against unauthorized access, tampering, theft, or loss. This involves utilizing various technologies and methods to ensure that information is handled and protected correctly. Techniques like encryption, data masking, and redaction may be employed, and automated reporting processes can make audits easier and guarantee compliance with regulations [37].

2.3 Cyberattacks

A cyber attack is defined as any offensive action that is carried out to compromise the confidentiality, integrity, or availability of a computer system or network and interrupt the normal operation of systems, networks, or computer devices.

The first recorded instance of a cyberattack occurred in the 1980s when hackers started using computer viruses and other malicious software to target computer systems. Since then, the frequency and sophistication of these attacks have continued to increase, making them a major threat to individuals and organizations worldwide.

According to [38], cyberattacks are a major threat to national security and economic stability, as they can cause significant harm to individuals, organizations, and society. The article's authors highlight the growing trend of cybercrime-as-a-service, where attackers use automated tools and techniques to carry out large-scale attacks on vulnerable systems, often for financial gain.

In addition, other researchers believe that simply relying on traditional security measures, such as firewalls and antivirus software, is no longer enough to protect against the increasingly sophisticated and targeted nature of cyberattacks [39]. Instead, organizations must adopt a risk-based approach to security, which involves regular threat assessments, ongoing monitoring, and incident response planning to mitigate the impact of cyber attack [40].

Cyberattacks can happen at any time and can come in many forms. The most common attacks include phishing scams, malware infections, and DDoS attacks. These attacks are often carried out by organized criminal groups or state-sponsored hackers who seek to gain unauthorized access to sensitive information or systems [41].

The importance of cyberattacks lies in the potential damage they can cause to individuals, organizations, and even entire countries. This damage can range from financial losses, theft of sensitive information, and reputational harm to critical infrastructure and service disruptions. As a result, it is crucial for individuals and organizations to be aware of the threats posed by these attacks and take steps to protect themselves and their systems [42].

According to a scientific paper [43], the increasing use of connected devices and the growing reliance on the internet have made cyberattacks a critical issue. The rise of artificial intelligence and machine learning has added a new dimension to the threat posed by cyberattacks, as attackers can now use these technologies to automate their attacks and carry out more sophisticated and targeted attacks.

2.3.1 CyberAttacks Types

Malwares

According to [44], Malware, short for malicious software, refers to any software program that is designed to cause harm to a computer system, its users, or the data stored in the system. This can include viruses, worms, Trojans, spyware, adware, and ransomware. Malware can spread through various means, such as email attachments, downloading infected software or files from the internet, or exploiting vulnerabilities in computer systems.

To understand how malware works, it is important to study the different types and their specific methods of operation. For example, [45] refers to viruses as infecting other files and spreading themselves to other computers, while worms can apply to other systems on the same network. Trojans are often disguised as legitimate software and are used to gain unauthorized access to a computer system, while ransomware encrypts the user's files and demands payment in exchange for the decryption key.

- **Ransomware:** This type of malware is designed to encrypt a victim's files or lock them out of their computer until they pay a ransom to the attacker. Examples of ransomware include WannaCry and Petya.
- **Trojan Horse:** A Trojan horse is a type of malware that disguises itself as legitimate software or application to trick the user into installing it. Once installed, the Trojan can perform various malicious actions, such as stealing personal information, downloading additional malware, or creating a backdoor to the victim's system.

- **Virus:** A computer virus is a type of malware that can replicate itself and spread to other systems. Once activated, viruses can cause many problems, from stealing data to crashing systems.
- **Adware:** Adware is a type of malware designed to display unwanted ads or pop-ups on a victim's computer. Adware can slow down the system and is often bundled with legitimate software.
- **Spyware:** Spyware is a type of malware designed to spy on a victim's activities and steal sensitive information, such as passwords or financial data.

Phishing Attacks

The authors in [46] mention that Phishing is a kind of social engineering attack in which criminals use spoofed emails to trick people into sharing sensitive information or installing malware on their computers. These attacks typically involve using fake emails, websites, or pop-up windows that appear to be from a trusted source, such as a bank, an online retailer, or a social media site. The attacker then uses the information from the phishing attempt to gain unauthorized access to the victim's accounts or steal their personal data.

The effectiveness of phishing attacks is largely due to their ability to exploit human psychology. Phishing attackers can manipulate individuals into giving up sensitive information by creating a sense of urgency or using social engineering tactics. In addition, sophisticated technologies, such as domain spoofing, make it easier for phishing attackers to create convincing fake emails and websites that appear to be from legitimate sources [47].

Zero-day Attacks

The authors in [48] mention that a zero-day attack exploits a vulnerability that has not been disclosed publicly. There is almost no defense against a zero-day attack: while the vulnerability remains unknown, the software affected cannot be patched, and anti-virus products cannot detect the attack through signature-based scanning. This makes these attacks particularly dangerous because they can go unnoticed for a long time.

Zero-day attacks typically work by reverse-engineering software or hardware to find and exploit vulnerabilities. Once a vulnerability is discovered, the attacker can create a custom payload to deliver the attack. The payload may be delivered through various methods, such as a malicious email attachment, a compromised website, or a vulnerability in a widely used software program [49].

Once the vulnerability is exploited, the attacker can gain unauthorized access to the targeted system or device. They can then steal sensitive information, install malware, or cause other damage. The long-term impact of a zero-day attack can be significant, as attackers may have access to a system for a long time before they are detected. This is why it is important to update software and procedures regularly and to have robust security measures in place to detect and respond to potential threats [50].

The effect of zero-day vulnerability also depends on the mode of detection. If white hat hackers identify vulnerability, it allows keeping it low profile until the security patches

are unavailable, whereas identification of such vulnerabilities by a notorious group (black hat hackers) may subject the entire enterprise to failure [51].

Denial of Service (DoS) Attacks

A Denial of Service (DoS) attack can be characterized as an attack to prevent legitimate users from using a specified network resource such as a website, web service, or computer system [52].

A DDoS attack is a cyber-attack that aims to make a network or website unavailable to users by overwhelming it with excessive traffic. The episode is carried out indirectly using many compromised computers, often called "secondary victims." The attacker uses these secondary victims to launch a much larger and more disruptive attack than possible through a single source, making it harder for network forensics to identify the attacker's origin. The goal is to render the services of the "primary victim" inaccessible, causing widespread disruption and potential harm [53].

SQL Injection Attacks

SQL injection attacks, according to [54], are a type of cyber attack that exploits vulnerabilities in software applications that interact with a database. They occur when an attacker can inject malicious SQL code into an application's input fields, which is then executed by the database. The consequences of a successful SQL injection attack can range from unauthorized data access to full system compromise, making it a serious threat to web application security.

According to a research article in [55], SQL injection attacks are among the most common web application attacks. The authors conducted a study of over 9,000 web applications and found that 13% of them were vulnerable to SQL injection attacks. Furthermore, the authors found that many of these vulnerabilities were caused by insufficient input validation or encoding. This highlights the importance of properly validating user input in web applications to prevent SQL injection attacks.

SQL injection attacks are a significant threat to the security of web applications. Research has been conducted to develop methods to prevent these attacks, including automated detection and prevention systems and machine learning techniques. These studies highlight the importance of implementing secure coding practices and using advanced technologies to prevent SQL injection attacks [56].

Man-in-the-Middle (MitM) Attacks

Man-in-the-middle (MitM) attacks are a common and significant threat to cybersecurity. In his article, [57] explains that MitM attacks occur when an attacker intercepts the communication between two parties and can read, modify, or inject malicious content into the exchanged messages. Sharma notes that MitM attacks are especially effective in public Wi-Fi networks, where attackers can easily eavesdrop on unencrypted traffic.

Another article [58] outlines various MitM attacks, such as IP spoofing, DNS spoofing, SSL stripping, and HTTP request hijacking. These attacks often steal sensitive information like login credentials, credit card numbers, or personal data. The article also highlights

the increasing sophistication of MitM attacks, such as using machine learning to predict encryption keys, making it more challenging for security experts to detect and prevent them.

Advanced Persistent Threats (APT) Attacks

Advanced Persistent Threats (APT) are complex and sophisticated attacks that aim to gain unauthorized access to a system or network, exfiltrate data, or disrupt normal operations [59]. APTs involve multiple stages and can take weeks, months, or even years to execute, often to remain undetected for as long as possible. According to [60], the attackers behind APTs often have a high level of expertise and resources, allowing them to target specific organizations or individuals with tailored and sophisticated attacks.

Also, the same authors identify four main stages of an APT attack: reconnaissance, initial compromise, establishing a foothold, and escalating privileges. During the reconnaissance phase, the attackers gather information about the target, including its infrastructure and vulnerabilities. In the initial compromise phase, the attackers use various techniques to gain an initial foothold in the system, such as phishing emails or watering hole attacks. Once inside the system, the attackers establish a foothold by installing backdoors or other malware that enables them to maintain access.

On the other hand, the authors in [59] note that APT attacks are becoming increasingly prevalent and sophisticated, making them difficult to detect and prevent.

In summary, APT attacks are a serious and growing threat to organizations, with attackers using sophisticated techniques and tools to remain undetected for long periods. Mitigating these attacks requires a multi-layered approach that combines technical controls, such as firewalls and endpoint protection, with employee training and awareness; however, as APT attacks continue to evolve and become more sophisticated, there is a need for ongoing research and development of new solutions to detect and prevent these attacks [61].

Below are some examples of APT attacks that have occurred over the years

- **Operation Aurora:** This cyber espionage campaign targeted Google and other large companies in 2009-2010. The attackers used sophisticated techniques to infiltrate the companies' networks and steal sensitive data. This attack is often cited as one of the earliest and most significant examples of an APT attack [62].

- **Duqu:** This was a sophisticated piece of malware discovered in 2011. It was designed to steal sensitive information from various targets, including government agencies and large corporations. The attack was believed to be connected to the creators of the Stuxnet worm, which was used to attack Iran's nuclear program [63].

- **Operation Red October:** This was a global cyber espionage campaign that was uncovered in 2013. The attackers targeted government agencies, diplomatic missions, and other high-value targets in more than 30 countries. The attackers used various techniques to infiltrate the targets' networks and steal sensitive information [64].

- **The Sony Pictures hack:** In 2014, the movie studio Sony Pictures was the victim of a devastating cyber attack. The attackers believed to be connected to North Korea,

stole sensitive data and released it publicly to damage the company's reputation. The attack was widely seen as an example of an APT attack due to its sophisticated nature and the attackers' use of multiple techniques to infiltrate the company's network [65].

Finally, we have some types of attacks that fall within an APT, and it is explained why they can be considered part of it:

- **Malware infection:** APT attacks often involve the installation of malware on the systems and devices of targets. This malware can be designed to steal confidential information, manipulate data, or cause damage [66].
- **Zero-day exploits:** APT attacks can take advantage of unknown vulnerabilities (zero-day) in software or hardware to infiltrate systems and devices to execute an attack called zero-day attack [67].
- **Phishing and identity theft:** APT attacks often involve phishing emails and identity theft to deceive users and gain access to their systems and networks.
- **Data exfiltration:** Once attackers have compromised a system or network, they can exfiltrate confidential and valuable information.
- **Stealthy and persistent infiltration:** APT attacks are known for their persistent and stealthy nature, allowing them to remain hidden for extended periods while stealing information or causing damage.

Some examples that fall within the types of attacks mentioned in this section are shown in Table 2.1

2.4 Detection Methods

As the number and complexity of cyberattacks continue to grow, it is becoming increasingly important for organizations to have effective detection methods to identify and respond to potential threats quickly. Several tools and techniques are available to detect cyberattacks, each with its own strengths and weaknesses.

According to [68], there are two commonly used methods for detecting cyberattacks: Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). These tools monitor network traffic for suspicious activity and can quickly alert security teams to potential security breaches. IDS and IPS can be used in various network environments, including large enterprise networks, small and medium-sized businesses, and home networks [69].

Another method used to detect cyberattacks is malware analysis [70], which involves analyzing suspicious files or code to identify whether they contain malicious code or are part of an attack. Malware analysis can be done manually or through automated tools, and it can help identify previously unknown threats that signature-based or heuristic-based detection methods may not detect. Malware analysis can be resource-intensive, but it is essential to many cybersecurity strategies.

Table 2.1: Summary of some examples of attacks that have happened in recent years

Category	Type	Year	Description
Malware	Malware	2017	WannaCry ransomware attack that affected 300,000 computers
Social Engineering	Phishing	2020	Target Corporation data breach involving stolen credentials
Network-based	Dos	2016	GitHub DDoS attack launched via memcached serversk
Exploit	Zero-day	2015	Stuxnet worm that targeted Iran's nuclear program
Web-based	SQL Injection	2014	Attack on Equifax that exposed sensitive information
Network-based	MitM	2017	Operation Aurora attack on Google and other companies
Targeted-Attacks	APT	2016	APT10 attack on technology companies and government agencies

2.4.1 Intrusion Prevention System (IPS)

An Intrusion Prevention System (IPS) [71] (see Fig. 2.2) is a type of security system that is designed to prevent intrusions from occurring. Use advanced technologies to detect and prevent intrusions in real time. An IPS aims to detect and prevent intrusions before they can cause damage.

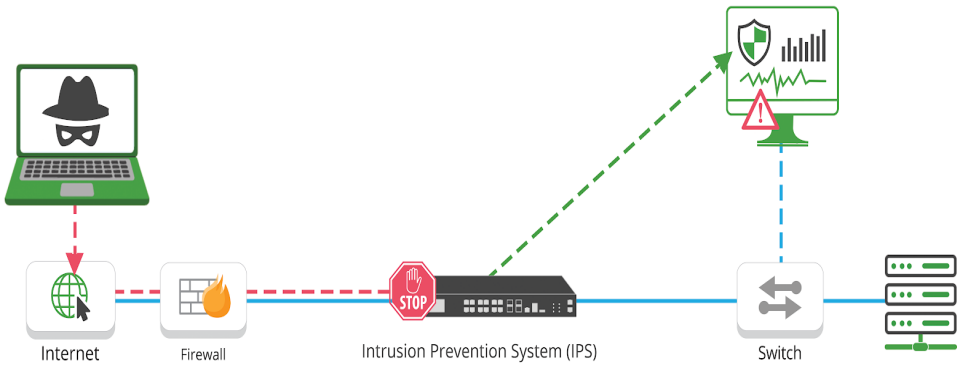


Figure 2.2: Basic Structure of IPS (Intrusion Prevention System)

One of the key benefits of an IPS is its ability to detect and prevent a wide range of

intrusions. These systems are designed to detect and prevent many intrusions, including viruses, malware, and zero-day vulnerabilities. This means that an IPS can protect against known and unknown threats, providing a comprehensive approach to intrusion prevention [72].

Another critical aspect of an IPS is its ability to respond quickly to intrusions. Intrusion Prevention Systems are designed to respond to intrusions in real-time, which helps to minimize the damage caused by intrusions. The real-time response of an IPS is critical in preventing intrusions from causing significant harm, as it ensures that appropriate action is taken as soon as an intrusion is detected [73]. Like a Firewall, an IPS proactively blocks network traffic when it detects a packet that matches a known security threat according to security profiles.

According to [74], the IPS constantly and vigilantly monitors real-time network traffic around the clock, searching for malicious activity and gathering information about such incidents. Administrators are promptly informed of these events, and the IPS takes preventive measures, which may involve blocking suspicious traffic, adjusting firewall settings, or closing vulnerable access points to thwart future attacks. IPS is also valuable in detecting violations of corporate security policies, countering unintentional threats from users, or deterring unauthorized probes from employees and guests.

Finally, an IPS is essential for organizations to secure their computer systems. This method is crucial in securing computer systems against intrusions and other threats.

2.4.2 Intrusion Detection System (IDS)

An Intrusion Detection System (IDS) [75] (see. Fig 2.3) is a technology used for monitoring a network or a computer system for malicious activities or policy violations. IDS can be host- or network-based and operate in real-time or after-the-fact. IDSs play a critical role in ensuring the security of computer systems by detecting and responding to unauthorized access, misuse, and other security-related events.

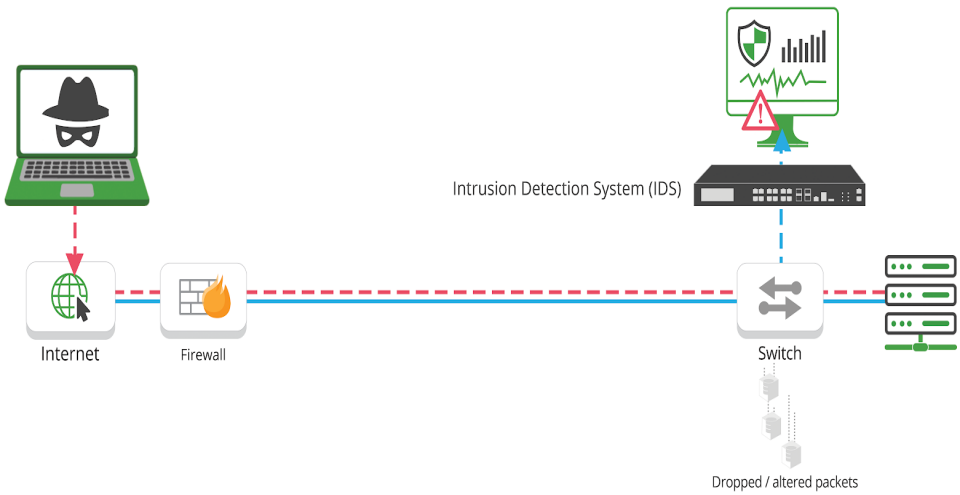


Figure 2.3: Basic Structure of IDS (Intrusion Detection System)

IDS is a crucial network security instrument that examines network traffic to identify any malevolent activities, attempts to exploit vulnerabilities, or breaches of security policies that seek to infiltrate or extract data from the network.

The IDS, according to [74], identifies potential threats by comparing ongoing network activity with a comprehensive database of known threats, focusing on various critical behaviors like security policy violations, malware presence, and port scanning activities. Whenever a breach or intrusion is detected, the IDS promptly notifies the administrator or collects and manages the information through a security information and event management (SIEM) system. The SIEM [76] significantly distinguishes genuine malicious activities from false alarms, ensuring that appropriate actions address potential security risks.

Security Threats Capable Of Being Detected By IDS

An intrusion detection system (IDS) can detect a wide range of security threats, including:

- **Malware infections:** Malware infections are a major threat to computer systems and networks, as they are designed to cause harm to the affected systems, steal sensitive information, or compromise user privacy.
- **Unauthorized access attempts:** such as hacking, brute force attacks, and unauthorized access to sensitive information
- Denial of Service (DoS) attacks: including network-based and application-based DoS attacks
- **Configuration changes:** including changes to system settings, firewall rules, and security policies
- **Data exfiltration:** including unauthorized transfer of sensitive information, such as intellectual property or personal data
- **Remote code execution:** including attempts to execute malicious code on a system through vulnerability exploits or phishing attacks
- **Injection attacks:** including SQL injection, cross-site scripting (XSS), and command injection attacks
- **Packet sniffing and eavesdropping:** including the unauthorized capture and analysis of network traffic
- **Man-in-the-Middle (MITM) attacks:** including attempts to intercept and modify network traffic
- **Distributed Denial of Service (DDoS) attacks:** including attacks that use multiple systems to overload a target system
- **Rogue software:** R including unauthorized software installations, such as backdoors, spyware, and adware

- **Policy violations:** including attempts to bypass security policies, such as attempts to use unauthorized software or access restricted resources

These are some of the common types of security threats that an IDS can detect. IDSs continuously monitor network and system activity, analyze traffic patterns, and alert organizations to potential security threats.

On the other hand, according to [77], IDS can be classified into two main categories: signature-based and anomaly-based; in addition to these, three other types are more specific to meet the desired objective in a specific problem, such as host-based, network-based and hybrids.

Signature-based IDS detects intrusions by matching the system's behavior against a database of known attack patterns. Anomaly-based IDS identifies intrusions by flagging deviations from normal behavior patterns in the system. IDS are more effective in detecting new and unknown attacks than signature-based IDS.

The deployment of IDSs, however, is not without challenges. False positive and false negative alarms, high costs, and difficulty in maintaining the systems are some of the common challenges faced by IDS users [78]. To overcome these challenges, some researchers, for example in [79], have proposed various approaches, including using machine learning algorithms and integrating IDSs with other security systems. It's important to mention the importance of combining multiple IDSs to increase intrusion detection accuracy and reduce false alarms.

Machine learning algorithms have become a popular approach for improving the performance of IDSs [80]. These algorithms can be trained on historical data to identify and detect deviations from normal behavior patterns. A study [79] compared the performance of several machine learning algorithms for intrusion detection and found that decision tree-based algorithms were the most effective.

Finally, integrating IDSs with other security systems, such as firewalls and antivirus software, can also enhance the security of computer systems [81]. This integration can provide a more comprehensive system view and reduce the likelihood of missed intrusions [82]. An article [83] discussed the benefits and challenges of integrating IDSs with other security systems and recommended approaches for successful integration.

These articles demonstrate ongoing efforts to improve the performance and effectiveness of IDS as a tool to protect computer systems. The most important or specific types that may exist to solve intrusion detection problems were also mentioned. The latter will be detailed below:

Signature-based Intrusion Detection System (SIDS)

A signature-based Intrusion Detection System (IDS) (see Fig. 2.4) is a type of IDS that detects intrusions by matching the behavior of a computer system against a database of known attack patterns or signatures. They are the most commonly used type of IDSs due to their ease of implementation and ability to detect well-known attacks [84].

Signature-based IDSs compare the system's behavior against a database of attack signatures, unique patterns characteristic of specific attacks. If a match is found, the IDS will raise the alarm and take appropriate action, such as blocking the attack or alerting the security administrator [85]. Signature-based IDSs effectively detect well-known attacks, but their effectiveness decreases as new and unknown attacks increase [86].

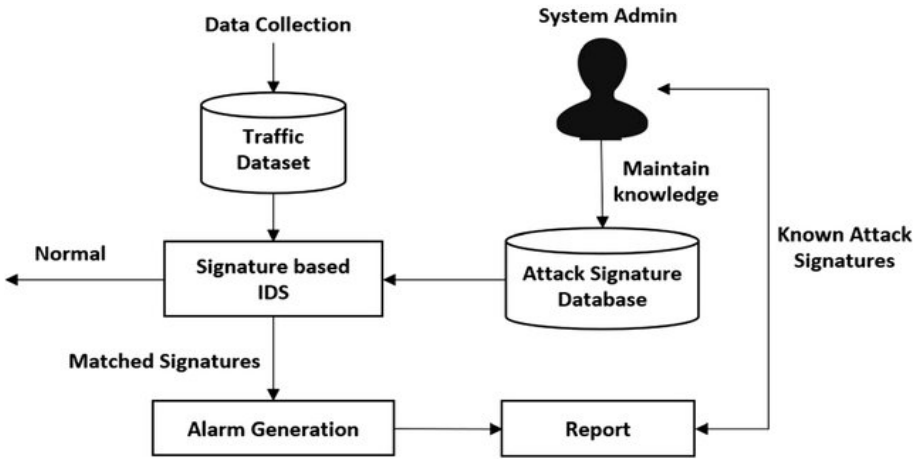


Figure 2.4: Signature-based IDS [1]

One of the key advantages of signature-based IDSs is their ease of implementation and maintenance. The attack signatures are pre-defined, and the IDS must only match the system's behavior against these signatures. This makes signature-based IDSs a popular choice for organizations that lack the resources or expertise to implement more complex IDSs. Signature-based IDSs are relatively easy to deploy and maintain, making them a popular choice for organizations with limited security budgets [87].

However, the efficacy of signature-based IDSs is limited by the quality and completeness of the attack signature database. The authors in [85] mention that if the database is outdated or incomplete, the IDS may be unable to detect new and unknown attacks. The effectiveness of signature-based IDSs is dependent on the quality and completeness of the attack signature database. If the database is not updated regularly, the IDS may be unable to detect new and unknown attacks.

In conclusion, according to [87], signature-based IDSs are a popular and effective tool for detecting well-known attacks. Still, their efficacy decreases as new and unknown attacks increase. Organizations that choose to implement signature-based IDSs should ensure that the attack signature database is regularly updated to maintain the effectiveness of the IDS. These articles demonstrate the importance of periodically updating the attack signature database for signature-based IDSs to remain an effective tool for securing computer systems.

Anomaly-based Intrusion Detection System (AIDS)

Anomaly-based intrusion detection systems (see Fig. 2.5) are a newer and more advanced method of IDS. They establish a baseline of normal network behavior based on statistical analysis, machine learning, or artificial intelligence. Then, they monitor the network traffic for any deviations from the baseline or anomalies that indicate malicious activity. If an abnormality is detected, the IDS alerts or blocks the attack [88].

According to this article [89], signature-based IDSs effectively detect common and well-known attacks, such as malware, worms, or denial-of-service (DoS) attacks. They are also easy to deploy and maintain, making them a popular choice for organizations with limited security budgets. However, the efficacy of signature-based IDSs is limited by the quality

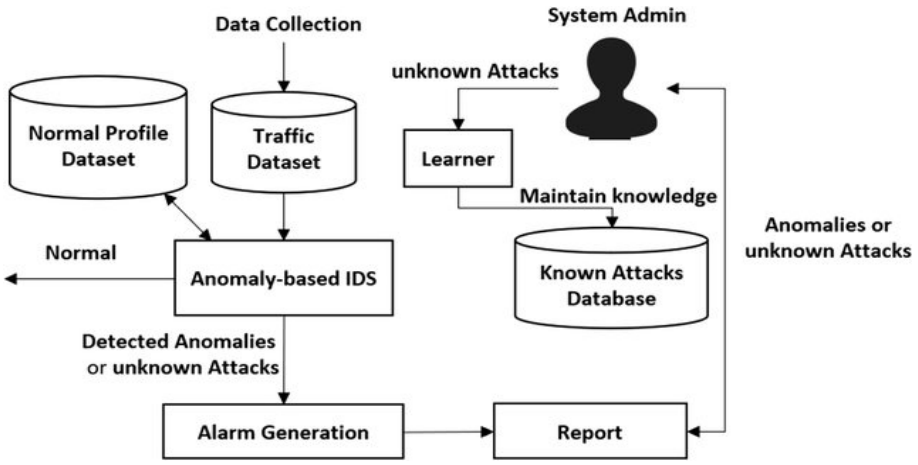


Figure 2.5: Anomaly-based IDS [1]

and completeness of the attack signature database. Anomaly-based IDSs, on the other hand, are capable of detecting new or unknown attacks, which may not have a signature yet. They are also adaptive and dynamic, as they can update their baselines or models according to the changes in the network environment.

This type of IDS works on the principle of detecting a deviation from the expected system behavior. They rely on baselines or models representing the expected network traffic or activities. However, anomaly-based IDSs also create many false positives, as they may detect legitimate activities as anomalies [90].

Also, in [91], the authors mention that these types of IDS are built by training the IDS on traffic data to learn the normal behavior of the network. They then use this knowledge to detect deviations from the normal, which may indicate malicious activity. However, anomaly-based IDSs require a lot of training data and may not be effective against attacks that are similar to normal network traffic

These articles highlight the benefits of using an anomaly-based IDS in detecting new and previously unseen threats and complex and difficult-to-detect intrusions. Anomaly-based IDSs can effectively complement signature-based methods and provide an additional layer of security for organizations looking to defend against intrusions.

Network-based IDS

Network-based IDS (NIDS) are designed to monitor and analyze network traffic for signs of malicious activity. They are placed at strategic points within the network to monitor traffic to and from all devices on the network. NIDSs analyze passing traffic on the entire subnet and match the traffic passed on the subnets to the library of known attacks. Once an attack is identified or abnormal behavior is sensed, the alert can be sent to the administrator [92].

NIDSs can detect various attacks, including denial-of-service (DoS) attacks, port scans, and malware infections. They can also detect attacks not visible to host-based IDSs, such as attacks that exploit vulnerabilities in network protocols or applications [93].

One of the main advantages of NIDSs, according to [92], is their ability to monitor network traffic in real-time and detect attacks as they occur. This allows for a faster response time and can help prevent the attack from causing significant damage. NIDSs

can also monitor network activity and identify potential security threats before they become problematic.

Moreover, some authors in the article [94] mention that NIDSs can be deployed in various ways, including as a standalone device, as part of a firewall, or as a software application running on a server. They can also be configured to monitor specific types of traffic, such as email or web traffic, or traffic on particular ports.

Finally, one of the main challenges of NIDS is the high number of false positives they can generate. This can be due to the complexity of network traffic and the difficulty of distinguishing between legitimate and malicious activity [95]. To address this issue, NIDSs can be configured to filter out known benign traffic and to use machine learning algorithms to improve their accuracy over time.

Host-based IDS

Host-based Intrusion Detection Systems (HIDS) are designed to monitor the computer infrastructure on which they are installed, analyzing traffic and logging malicious behavior. They can watch all or parts of a computer system's dynamic behavior and state based on how it is configured. Besides such activities as dynamically inspecting network packets targeted at this specific host, an HIDS might detect which program accesses what resources and discover that. For example, a word processor has suddenly and inexplicably started modifying the system password database [96].

Also, they are installed locally on endpoints, such as computers. After configuring it, the host-based IDS will monitor traffic on your business's network and the computer on which it's installed. They can monitor user and file access activity, including file accesses, changes to file permissions, attempts to establish new executables, etc, [97].

HIDSs raise more management concerns as they are configured and operated on each monitored host. They are harder to deploy and manage than network-based IDSs, which are typically easier to set up. Installing a host-based IDS requires a bit more effort and expertise [98].

These can detect attacks that a network-based IDS cannot see since they monitor events local to a host. They can often operate in an environment where network traffic is not present or is encrypted. HIDSs can also analyze a host's file system, users' logon activities, running processes, data integrity, etc [96].

On the other hand, according to [99], these provide real-time visibility into what activities are taking place on the servers, which adds to the additional security. They can verify the success or failure of an attack since they use system logs containing events that have actually occurred; they can determine whether a seizure occurred or not.

Finally, these types of IDS can detect attacks that network-based IDSs fail to detect, but they are harder to deploy and manage than network-based IDSs. They are installed locally on endpoints, such as computers, and can monitor user and file access activity, including file accesses, changes to file permissions, attempts to establish new executables, etc [98] [97].

Hybrid Intrusion Detection System

A hybrid intrusion detection system (IDS) (see Fig. 2.6) combines two or more approaches to intrusion detection to provide a more effective and comprehensive view of the network system [99]. Hybrid IDSs are designed to overcome the limitations of single IDSs, such as host-based IDSs (HIDS) and network-based IDSs (NIDS), by combining their strengths [100]. One of the most common hybrid IDSs combines the signature and anomaly models, known as a hybrid-based IDS [101].

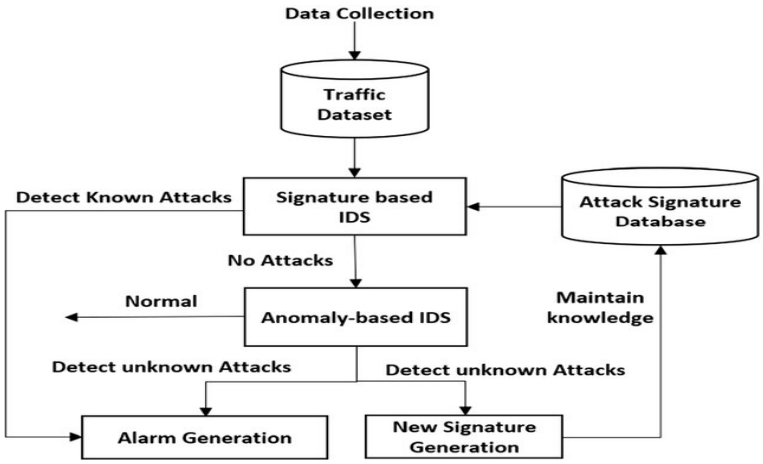


Figure 2.6: Hybrid IDS [1]

A hybrid IDS can be more challenging for an adversary to manipulate or spoof both cyber and physical data streams to execute an attack [102]. It can also detect known and unknown attacks with high accuracy and low false-alarm rates [103]. In addition, a hybrid IDS can provide a more complete view of the network system by combining host agent or system data with network information [99].

To improve the detection rate and processing time, most IDSs share the signatures of the novel attacks detected by anomaly approach [104]. However, a hybrid IDS can capture both host and network data and apply an analysis methodology to provide a more effective solution for network security [105].

One of the challenges of a hybrid IDS is the system's complexity, which requires a deep exploration of various detection methods and the integration of statistical and machine learning-based techniques [106].

Overall, a hybrid IDS can provide a more comprehensive and effective solution for network security by combining the strengths of different IDS approaches. It can detect known and unknown attacks with high accuracy and low false-alarm rates, making it a valuable tool for businesses to protect themselves from cyber threats [107].

2.5 Preventing Cyberattacks

As cyberattacks increase in frequency and complexity, the need for advanced cybersecurity solutions has become increasingly urgent. Artificial intelligence (AI) is emerging as a promising tool for preventing cyberattacks. AI can help detect threats, analyze data, and

develop countermeasures in real-time. According to an article, the use of AI in cybersecurity is expected to grow by 23.7% annually through 2024.

One of the key benefits of AI in cybersecurity is its ability to analyze vast amounts of data quickly and accurately. Using machine learning algorithms, AI can identify patterns and anomalies in network traffic that may indicate a cyberattack. This can help security teams respond to threats in real-time, minimizing the potential damage of a cyberattack.

AI can also help organizations stay ahead of the curve when identifying and preventing new cyber threats. By analyzing historical data and identifying patterns, AI can predict future threats and develop countermeasures to stop them. This proactive approach can be highly effective in preventing cyberattacks before they occur.

Another advantage of AI in cybersecurity is its ability to adapt to new threats. Cybercriminals are constantly developing new tactics and techniques to evade traditional security measures. AI can analyze these new threats and develop countermeasures quickly, helping organizations stay ahead of cybercriminals.

However, some challenges and potential drawbacks to using AI in cybersecurity exist. One concern is the potential for false positives, where benign activities are identified as potential threats. This can lead to unnecessary alerts and increased workload for security teams. Another concern is the potential for cybercriminals to use AI to develop more sophisticated attacks and evade detection by traditional security systems.

To address these concerns, organizations must ensure they have the right people, processes, and technologies to leverage AI in cybersecurity effectively. This includes investing in AI-based security solutions and providing training and education for security teams to use these tools effectively.

In conclusion, the use of AI in cybersecurity is becoming increasingly important as cyber threats evolve and become more sophisticated. By leveraging AI-based solutions, organizations can improve their ability to detect and prevent cyberattacks and better protect their sensitive data and assets. While some challenges and concerns are associated with AI in cybersecurity, with the right strategy and implementation, AI can be an incredibly effective tool in preventing cyberattacks.

2.6 Machine Learning

Machine learning is a subfield of artificial intelligence that uses algorithms and statistical models to enable computers to improve their performance on a given task without explicitly being programmed [108]. Machine learning aims to develop algorithms to learn from data and make predictions or decisions without being explicitly programmed [109]. This is achieved through different techniques such as supervised learning, unsupervised learning, and reinforcement learning [110]. Supervised learning algorithms are trained on labeled data, while unsupervised learning algorithms work with unlabeled data [111]. Reinforcement learning involves training systems using rewards and punishments [112]. Machine learning is used in a wide range of applications such as image recognition [113], natural language processing [114], fraud detection [115], and self-driving cars [116]. With increasing data availability and computational power, machine learning is becoming more prevalent and important in today's world [117].

Machine learning constantly evolves, and new techniques and architectures are being

developed. The availability of large amounts of data and advances in computational power and algorithms have made it possible to train highly accurate models. However, there are also some challenges in machine learning, such as avoiding bias in data, interpretability of models, and ensuring the system's safety.

2.6.1 Supervised Learning

Supervised learning is a type of machine learning that involves training a model on labeled data to make predictions about new, unseen data. The goal of supervised learning is to learn a mapping between input features and output labels, such as class labels or numerical values [118]. Supervised learning is commonly used in classification and regression problems, where the goal is to predict a categorical or continuous variable, respectively.

In supervised learning, the training data consists of input-output pairs, where the input is a set of features, and the output is a label or value. The model is trained to learn the relationship between the input and output by minimizing a loss function that measures the difference between the predicted and actual output. Once the model is trained, it can expect new, unseen data. Supervised learning algorithms include decision trees, logistic regression, and support vector machines.

One of the advantages of supervised learning is that it can be used to make accurate predictions on new data, even when the data is complex or noisy. Supervised learning can also gain insights into the underlying relationships between variables and identify important features contributing to the prediction [119]. However, one of the main challenges of supervised learning is the need for labeled data, which can be time-consuming and expensive.

Supervised learning has been applied in various domains, including healthcare, finance, and natural language processing. For example, a study focused on the use of supervised learning for predicting the risk of heart disease in patients [120]. The research explored the application of supervised learning algorithms to predict the risk of heart disease based on patient data, enabling the identification of high-risk patients and the development of targeted interventions.

Logistic Regression

Logistic regression is a statistical analysis method used to predict a binary outcome based on prior observations of a dataset. It is commonly used in machine learning for binary classification problems, which are problems with two class values, including predictions such as "this or that," "yes or no," and "A or B." Logistic regression models predict a dependent data variable by analyzing the relationship between one or more existing independent variables. For example, logistic regression could be used to predict whether a political candidate will win or lose an election or whether a high school student will be admitted or not to a particular college [121].

Logistic regression is a supervised machine learning algorithm that accomplishes binary classification tasks by predicting the probability of an outcome, event, or target variable. It indicates the class or category of individuals based on one or multiple predictor variables. Logistic regression is widely used in various domains, including healthcare, finance, and marketing. For example, a study focused on using logistic regression to predict heart disease

risk in patients [120]. The research explored the application of logistic regression algorithms to predict the risk of heart disease based on patient data, enabling the identification of high-risk patients and the development of targeted interventions.

Decision Tree

Decision tree learning is a popular supervised learning method in machine learning, data mining, and statistics. It involves constructing a tree-like model of decisions and their possible consequences, including chance events and resource costs. Decision trees are used for both classification and regression tasks and are a non-parametric method, meaning that they make no assumptions about the underlying distribution of the data [122]. Decision trees are a widely used and practical method for supervised learning, and they can be used to model complex relationships between variables

Decision trees are constructed by recursively splitting the data into subsets based on the values of one or more input features. The goal is to create a tree that predicts the value of a target variable based on several input variables. The tree is constructed by selecting the feature that best splits the data into subsets that are as homogeneous as possible concerning the target variable. This process is repeated until the tree is fully grown or until a stopping criterion is met [123]. Decision trees are easy to interpret and visualize, making them popular for data analysis and decision-making.

One of the main advantages of decision trees is their ability to handle both categorical and numerical data. They can also handle missing data and outliers, making them a robust method for data analysis. However, one of the main challenges of decision trees is overfitting, which occurs when the tree is too complex and fits the training data too closely, resulting in poor generalization performance on new data. To address this issue, pruning techniques can simplify the tree and reduce overfitting [124].

Random Forest

Random Forest is a popular machine-learning algorithm for classification and regression tasks. It is an ensemble learning method that combines multiple decision trees to improve the accuracy and robustness of the model. Random Forest is a flexible and easy-to-use algorithm that produces great results without hyperparameter tuning. It is widely used in various domains, including finance, healthcare, and marketing [125].

Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. When building each decision tree, the algorithm uses bagging and feature randomness to reduce overfitting and improve the model's generalization performance. Random Forest is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and improve the performance of the model [126].

One of the main advantages of Random Forest is its ability to handle high-dimensional data and nonlinear relationships between variables. It can also handle missing data and outliers, making it a robust method for data analysis. Random Forest is computationally efficient and can handle large datasets with many features. However, one of the main challenges of Random Forest is the difficulty of interpreting the model and understanding the underlying relationships between variables. To address this issue, feature importance mea-

sures can be used to identify the most important features that contribute to the prediction [127].

2.6.2 Unsupervised Learning

Unsupervised learning is a type of machine learning that involves training models on unlabeled data and allowing them to learn patterns and relationships independently. It is used to discover hidden patterns in data and to gain insights into the underlying structure of the data. Unsupervised learning is commonly used in data clustering, where the goal is to group similar data points together based on their features [128]. Clustering is an unsupervised data science technique where the records in a dataset are divided into groups based on their similarity.

Unsupervised learning is a machine learning technique that involves training models on data without prior knowledge of the output. Instead, the model learns to independently identify patterns and relationships in the data. Unsupervised learning is used to identify hidden structures in data, such as clusters, and to gain insights into the underlying structure of the data. It is commonly used in data mining, where the goal is to discover patterns and relationships in large datasets [129].

One of the main advantages of unsupervised learning is its ability to handle large and complex datasets without the need for labeled data. Unsupervised learning can also identify outliers and anomalies in data, which can be useful in detecting fraud or other unusual activity. However, one of the main challenges of unsupervised learning is the difficulty of interpreting the results and understanding the underlying relationships between variables [130].

Unsupervised learning has been applied in various domains, including finance, healthcare, and marketing. For example, a study focused on using unsupervised learning for customer segmentation in marketing [131]. The research explored the application of unsupervised learning algorithms to segment customers based on their purchasing behavior, enabling the development of targeted marketing campaigns.

2.6.3 Semi Supervised Learning

Semi-supervised learning is a branch of machine learning that leverages labeled and unlabeled data to improve the performance of learning algorithms. It aims to use the abundance of unlabeled data available in many real-world scenarios. By incorporating unlabeled data, semi-supervised learning can enhance the generalization capability of models and address the limitations of supervised learning approaches [132].

In semi-supervised learning, a small portion of labeled data is combined with more unlabeled data during training. This combination allows the model to learn from the labeled data while leveraging the additional information in the unlabeled data. By utilizing the unlabeled data, semi-supervised learning can capture the underlying structure of the data and make more accurate predictions on unseen instances [133].

The benefits of semi-supervised learning are particularly evident in domains where obtaining labeled data is costly or time-consuming. It offers a middle ground between supervised and unsupervised learning, providing a way to use unlabeled data without

requiring extensive labeling efforts. Semi-supervised learning has been successfully applied in various fields, including natural language processing, computer vision, and healthcare [134].

Research in semi-supervised learning focuses on developing algorithms and techniques that effectively utilize labeled and unlabeled data. These approaches aim to exploit the inherent structure and relationships within the data to improve the learning process. Continual semi-supervised learning, for example, explores methods that leverage contrastive interpolation consistency to enhance supervised learning methods [135].

2.7 Anomaly Detection

Anomaly detection is a technique used to identify unusual events, items, or observations that deviate significantly from normal behavior. It is crucial in various domains, including cybersecurity and infrastructure monitoring. Machine learning-based approaches have been widely employed for anomaly detection, leveraging the power of algorithms to learn patterns and identify deviations from normal behavior. These methods have shown promising results in detecting anomalies in complex and dynamic datasets [136].

Anomaly detection rests upon two basic assumptions: anomalies occur rarely, and their features significantly differ from those of normal instances. Anomalies can be linked to problems or rare events, such as hacking, fraud, equipment malfunction, or infrastructure failures. Accurate identification of actual anomalies is crucial for effective anomaly detection, as false positives or data noise can have significant implications from a business perspective [137].

2.7.1 Machine Learning-based Anomaly Detection

Machine learning-based anomaly detection techniques have gained significant attention in various domains, including cloud computing and network security. These techniques leverage the power of machine learning algorithms to identify abnormal patterns or behaviors in data. In cloud computing, a study focused on anomaly detection in cloud virtual machine resource usage using machine learning approaches [138]. The research explored the application of machine learning algorithms to detect anomalies in resource usage patterns, enabling the identification of potential security threats or performance issues in cloud environments.

An important aspect of machine learning-based anomaly detection is the ability to adapt to evolving data distributions and concept drift. As the statistical distribution of data and their corresponding classes change over time, developing concept drift-aware intrusion detection systems is crucial. This ensures that the anomaly detection models effectively detect new and emerging threats [139]. The systematic review of machine learning for anomaly detection highlighted the need for developing robust and scalable algorithms that can handle concept drift and adapt to changing environments [140]. Algorithms like Isolation Forest and One-Class Support Vector Machines are commonly used for anomaly detection.

Isolation Forest (IF)

The Isolation Forest algorithm is a fast tree-based algorithm for anomaly detection. It leverages the concept that anomalies are "few and distinct" data points in a dataset [141]. Similar to random forests, Isolation Forests are built using decision trees and implemented in an unsupervised fashion. The algorithm randomly selects characteristics and processes the randomly subsampled data in a tree structure. Data points that require fewer cuts to separate them have a higher probability of being anomalies. This algorithm has been widely used for anomaly detection in various domains, including cloud computing and network security [141].

Isolation Forest is a popular unsupervised machine learning algorithm for detecting anomalies within datasets. It has several advantages compared to traditional distance and density-based models, including reduced computational times, scalability to high-dimensional datasets, and the ability to handle irrelevant features [142]. The algorithm works by isolating anomalies in the dataset using randomly constructed decision trees. By measuring the average path length of data points in the trees, anomalies are identified as those with shorter average path lengths, indicating their distinctness from most of the data [143]. Isolation Forest has been successfully applied in various domains, including automotive communication systems and wind turbine fault detection [144].

To continue, in Fig. 2.7, shows a basic operation of the IF:

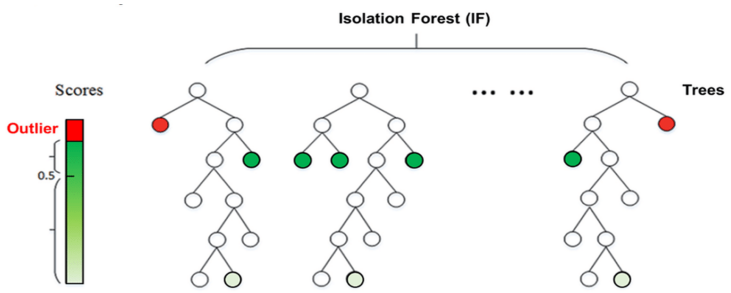


Figure 2.7: IF Model Functionality Example [1]

One-Class Support Vector Machine (OC-SVM)

One-Class Support Vector Machine (OC-SVM) is a machine learning algorithm for anomaly detection. It is a variant of the Support Vector Machine (SVM) algorithm that is trained on a single class of data, typically the normal class, and then used to identify anomalies in new data [145]. OC-SVM has been applied in various domains, including structural health monitoring and intrusion detection systems. For example, a study focused on simulation-based anomaly detection and damage localization in structural health monitoring using OC-SVMs [146]. The research explored the application of OC-SVMs to detect anomalies in structural health monitoring data, enabling the identification of potential damage or performance issues.

One of the main advantages of OC-SVM is its ability to handle high-dimensional data and nonlinear relationships between variables. It can also be trained on a small amount of data, making it suitable for applications where data is scarce [147]. OC-SVM works

by mapping the data into a high-dimensional feature space and finding a hyperplane that separates the normal data from the anomalous data. The distance between the hyperplane and the data points is used to identify anomalies, with data points farther away from the hyperplane being more likely to be anomalous [148]. OC-SVM has been successfully applied in various domains, including intrusion detection systems and tunnel personnel safety detection technology [149].

2.8 Dimensionality Reduction Techniques

These techniques refer to the methods used to transform high-dimensional datasets into lower-dimensional representations while retaining the essential information and properties of the original data. These techniques are commonly employed as a preprocessing step in data science and machine learning tasks to address the challenges posed by high-dimensional data. By reducing the dimensionality, these techniques aim to simplify the data, improve computational efficiency, and enhance the interpretability and visualization of the data [150].

2.8.1 Why is it Important?

Dimensionality reduction techniques are important in data science and machine learning because they help to simplify high-dimensional datasets, improve computational efficiency, and enhance the interpretability and visualization of the data [151]. High-dimensional data can contain many features, some of which may be redundant or unnecessary, making modeling complicated. Furthermore, interpreting and understanding the data by visualization becomes difficult because of the high dimensionality. Dimensionality reduction techniques address these challenges by transforming the data into a lower-dimensional space while preserving important information.

They are also used in various applications, including data visualization, feature extraction, and improving the performance of machine learning models [152]. They are beneficial when labeled data is scarce or expensive to obtain. Dimensionality reduction techniques are a part of the data pre-processing step, performed before training the model. The choice of dimensionality reduction technique depends on the specific problem and the characteristics of the data. Systematic and controlled experiments can be used to figure out which technique works best for a given dataset and model [153].

Finally, these enable more efficient and effective analysis of complex datasets by reducing the dimensionality of the data while preserving important information. On the other hand, these are a part of the data pre-processing step, performed before training the model, and the choice of technique depends on the specific problem and the data characteristics [154].

2.8.2 Approaches and Algorithms

In this area, several approaches and algorithms help us with data dimensionality problems; these are used based on the objective of the problem we want to solve. In Fig. 2.8, we can

observe in detail the two approaches to dimensionality reduction, Feature Selection and Feature Projection, where each one has its respective types with their algorithms. This will be described in detail below:

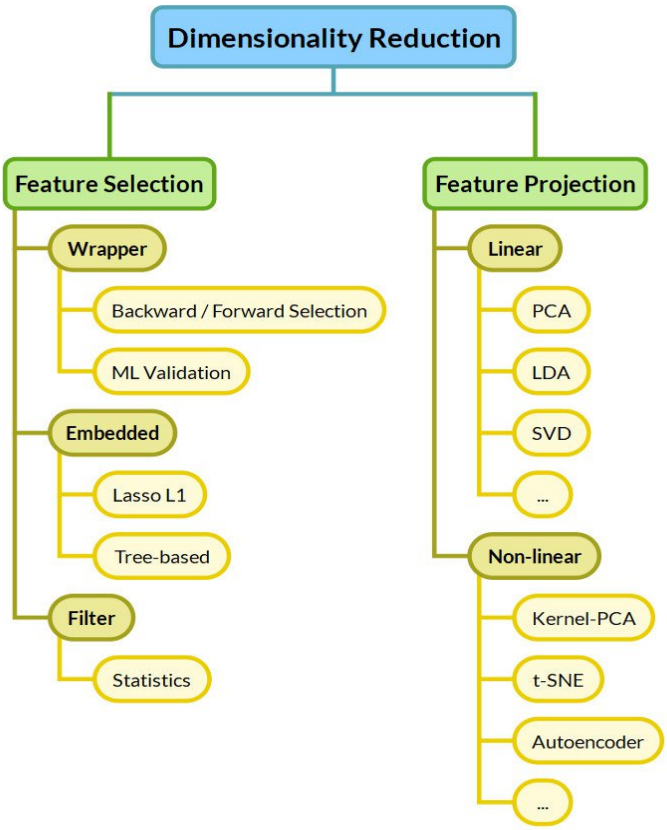


Figure 2.8: Dimensionality reduction approaches and algorithms [2]

2.8.3 Feature Selection

Feature selection is one of the two main approaches to dimensionality reduction, which involves selecting a subset of the original features most relevant to the problem. The goal is to reduce the dimensionality of the dataset while retaining the most important features [150]. Feature selection helps mitigate the effect of high dimensionality on the dataset by finding the subset of features that will effectively define the data [155]. Feature selection is different from dimensionality reduction, as both methods seek to reduce the number of attributes in the dataset, but a dimensionality reduction method does so by creating new combinations of attributes, whereas feature selection methods include and exclude attributes present in the data without changing them [156].

Feature selection is useful in its own right, but it mostly acts as a filter, muting out features that are not useful in addition to your existing features [156]. Feature selection is an important part of any machine learning process, as it can help improve a learning algorithm's performance and reduce the risk of overfitting. Feature selection can also

help reduce the model's computational complexity and improve its interpretability [157]. Various feature selection techniques are available, including filter, wrapper, and embedded methods. Each technique uses a different criterion to evaluate the relevance of the features and select the most important ones [158].

Wrapper

It is a feature selection approach that involves selecting a subset of features by training and testing a specific machine learning model. It assesses the model's performance with different subsets of features and determines the one that yields the best performance [158]. Wrapper methods are so-called because they wrap a classifier up in a feature selection algorithm. Typically, a set of features is chosen, the efficiency of this set is determined, some perturbation is made to change the original set, and the efficiency of the new set is evaluated [159]. The problem with this approach is that the feature space is vast, and looking at every possible combination would take a lot of time and computation. However, wrapper methods are more computationally expensive than filter methods, as they require training and testing the model multiple times [155]. This method has two principal algorithms or techniques, which are:

- **Backward / Forward Selection:** Backward and forward selection are two common wrapper methods used to select a subset of features most relevant to the problem. Backward selection starts with all the features and iteratively removes the least important feature until the desired number of features is reached. Forward selection starts with no features and iteratively adds the most important feature until the desired number of features is reached [160]. These methods are computationally efficient and can be used with any machine learning algorithm [161].

  Backward and forward selection are widely used in various applications, including clinical prediction modeling, genomic prediction, and credit scoring models. In a study on genomic prediction, [162] applied backward selection to identify the most important features for predicting human complex traits. In a study on credit scoring models, [163] used backward stepwise selection to identify the most important features for predicting credit risk. In clinical prediction modeling, variable selection is an important step, and both forward and backward selection methods are used to identify the most important features.

- **ML Validation:** Machine learning validation is a part of the wrapper method that evaluates the model's performance using cross-validation techniques. Cross-validation is a technique used to assess the performance of machine learning models by splitting the data into training and testing sets. The model is trained on the training set and evaluated on the testing set. This process is repeated multiple times, and the average performance is used to evaluate the model [164]. Machine learning validation is an important step in the wrapper method, as it helps to evaluate the model's performance and avoid overfitting.

Embedded

These methods are a type of feature selection technique that involves selecting a subset of features by incorporating the feature selection process into the model training process. Embedded methods are built into the machine learning algorithm and select the most relevant features during the training process [165]. Embedded methods are computationally efficient and can be used with any machine learning algorithm [166].

Also, they are widely used in various applications, including image classification, text classification, and bioinformatics. A study on image classification [167] used an embedded method to select the most relevant features for classifying images of different objects. In a study on text classification, [168] used an embedded method to choose the most pertinent features for organizing text documents into different categories.

Finally, these methods have advantages over other feature selection techniques, such as filter and wrapper. They are computationally efficient and can be used with any machine learning algorithm. Furthermore, they can perform better than filter methods, as they consider the interactions between features [158]. However, embedded methods may not be suitable for datasets with many features, as the computational cost may become prohibitive [160].

In this method, like the previous one, there are two main techniques or algorithms, which we will explain in detail below:

- **Lasso (L1):** This is a popular embedded method for feature selection in dimensionality reduction. It is a regularization technique that adds a penalty term to the model's objective function, encouraging the coefficients of irrelevant features to be reduced to zero. This effectively performs feature selection by shrinking the coefficients of less important features towards zero while keeping the coefficients of important features non-zero [165]. This can help to improve the interpretability of the model and reduce overfitting. However, it is important to tune the regularization parameter to control the sparsity of the selected features and balance between feature selection and model performance [166].

- **Tree-based:** Tree-based methods are a type of embedded feature selection technique that involves selecting a subset of features by incorporating the feature selection process into the model training process. Tree-based methods are built into decision tree algorithms and select the most relevant features during the training process [165]. Tree-based methods are computationally efficient and can be used with any machine learning algorithm [159]. Also, these have several advantages in feature selection. They are computationally efficient and can handle both continuous and categorical features. Furthermore, they can capture non-linear relationships between features and improve the interpretability of the model by identifying the most important features [165]. However, tree-based methods may not be suitable for datasets with many features, as the computational cost may become prohibitive [159].

Filter

These methods utilize statistical techniques to assess the relationship or dependence between input variables and select the most relevant features based on their scores or measures [169]. According to [170], statistical measures such as information gain, chi-square

test, Fisher score, correlation coefficient, and variance threshold are commonly used in filter-based feature selection to evaluate the importance of features. These measures help identify irrelevant attributes and filter out redundant columns from the dataset. The selected features are ranked based on their scores, allowing for isolating measures that enrich the model.

Filter methods are not subjected to overfitting and can be used on high-dimensional data. However, filter methods may provide inferior results compared to the wrapper or embedded methods if the data cannot model the statistical correlation between the feature variables [171].

2.8.4 Feature Projection

Feature projection, also known as projection-based dimensionality reduction, is one of the approaches used in dimensionality reduction techniques. It involves mapping the data from a high to a lower-dimensional space while preserving important information [172].

The goal is to reduce the number of features or dimensions while retaining the most relevant information for analysis or modeling purposes. There are various techniques for feature projection in dimensionality reduction, including principal component analysis (PCA), singular value decomposition (SVD), linear discriminant analysis (LDA), t-distributed stochastic neighbor embedding (t-SNE) [150]. These linear and non-linear techniques aim to transform the data into a new set of variables that capture the most significant patterns or variances in the original data. By projecting the data into a lower-dimensional space, feature projection methods can simplify the data representation, improve computational efficiency, and potentially enhance the interpretability of the results. There are two feature projection techniques available, including linear and non-linear methods. Each approach uses a different criterion to evaluate the relevance of the features and select the most important ones [158].

Linear

Linear feature projection is a feature projection technique used in dimensionality reduction. It involves mapping the data from a high-dimensional space to a lower-dimensional space using a linear transformation while preserving important information [172]. The goal is to reduce the number of features or dimensions while retaining the most relevant information for analysis or modeling purposes.

This method is so named because it uses a linear transformation to project data from a high to a low-dimensional space. The linear transformation is a mathematical function that maps an input vector to an output vector using a linear transformation matrix. The linear transformation is applied to each input feature vector to obtain a new data representation in the space of least dimension [165].

Principal Component Analysis (PCA) serves as a prevalent method for feature projection, pinpointing the dimensions with the highest data variance and subsequently mapping the data onto these principal components [173]. Singular Value Decomposition (SVD) is another valuable technique, breaking down the data matrix into singular values and their associated singular vectors, thereby enabling dimensionality reduction [172]. In contrast,

Linear Discriminant Analysis (LDA) focuses on uncovering a linear blend of features that optimizes the differentiation among various classes or categories [174].

- **PCA:** This dimensionality reduction technique simplifies large data sets. This statistical technique is used to reduce data complexity while maintaining as much information as possible. PCA is used to find patterns in data and reduce the number of variables needed to describe the data. PCA is an unsupervised technique that does not require previous labels or categories for the data [175].

  This technique can be broken down into five steps. First, the original variables are standardized to ensure that they all have the same scale. Then, the covariance matrix of the variables is calculated, which measures the relationship between them. The eigenvalues and eigenvectors of the covariance matrix are then computed, where the eigenvalues represent the amount of variance explained by each principal component, and the eigenvectors indicate the direction of each component. The eigenvalues are then sorted in descending order, and the corresponding eigenvectors are selected. These eigenvectors form a transformation matrix that allows the original data to be projected into a new lower-dimensional space. Finally, the original data is transformed using the selected eigenvectors, resulting in a transformed data set in the principal component space [176]. A summary of these steps can be observed in Fig. 2.9:

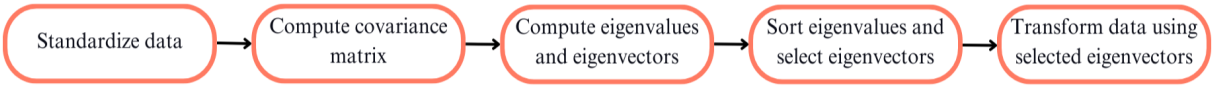


Figure 2.9: Steps to perform a PCA

  Although PCA is a powerful technique for reducing the dimensionality of data, it has some limitations. One of the main limitations of PCA is that it does not work well with non-linear data. If the data has non-linear patterns, such as curves or cycles, the PCA may not capture the underlying structure well [177]. In addition, this technique can be difficult to interpret since the principal components are linear combinations of the original variables and do not have a direct interpretation [178].

  In summary, this method is used to project the data onto the first principal components and obtain a lower-dimensional representation that preserves the largest possible variance of the original data. The first principal component captures the data's largest variance, followed by the second principal component, and so on [179].

  Finally, in Fig. 2.10, we can see how the data was used as an example by reducing its dimensionality. This figure, taken from [3], shows us the MNIST data set with the respective separation of its data in two-dimensional space. Each group represents a number between 1 and 6, and it is also observed that the group with the digit 0 is well separated. However, the group with the digit 5 is distributed more diffusely.

- **LDA:** This is another technique commonly used in machine learning for supervised classification problems. LDA aims to find a lower-dimensional data representation that maximizes the separation between different classes or categories. It projects the

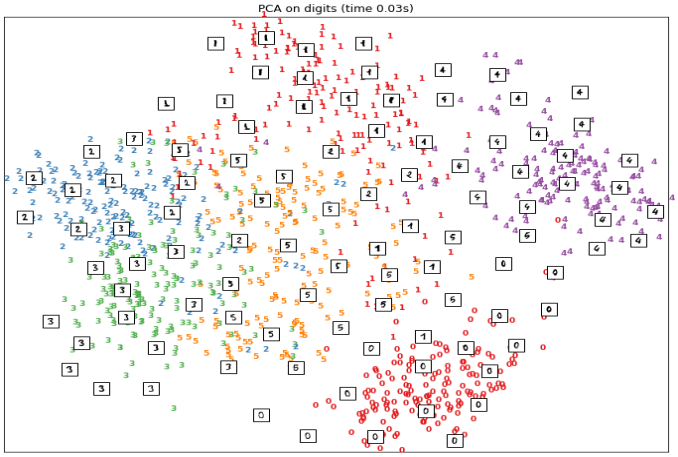


Figure 2.10: Visualization example of a dimensionality reduction using PCA [3]

original high-dimensional data onto a lower-dimensional space while preserving the linear structure of the data. It works by finding a linear combination of features that maximizes the ratio of between-class variance to within-class variance [180]. This algorithm reduces the dimensions from the original features to C-1, where C is the number of classes. It achieves this reduction by selecting the linear discriminants that capture the most discriminative information between types. These linear discriminants are computed by analyzing the mean and covariance of the data from different classes. The resulting lower-dimensional representation can be used for classification tasks [181].

Non-Linear

Nonlinear feature projection is a common approach for dimensionality reduction in problems where data has a nonlinear structure that cannot be easily separated in a lower-dimensional space using linear techniques [182]. Nonlinear dimensionality reduction techniques are essential for discovering the nonlinear degrees of freedom that underlie complex natural observations, such as human handwriting or images of a face under different viewing conditions [183]. One example of a nonlinear feature projection technique is the hybrid nonlinear-discriminant analysis feature projection technique, which uses a two-layer projection technique to perform the nonlinear mapping task [182]. This technique is computationally efficient and robust, and it preserves the properties of the original model. Interpretable dimensionality reduction is a recent development in nonlinear feature projection that addresses the lack of interpretability in nonlinear dimensionality reduction techniques due to the absence of source features in low-dimensional embedding space [184].

One of the most recognized techniques within the projection of nonlinear characteristics is the t-SNE, which is detailed below:

- **t-SNE:** It is based on preserving the data's local structure while mapping it to a lower-dimensional space. t-SNE achieves this by modeling pairwise similarities between data points in high- and low-dimensional spaces. This technique focuses on preserving relationships between points in the original high-dimensional space when

projecting them into low-dimensional space (usually 2D or 3D) so that similar points in the actual space are close in the confined space and the least similar points are farther away. Also, it is particularly useful in cases where the data has complex and nonlinear relationships that linear techniques cannot easily capture. Also, it is widely used in various fields, including image classification, anomaly detection, and molecular structure classification [185].

One of the applications of t-SNE is in anomaly detection. In a study on catenary abnormal status detection [186], this technique was used to reduce the dimensionality of catenary data, and an improved gray wolf algorithm was applied to optimize the least squares support vector machine. The combination of t-SNE and the optimization algorithm resulted in high accuracy and low operation time for detecting abnormal statuses in the catenary system.

According to [3], the main steps to perform a t-SNE in a dataset are three. The first step involves calculating the similarities between points in the high-dimensional space. Gaussian distributions are centered around each point, and the density under these distributions is measured for other points. This process results in a list of observed conditional probabilities. In the second step, a smaller dimensional space represents the data. Initially, the points are randomly distributed in this new space. Similarities between points in the newly created space are calculated using a Student's t-distribution instead of a Gaussian distribution. This step also generates a list of probabilities. The third step aims to accurately represent the points in the lower-dimensional space by ensuring the similarity measures align. The similarities of points in both spaces are compared using the Kullback-Leibler (KL) divergence measure. Gradient descent is then used to minimize the KL divergence and find the best positions for the points in the lower-dimensional space. This minimization process reduces the difference between the probability distributions in the original space and the lower-dimensional space.

Finally, in Fig. 2.11, we can see how the data used as an example is grouped quite well when reducing its dimensionality. However, those that differ greatly from each class are far from these groups.

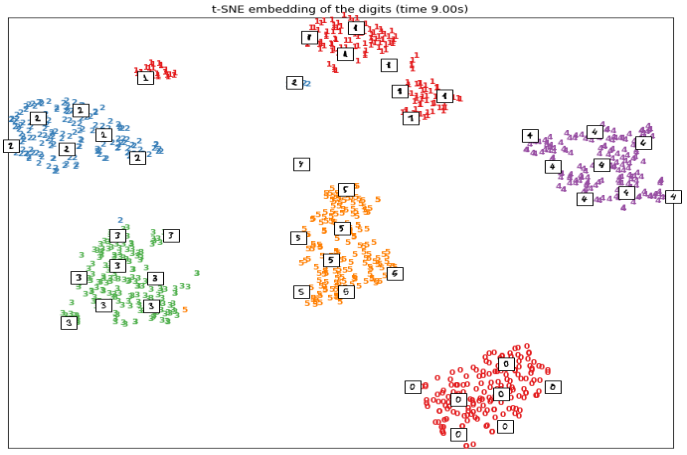


Figure 2.11: Visualization example of a dimensionality reduction using t-SNE [3]

2.9 Summary

In this chapter, several concepts of Machine Learning and Cybersecurity were analyzed. The types of ML were addressed, such as Supervised, Unsupervised, and Semi-supervised Learning, in addition to analyzing different models of each of these types. On the other hand, topics such as the types of security, the security triad, the types of cyber attacks, and some examples of each were addressed, as well as the types of cybersecurity and methods such as the IDS and the IPS capable of mitigating attacks that were mentioned. Anomaly Detection topics based on ML, such as OC-SVM and Isolation Forest, were also addressed, whose model we will use for the methodology and results of this work. In addition, the different types and techniques of linear and non-linear dimensionality reduction were mentioned. Still, we especially focused on PCA and t-SNE, which we will use later in conjunction with the Isolation Forest. All the concepts discussed above are essential and are the basis for the development of our algorithm. These concepts will be used to obtain the desired results that meet the objectives of this work.

Chapter 3

State of the Art

This section presents ML and DL works related to detecting zero-day attacks through an IDS. We discuss several approaches that use SIDS and AIDS to achieve their goals. In addition, we compare different techniques and data sets used to detect zero-day attacks.

Zero-day attacks pose a significant threat to computer systems [10, 187]. Traditional Signature-based Intrusion Detection Systems (SIDS) are ineffective against these attacks, which has led to the development of new approaches [188]. One such approach is Anomaly-based Intrusion Detection Systems (AIDS), which can use ML techniques to identify patterns of normal behavior and flag any deviations from those patterns [10]. Another approach is behavior-based detection, which monitors system behavior and alerts administrators to any anomalous actions [188]. Hybrid detection systems that combine multiple approaches have also been proposed. These systems use techniques, such as signature-based detection, anomaly-based detection, and behavior-based detection, to improve detection accuracy [10, 188]. Table 3.1 shows some approaches to detect zero-day attacks using ML, DL, and Transfer Learning (TL) techniques based on anomalies and SIDS, as well as approaches that span various AI models and traditional methods such as using Snort, which is an open-source network intrusion detection and prevention system software [189]. It is worth noting that this table will be referenced throughout the chapter to discuss each approach in detail.

Table 3.1: Comparison Of Detection Methods Used For Zero-day Attacks Detection Based on AIDS and SIDS

Approach	***Year***	***SIDS***	***AIDS***	***Detection Method***	***Algorithm / Technique***	***Dataset***	***Performance***
Sarhan, M., et al. [190]	2023		X	ML	ZSL	CICIDS18	Accuracy: 91.33%
Peppes, N., et al. [191]	2023		X	DL	GANs on DL Classifiers	Own dataset	Accuracy: 96.02%
Mbona and Eloff [192]	2022		X	ML	Benford's Law, One Class SVM (OCSVM)	CIC-DDOS2019, CIRA-CIC-DOHBRW-2020 IOT-INTRUSION2020	OCSVM: F_1 Score - 86%
Pitre et al. [193]	2022		X	ML	SVM - Decision Trees	IDS 2018 Intrusion CSVs	Accuracy: 95.1%
Ali et al. [194]	2022		X	AI	Comparative Evaluation	SPMD	Precision: 99.8%
Popoola, S., et al. [195]	2022		X	DL	Federated Deep Learning Federated Averaging	Bot-IoT	Accuracy: around 90.00%
Drozdenko, B., & Powell, M. [196]	2022		X	DL	CNN and LSTM	PCAP files	Accuracy: around 90.00%
Soltani et al. [197]	2021		X	DL	DOC, DOC++, OpenMax, AutoSVM	CICIDS2017, CSECICIDS2018	Accuracy: 98.00%
Bu, S. J., & Cho, S. B. [198]	2021		X	DL	First-order logic programmed	Own dataset	Recall: 97.00%
Zoppi, T., et al. [199]	2021		X	ML	Unsupervised Algorithms	SDN20	Recall: 99.40%
Hindy et al. [200]	2020		X	ML DL	Autoencoders	NSL-KDD CICIDS2017	Accuracy: 89–99% Accuracy: 75–98%
Sameera, N., & Shashi, M. [201]	2020		X	TL	Federated Deep Learning, Federated Averaging Algorithm	NSL-KDD, CIDD	Accuracy: 89–99%
Tang, R., et al. [202]	2020		X	ML	ZeroWall	Own dataset	F1-score: around 98.00%
Tang, R., et al. [203]	2020		X	ML	Unsupervised port-based	MAWI and UCSD Network Telescope	Around 90.00%
Wang H., et al. [204]	2020		X	ML	Microarchitectural Side-Channel	Own dataset	Accuracy: 98.00%
Zhou and Pezaros [205]	2019		X	ML	Classifiers	CIC-AWS-2018	Accuracy: 100%
Abri, F., et al. [206]	2019		X	Several	Several	malware security partner of Meraz'18	Detection Rate: 99.51%
Radhakrishnan, K., et al. [207]	2019		X	Several	Several	malware security partner of Meraz'18	Detection Rate: 99.51%
Zhisheng Hu., et al. [208]	2019		X	RL	Bayesian attack graphs	real-world DSMA scripts	Time: 1 to 10 s'
Kim J., et al. [209]	2018		X	DL	Deep Autoencoders	Microsoft Malware Classification	Accuracy: 95.74%
Jung, W., et al. [210]	2015		X	DL	GANs on DL Classifiers	CVE-2014-0515	Accuracy: around 90.00%
Holm [211]	2014	X		Snort	NA	NA	Detection Rate: 8.2%

3.1 Signature-based Intrusion Detection System (SIDS) Approaches

In [211], authors study the ability of signature-based network intrusion detection systems (SNIDS) to detect zero-day attacks. They test 356 severe attacks on SNIDS Snort, configured with official rules published before 1990. The article aims to validate the claim that signature-based network intrusion detection systems cannot detect zero-day attacks. As a result, the conservative estimate on zero-day detection by Snort is 8.2%. This result indicates the inability of SIDS to detect zero-day attacks, and it is better to look for other ways, such as AIDS, using ML and DP tools.

3.2 Anomaly-based Intrusion Detection System (AIDS) Approaches

3.2.1 AIDS - Machine Learning Approaches

In [192], the authors use a combination of supervised and unsupervised learning to detect anomalies in network traffic. They evaluate their approach on a real-world dataset and show that it outperforms traditional signature-based IDSs in detecting zero-day attacks. This research proposes the adoption of Benford's law, also known as the law of anomalous numbers, as a promising technique to identify significant network features that indicate abnormal behavior and facilitate the detection of zero-day attacks. Among the models tested, one-class support vector machines demonstrated the best performance, achieving a Matthews correlation coefficient of 74% and an F1 score of 85% in detecting zero-day network attacks. The results suggest that semi-supervised machine learning approaches can be a promising solution for detecting zero-day attacks in network security.

Another investigation [205], proposed an evaluation of different machine learning classifiers for detecting zero-day intrusion attacks using the CIC-AWS-2018 dataset, which contains network traffic data from various sources. The authors compare the performance of five classifiers, including decision tree, random forest, k-nearest neighbors, support vector machine, and neural network, in terms of accuracy, precision, recall, and F1-score. They also conduct experiments to test the robustness of the classifiers against attacks that try to evade detection. The results show that the neural network classifier outperforms the other classifiers in terms of accuracy, precision, recall, and F1-score. Also find that the classifiers are able to detect zero-day attacks with high accuracy, even when the attacks are modified to evade detection.

On the other hand, sometimes, some models can trigger false positive alarms, which can be out of place in being forewarned of something that is not happening, however for this type of case, in the article [193], the authors present an innovative solution to address the challenge of detecting zero-day attacks through the use of an Intrusion Detection System (IDS). They specifically focus on reducing the occurrence of false positives, which are alarm triggers that mistakenly identify benign activities as malicious attacks. Existing IDS systems often suffer from this issue, resulting in inefficient and costly security measures. To

tackle this problem, the authors propose a novel approach that combines machine learning algorithms with statistical techniques, forming a hybrid system. This integration aims to enhance the accuracy of zero-day attack detection while minimizing false alarms. The effectiveness of their approach is evaluated through several metrics, including the true positive rate, false positive rate, and detection time. The evaluation results demonstrate the proposed approach's superiority over existing IDS systems, exhibiting higher accuracy with a reduced false positive rate and an increased true positive rate for zero-day attack detection.

3.2.2 AIDS - Deep Learning Approaches

Machine learning and Deep learning techniques have been used to create the Intrusion Detection System (IDS). New cyber-attacks are increasing in number and variety which poses a significant challenge to IDS solutions. As a result, there is a growing need in the industry for robust IDSs capable of flagging zero-day attacks. Current bias-based zero-day detection studies have a high false-negative rate, which limits their practical application and performance.

The article [200] presents an implementation of an autoencoder to Detect zero-day attacks. The goal is to create an IDS model with high memory and reduce the error rate (false negative) to an acceptable minimum. They compared their results with a one-stage Support Vector Machine (SVM). The proposed model greatly benefits from the encoding and decoding capabilities of autoencoders. In addition, they use the two well-known IDS datasets for evaluation: CICIDS2017 and NSL-KDD. On the other hand, authors in [194] proposed a comparative evaluation of AI-based techniques for detecting zero-day attacks. The study evaluates the effectiveness of various AI-based techniques, including machine learning, deep learning, and fuzzy logic, in detecting zero-day attacks in a simulated environment. The evaluation criteria include accuracy, detection rate, false positive rate, and processing time. This investigation shows that deep learning algorithms, particularly convolutional neural networks, outperform other evaluated techniques regarding accuracy and detection rate. However, these algorithms also have higher false positive rates and processing times compared to other techniques such as fuzzy logic. This study suggests that a combination of these techniques may provide a more effective approach to detecting zero-day attacks, and further research is needed to improve the accuracy and efficiency of these systems. Overall, the study highlights the potential of AI-based techniques for detecting zero-day attacks and provides insights into the strengths and limitations of different approaches.

In the same way and under the same approach, the authors in [197] present an adaptable deep learning-based Intrusion Detection System (IDS) to detect zero-day attacks. The proposed system uses deep learning algorithms to learn patterns and anomalies in network traffic and can adapt to changing attack scenarios. The IDS system consists of two parts: feature extraction and classification. The feature extraction part uses Convolutional Neural Networks (CNN) to extract features from network traffic data. The classification part uses a Recurrent Neural Network (RNN) to classify the network traffic as normal or malicious. The system also includes an adaptive learning mechanism that automatically updates the model to learn new attack patterns. The study evaluates the effectiveness of the proposed IDS system using the CICIDS2017 and CSECICIDS2018 dataset, which contains a wide

range of attack scenarios. The results show that the proposed system achieves a high detection rate for known and unknown attacks, outperforming traditional IDS systems and other deep learning-based methods. The system's adaptability allows it to detect new zero-day attacks without requiring manual intervention or extensive retraining.

On the other hand, in [212] discusses the use of unsupervised machine learning techniques for detecting zero-day fast-spreading attacks and botnets in network intrusion detection systems. The authors explain that zero-day fast-spreading attacks and botnets are major threats to network security because they can spread rapidly and cause significant damage before they are detected. Traditional signature-based intrusion detection systems are not effective against these types of attacks because they rely on predefined rules and patterns. To address this problem, the authors propose using unsupervised machine learning techniques to detect anomalies in network traffic that may indicate the presence of zero-day fast-spreading attacks and botnets. They focus on two unsupervised techniques: k-means clustering and autoencoder-based anomaly detection. The authors evaluate the performance of their approach on real-world datasets, DARPA and ISCX, and compare it to a traditional signature-based intrusion detection system. The results show that the unsupervised machine learning approach achieved a much higher detection rate for zero-day fast-spreading attacks and botnets than the signature-based system.

3.3 ML Zero-day Attacks Detection Approaches

According to [213], provides a comprehensive review of machine learning-based zero-day attack detection approaches, identifies current limitations, and discusses future directions for research in this field. The contribution of this paper lies in its thorough analysis of current approaches to zero-day attack detection and in its identification of the limitations of these approaches. Additionally, the author offers a detailed insight into future directions for research in this field and suggests ways to enhance the efficacy of zero-day attack detection using machine learning techniques. Overall, this work is significant because it improves the understanding and ability to detect zero-day attacks, which are a significant cybersecurity threat today.

Other authors in [206] examine the effectiveness of machine/deep learning classifiers in detecting zero-day malware; the study evaluates the performance of various machine learning algorithms in detecting zero-day malware, including decision tree, k-nearest neighbor, random forest, and support vector machine. The results show that these algorithms can achieve high accuracy in detecting zero-day malware, with support vector machines achieving the highest accuracy among the tested algorithms. Overall, the study suggests that machine/deep learning classifiers can effectively detect zero-day malware and provide a valuable addition to existing malware detection systems. However, the authors note that the effectiveness of these classifiers can be impacted by factors such as the quality of the training data, the selection of features, and the choice of algorithms, and further research is needed to improve the accuracy and effectiveness of these systems.

Also, in [199], present a strategy for using unsupervised algorithms to detect zero-day attacks. The proposed strategy involves using unsupervised machine learning algorithms to analyze system behavior and detect anomalies that may indicate a zero-day attack. The study evaluates the effectiveness of various unsupervised algorithms, including clustering,

principal component analysis, and autoencoder, in detecting zero-day attacks in a simulated environment. The results show that the proposed strategy can effectively detect zero-day attacks with high accuracy, and the autoencoder algorithm outperforms the other evaluated algorithms. The study also discusses the limitations of the proposed approach, such as the need for high-quality data and the potential for false positives.

On the other hand, the study [195], proposes a federated deep learning approach for detecting zero-day botnet attacks in IoT-edge devices. The proposed approach uses deep learning algorithms to analyze network traffic data and detect anomalous behavior that may indicate a botnet attack. The study focuses on IoT-edge devices, which have limited resources and processing power, and uses a federated learning approach to overcome these limitations by distributing the learning process among multiple devices. The results show that the proposed approach can effectively detect zero-day botnet attacks with high accuracy, even with noise and limited data. The study also indicates that the federated learning approach can improve the performance of deep learning algorithms and reduce the computational burden on individual devices.

Moreover, in another article [194], the authors proposed a comparative evaluation of AI-based techniques for detecting zero-day attacks. The study evaluates the effectiveness of various AI-based techniques, including machine learning, deep learning, and fuzzy logic, in detecting zero-day attacks in a simulated environment with three real-world datasets consisting of 222,541 URLs. Finally, this investigation shows that deep learning algorithms, particularly CNN, outperform other evaluated techniques regarding accuracy and detection rate. However, these algorithms also have higher false positive rates and processing times compared to other techniques such as fuzzy logic. The study suggests that combining these techniques may provide a more effective approach to detecting zero-day attacks, and further research is needed to improve the accuracy and efficiency of these systems.

In another approach, an article proposed by [201], using deep transductive transfer learning consists of three main components: a feature extractor, a transductive transfer learning module, and a classifier. The feature extractor is a deep convolutional neural network trained to learn high-quality representations of input data, which in this case are network traffic flows. The feature extractor is trained on a pre-labeled dataset, and its task is to extract relevant features from traffic flows. The transductive transfer learning module uses pre-trained models on related tasks to improve the accuracy of zero-day detection. This module transfers the knowledge learned from related tasks and applies it to the task of zero-day attack detection. To do this, a transductive transfer algorithm is used based on the analysis of the similarity between labeled and unlabeled traffic flows. The SVM classifier uses the representations learned by the feature extractor and the transductive transfer learning module to classify traffic flows as normal or malicious. The classifier is trained on a labeled dataset and used for real-time detection of zero-day attacks.

On the other hand, another approach [198], proposes a novel approach for detecting zero-day phishing attacks by integrating deep learning with first-order logic programmed constraints. The proposed method uses deep learning algorithms to analyze email content and detect suspicious patterns that may indicate a phishing attack. The study combines this with first-order logic programmed constraints that encode knowledge about the characteristics of phishing emails to improve the accuracy of the detection system. The results show that the proposed approach can effectively detect zero-day phishing attacks with high accuracy, outperforming traditional signature-based detection systems. The study

also shows that the first-order logic programmed constraints can improve the interpretability of the deep learning algorithms and provide insights into the characteristics of phishing emails.

An article [208], suggests a reinforcement learning method as a way to develop an adaptive cyber defense against zero-day attacks. These attacks can bypass traditional detection systems that rely on recognizing specific signatures. The proposed method uses reinforcement learning algorithms to develop a defense strategy that can effectively identify and respond to zero-day attacks. The study evaluates the effectiveness of this approach in a simulated environment and compares it with traditional signature-based detection systems and static defense methods. The findings indicate that the proposed approach successfully detects and responds to zero-day attacks with high accuracy, outperforming traditional detection systems and static defense methods. Furthermore, the reinforcement learning approach can adjust to variations in the attack environment and enhance the overall security of computer systems. Therefore, the study proposes that reinforcement learning can be a useful technique in developing adaptive cyber defense against zero-day attacks and enhancing security.

Another approach [202] proposed a method for detecting zero-day web attacks using encoder-decoder recurrent neural networks, called Zerowall. Zero-day attacks refer to attacks that exploit unknown vulnerabilities in web applications, making them difficult to detect using traditional signature-based methods. The proposed method involves using a recurrent neural network to encode the input data and a decoder network to reconstruct the original data, while also detecting any malicious input data. The study evaluates the effectiveness of the Zerowall approach in detecting zero-day web attacks by comparing it to several traditional and deep learning-based detection methods. The results show that the proposed method can effectively detect zero-day web attacks with high accuracy, outperforming other detection techniques. The study also demonstrates that the Zerowall approach can identify new types of zero-day attacks that were not present in the training data, making it a promising tool for enhancing the security of web applications.

In the same way, in [204] created a multi-phase machine learning framework called Phased-guard, which aims to detect and identify zero-day microarchitectural side-channel attacks. These attacks exploit vulnerabilities in hardware systems, making them difficult to detect using traditional software-based methods. The proposed framework uses multiple machine learning models in a phased approach to detect and classify different types of side-channel attacks. The study evaluates the effectiveness of the Phased-guard framework by comparing it with other machine learning-based approaches in a simulated environment. The results show that the proposed framework can effectively detect and identify different types of zero-day microarchitectural side-channel attacks with high accuracy, outperforming other detection methods. The study also demonstrates that the phased approach of the framework can improve the overall efficiency and effectiveness of the detection system. Overall, the study suggests that the Phased-guard framework can be a valuable tool for enhancing the security of hardware systems against zero-day microarchitectural side-channel attacks.

In [190], proposes a novel model for zero-day attack detection based on zero-shot machine learning. The authors argue that zero-day attacks are becoming more prevalent and sophisticated, and traditional signature-based detection methods are no longer sufficient. To address this challenge, the authors propose a zero-shot machine learning approach that

can detect new and unknown attacks without requiring any prior training on those specific attack types. The proposed model involves learning a generalized model to classify known and unknown attacks based on their underlying features. The authors use a combination of convolutional neural networks (CNNs) and long short-term memory (LSTM) networks to extract high-level features from network traffic data and then apply a zero-shot learning algorithm to train the model to recognize new and unknown attacks based on their similarities to known attacks. To evaluate the performance of the proposed model, the authors conduct experiments on several real-world datasets.

The article [191] explores the effectiveness of using Generative Adversarial Networks (GANs) to generate data samples for zero-day attacks on Deep Learning (DL) classifiers. The authors first explain that zero-day attacks are cyber-attacks that exploit vulnerabilities in computer systems that are not yet known to the public or software developers. As a result, they cannot be prevented by traditional security solutions such as antivirus software or firewalls. DL classifiers are increasingly used to detect and prevent zero-day attacks, but they require large amounts of data to be trained effectively. To overcome this problem, the authors propose using GANs to generate synthetic data samples that mimic the behavior of zero-day attacks. They then compare the effectiveness of DL classifiers trained on real and artificial data samples using various evaluation metrics such as accuracy, precision, recall, and F1 score. The results of their experiments indicate that DL classifiers trained on both real and synthetic data samples perform similarly, with no significant differences in accuracy or other evaluation metrics.

The authors in [210] explain that zero-day flash malware refers to malware that exploits previously unknown vulnerabilities in Adobe Flash Player, which is widely used in web browsers. Traditional signature-based antivirus software cannot detect such malware because they do not have prior knowledge of the malware's characteristics. To overcome this problem, the authors proposed using a deep learning algorithm called a Convolutional Neural Network (CNN) to detect zero-day flash malware. They trained the CNN on a dataset of malicious and benign Flash objects and used it to classify new Flash objects as malicious or benign. The authors evaluated their approach's performance on a real-world zero-day flash malware dataset and compared it to traditional signature-based antivirus software. The results showed that their approach using deep learning achieved a much higher detection rate for zero-day flash malware than traditional antivirus software.

On the other hand, in [209] proposes a new approach to detecting zero-day malware using transferred generative adversarial networks (GANs) based on deep autoencoders. The authors begin by discussing the challenges of detecting zero-day malware, which can bypass traditional signature-based detection methods. They then propose their approach, which involves training a GAN on a large dataset of normal files and using the learned representation to detect anomalies in new files. The authors detail their approach, which involves training the GAN on a source domain of normal files and transferring the learned representation to a target domain of potentially malicious files. They also discuss the use of deep autoencoders to learn a compressed representation of the files, which helps to reduce the dimensionality of the input data and improve the effectiveness of the GAN. The authors evaluate their approach in their dataset and compare it to other state-of-the-art methods for zero-day malware detection. They find that their approach achieves high accuracy and sometimes outperforms other methods.

Another approach [196] discusses the use of deep learning techniques for detecting

zero-day exploits in network traffic flows. The article begins by discussing the challenges of detecting zero-day exploits, including the need for timely and accurate detection to prevent potential attacks. The authors propose using deep learning techniques, specifically Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, to detect these types of exploits in network traffic. The authors detail their approach, which involves preprocessing the network traffic data and using the CNN and LSTM networks to learn patterns in the data indicative of zero-day exploits. They also discuss transfer learning, which involves using pre-trained models on related tasks to improve the detection accuracy. The authors evaluate their approach on a dataset of network traffic flows and compare it to other state-of-the-art methods for zero-day exploit detection. They find that their approach achieves high accuracy and sometimes outperforms other methods.

The authors in [207] provide an overview of zero-day malware attacks and various techniques for their detection. The authors review the various types of zero-day attacks, including those that exploit software vulnerabilities, use social engineering techniques, and leverage advanced persistent threats (APTs). It also provides an overview of the existing detection techniques for zero-day attacks, including signature-based methods, anomaly detection, and behavior-based detection. They discuss the limitations of each approach and highlight the importance of combining multiple detection methods for improved accuracy. The article then delves into the machine-learning techniques used for zero-day malware detection, including supervised, unsupervised, and hybrid approaches. The authors also discuss the challenges and limitations of using machine learning techniques for detection. Finally, the authors review the current state of the art in zero-day malware detection and discuss future research directions, including artificial intelligence and deep learning techniques.

The last approach proposed in [203] focuses on detecting zero-day attacks by analyzing the traffic generated by network services running on different ports. The methodology comprises three main stages: port selection, feature extraction, and clustering. In the first stage, a subset of ports is selected based on their relevance for network security. The authors use two criteria for port selection: the frequency of use and known vulnerabilities associated with the service running on the port. The selected ports are monitored for traffic analysis. In the second stage, traffic features are extracted from the network traffic generated by the selected ports. The authors propose to use the number of bytes sent and received, the number of packets sent and received, and the time between packets as traffic features. In the third stage, clustering is performed on the extracted traffic features to identify anomalous traffic patterns indicative of zero-day attacks. The authors use a hierarchical clustering algorithm to group traffic flows based on their similarity. The resulting clusters are analyzed for anomalous patterns that could indicate zero-day attacks. The authors evaluated their methodology using a dataset containing different types of attacks, including zero-day attacks.

3.4 Related Datasets

Below are some different datasets used in the works presented in this section. These data sets detect zero-day attacks by recognizing anomalies in their behavior; however, they do not have data specifically labeled with this type of attack since, as mentioned above, they

are attacks with no record. Table 3.2 details each of these datasets in a summarized manner for better understanding.

Table 3.2: Summary of datasets used in the works presented in this section

Name	***Year***	***Origen***	***Features***	***N. registers***
CIC-AWS-2018	2018	Amazon Web Services (AWS)	Benign traffic and 14 types of intrusions	NA
CICIDS2017	2017	Various network environments	NA	3'119.345
UNSW-NB15	2015	University of New South Wales	49	2'540.044
NSL-KDD	2009	Derived from KDD-CUP99	NA	NA
KDD-CUP99	1999	DARPA'98 IDS evaluation program	41	4'900.000 (training) 2'000.000 (testing)

3.4.1 KDD-CUP99:

The KDD-CUP99 dataset [214] has been commonly used to develop methods to detect anomalies. This dataset is based on data collected during the DARPA'98 IDS evaluation program. DARPA'98 comprises approximately 4 gigabytes of compressed raw tcpdump data, representing seven weeks of network traffic. This data can be processed to generate about 5 million connection records, each containing about 100 bytes. The test data for two weeks consists of approximately 2 million connection records. The KDD training dataset includes approximately 4'900.000 unique connection vectors, each with 41 features. These vectors are labeled as normal or attack, with a specific type of attack [215].

3.4.2 NSL-KDD:

This dataset [216] is derived from the KDD-CUP99 dataset and is widely used for network intrusion detection research. It consists of network traffic data captured in a controlled environment, encompassing normal and various attack traffic types. This improved some problems that the KDD-CUP99 had since it had very redundant data that could cause very skewed results, as well as having similar instances, which could cause the model to generate a bit of noise and unnecessary complexity at the moment of the analysis [217]. The major disadvantage of NSL-KDD is that it does not represent the modern low-footprint attack scenarios [218]

3.4.3 UNSW-NB15:

This dataset contains real-world network traffic logs captured from a college campus network environment and was created via virtual network testbeds representing modern network structures [190]. This dataset contains about 49 features, grouped into six groups: Basic Features, Flow Features, Time Features, Content Features, Additional Generated Features, and Labeled Features [219]. Finally, this dataset's total number of records is 2'540.044, which were stored in four CSV files [220]. However, there are two sets of CSV files, both for training and for testing, for the multiple models that are to be developed and need to be used.

3.4.4 CICIDS-2017:

This data set contains traffic logs collected from a realistic and diverse network environment. According to the author of CICIDS-2017 [221], the dataset spanned over eight files containing five days of normal and attack traffic data of the Canadian Institute of Cybersecurity. This dataset is presently scattered in 8 files. Combining these eight files into one will get a total of 3'119.345 instances. Because of this, it is observed that the combined dataset contains data from all possible recent attack tags in one place, causing problems in processing them within a model.

3.4.5 CIC-AWS-2018:

This dataset is collected from the Amazon Web Services (AWS) cloud environment and is designed to evaluate intrusion detection systems' performance in cloud-based scenarios. This containing benign traffic and fourteen different types of intrusions are used as training datasets, eight different types of intrusions traffic data collected online and benign traffic data from our research production network are used as testing datasets [205].

Chapter 4

Methodology

This section develops a zero-day attack detection model using Machine Learning and reduction techniques, specifically using a combination of PCA or t-SNE and the Isolation Forest (IF) (anomaly-focused, unsupervised learning model). First, the data acquisition process will be described, how data are distributed, and each numerical or categorical feature will be detailed. Then, the t-SNE will be carried out on these data, and finally, based on this method, we can use the IF to detect anomalies.

4.1 Data Acquisition

To develop attack detection models, specifically IDS or AIDS, some researchers use various datasets, such as those described in Table 4.1 These data sets help researchers to be able to detect malware, simple attacks, and even zero-day attacks. However, the latter is done by recognizing anomalies in the behavior of the data; that is, researchers rely on these datasets, making qualitative recognitions. For example, when many packets are being sent from an IP address, and nothing is being received in the arrival direction, another case may be when a connection is open for too long, and nothing is being sent receiving information packets. For this reason, this work is usually very exhaustive and requires a lot of knowledge on the subject. Finally, to complement, this type of work is complicated since the datasets do not contain data specifically labeled for the type of attack we are investigating since, as mentioned above, they are attacks without any registration.

4.1.1 Chosen Dataset: UNSW-NB15

We have chosen the UNSW-NB15 dataset because it was generated in a real environment and based on modern network structures with current and varied attack flows. It also gives us a large amount of generated data and another amount of pre-processed data for training and testing, where you can have clean data and each correctly labeled. Below, we provide a more detailed description of the UNSW-NB15 dataset and its relevance to our work.

Table 4.1: Existing attacks within the Datasets Used For The Detection Of Zero-day Attacks

Name	***Year***	***Types of Attacks It Contains***
CIC-AWS-2018	2018	DoS, Probe, R2L, U2R
CICIDS2017	2017	DoS, Probe, R2L, U2R, Reconnaissance
UNSW-NB15	2015	DoS, Probe, R2L, U2R, Reconnaissance
NSL-KDD	2009	DoS, Probe, R2L, U2R
KDD-CUP99	1999	DoS, Probe, R2L, U2R

Network Traffic Available in UNSW-NB15

To achieve this thesis's objective, we will use pre-classified and pre-processed training and testing data for use in models; these two sets have 82332 rows and 175341 for training and testing correspondingly, with 45 features of the original 49. In the data, each of the rows of the CSV file refers to a network traffic, among them are the benign ones categorized as normal and the malignant ones in which each type of attack is observed. The percentage of the 2 types of network traffic available in the training dataset is shown in Fig. 4.1, where the blue section shows us the benign flow and the red section shows us the malignant flow, that is, flows where the different types of attacks are found. It is worth mentioning that the malignant flow predominates within our dataset with a percentage greater than 50%.

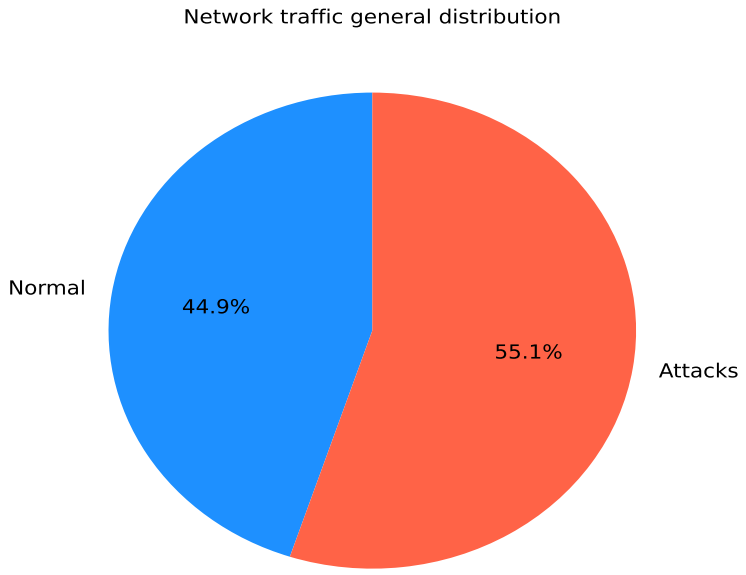


Figure 4.1: Network traffic general distribution observed in the UNSW-NB15 dataset

To explain in more detail the existing attacks of our dataset compared to the benign flow, Fig. 4.2 shows the percentage of each type of attack with the percentage of benign flow.

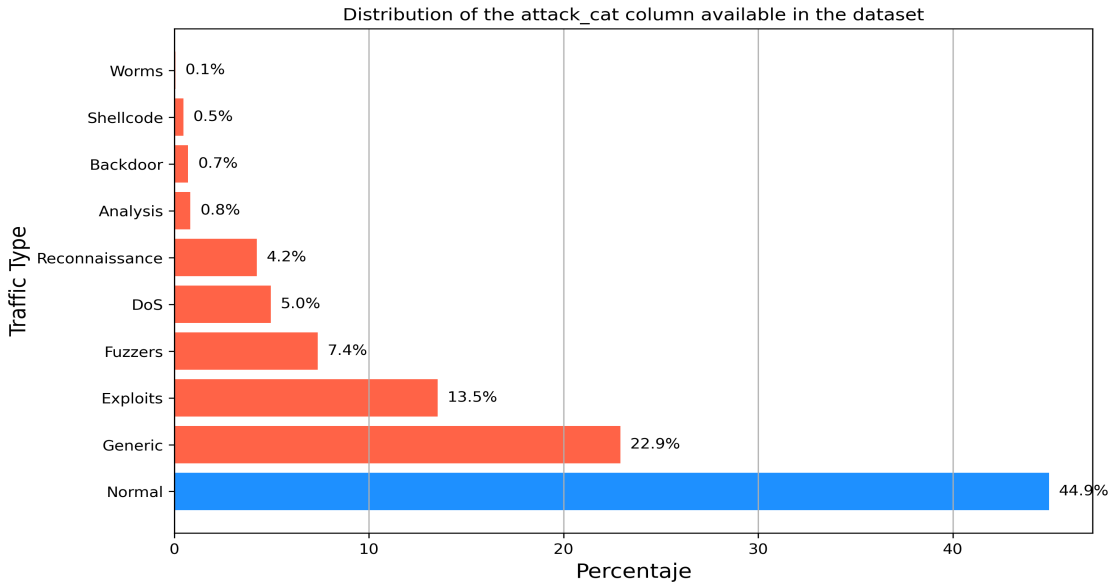


Figure 4.2: Distribution of traffic types within the UNSW-NB15 dataset; normal traffic and traffic with attacks

Taking as reference Fig. 4.2, we have nine different types of attacks, which are described below:

- **Worms:** Attacks that use malware to exploit system vulnerabilities and spread malware through networks and connected devices.
- **Shellcode:** This Is a small piece of code executed to exploit a software vulnerability, allowing the attacker to take control of the target computer [222].
- **Backdoor:** It refers to a covert technique used to circumvent standard authentication or security measures, enabling unauthorized entry into a computer system. Malicious actors frequently employ this method to establish remote access to a computer while evading detection during casual scrutiny [223].
- **Analysis:** It contains different attacks of port scan, spam, and HTML file penetrations [224].
- **Reconnaissance:** These are employed to gather information about a target network or system. These attacks typically serve as the initial step in a larger attack, providing the attacker with valuable insights about the target that can be used to plan and execute a more sophisticated attack [225].
- **DoS:** This attack disrupts the normal operation of the target system by overwhelming it with excessive traffic or exploiting vulnerabilities to trigger a crash. When this happens, the different users will have a denial of said service [226].
- **Fuzzers:** Attempting to cause a program or network to be suspended by feeding it the randomly generated data [224].

- **Exploits:** Refers to a specific type of attack that takes advantage of a bug or system flaw [227].
- **Generic:** A technique works against all block ciphers (with a given block and key size), without consideration about the structure of the block-cipher [224].

Features Available in UNSW-NB15

This dataset has 49 features, Table. A1 shows a summary of these features; these can be numerical or categorical; they are also grouped into six different groups, which are explained in detail below:

- **Flow features** refers to the specific flow or network connection features in a data set. These are designed to provide detailed information about the behavior and properties of a data flow between two points in the network.
- **Basic features** refers to the fundamental features used to describe a network connection in a data set. These features are important attributes used to analyze and classify connections based on their behavior.
- **Content features** refers to the features and properties of the payload or content of network communications. These features provide insight into the data transmitted within a network connection, allowing for a deeper understanding of the nature of communication. Content features may include information such as specific keywords, patterns, or structures present in the payload data. Content functions play a crucial role in capturing the granular details of network communications, improving the effectiveness of network monitoring and security systems.
- **Additional generated features** refers to supplementary attributes derived from existing network traffic data. These features are created based on specific calculations or transformations applied to the original data. Also can include metrics such as connection counts, time-based statistics, and relationships between different network entities.
- **Labelled features** consists of attributes that provide labels or classifications for the network connections.

4.2 Feature Selection and Projection for Model

In this section, we will select features, which will be the respective data to enter our model so that it can be trained. Said model was mentioned as the Isolation Forest, which focuses on unsupervised learning based on anomalies.

We will focus this feature selection on dimension reduction techniques since our dataset, although it has little training data, contains a large number of both numeric and categorical features, as detailed in the previous subsections 2.8.

4.2.1 Exploratory Data Analysis

Since our dataset contains too many numerical and categorical features (see Table. A1), we can use dimensionality reduction techniques to better select and project our data before training the ML model. To remember a little, dimensionality reduction techniques are used to reduce the number of dimensions or features in a dataset while retaining as much information as possible. This is done to remove redundancy, improve the performance of learning algorithms, and make it easier to visualize the data [2]. First, the tools used to carry out this analysis will be explained, and then we will begin using the data projection techniques, starting with the PCA.

Used Tools in the Process

To perform the respective analysis on the chosen dataset, we will use Google Colab with the following specifications:

- Execution environment: Python3
- GPU accelerator: T4 GPU

In addition, it uses the Python language and the different modules or libraries that it provides and facilitates. The latter are described in Table 4.2.

Table 4.2: Modules, Libraries, and classes used in the methodology of this work

Module / Library	***Class***
pandas	-
matplotlib.pyplot	-
mpl_toolkits.mplot3d	Axes3D
numpy	-
sklearn.ensemble	IsolationForest
matplotlib.lines	Line2D
sklearn.manifold	TSNE
seaborn	
sklearn.decomposition	PCA
sklearn.preprocessing	StandardScaler

Before starting using the PCA, we first proceed to load our dataset from its official page[1] in the unsw_dataset variable, which we will be referring to in all our code, we will do this with the help of the panda library and in the following way:

```
1 import pandas as pd
2
3 # Load the data
```

[1]https://research.unsw.edu.au/projects/unsw-nb15-dataset

```
unsw_dataset = pd.read_csv('https://cloudstor.aarnet.edu.au/plus/s/2DhnLGDdEECo4ys/download?path=%2FUNSW-NB15%20-%20CSV%20Files%2Fa%20part%20of%20training%20and%20testing%20set&files=UNSW_NB15_training-set.csv')
```

Also, we will select the numeric variables of said dataset:

```
# Select the numeric columns
numeric_cols = ['dur', 'spkts', 'dpkts', 'sbytes', 'dbytes', 'sload', 'dload', 'smean', 'dmean', 'sloss', 'dloss', 'sinpkt', 'dinpkt', 'sjit', 'djit', 'swin', 'stcpb', 'dtcpb', 'dwin', 'tcprtt', 'synack', 'ackdat', 'trans_depth', 'response_body_len', 'ct_srv_src', 'ct_state_ttl', 'ct_dst_ltm', 'ct_src_dport_ltm', 'ct_dst_sport_ltm', 'ct_dst_src_ltm', 'ct_ftp_cmd', 'ct_flw_http_mthd', 'ct_src_ltm', 'ct_srv_dst']
```

After loading our dataset and assigning important variables to be used throughout our code, we will detail the process carried out with the PCA.

Preparation of the UNSW-NB15 Dataset for PCA:

PCA is a dimensionality reduction technique that projects the original data into a lower-dimensional feature space, preserving as much information as possible. Data standardization is one of the first steps to carry out a PCA within the selected data set, as shown in Fig. 2.9. Data standardization [228] is the process of transforming data to have a common scale or range. This is done to ensure that the data is comparable and can be analyzed together, and helps to remove any bias due to different scales of the features. Standardization involves subtracting the mean from the data and dividing by the standard deviation. This process results in data with:

- **A mean of zero:**the sum of the feature values is adjusted so that the average of all the features is zero.
- **Standard deviation of one:** This deviation measures how much they have spread the values around the means. Then, the standard deviation of all features is set to one.

The formula used for standardization is as follows: $z = \frac{x-\mu}{\sigma}$ where z is the standardized value, μ is the mean of the feature, x is the original value or observation to be standardized and σ is its standard deviation. It is important to note that the minimum value after standardization will not be exactly zero due to the subtraction of the mean operation. The maximum value is also reduced relative to the standard deviation.

On the other hand, in our code, the first thing to do is verify if our dataset is standardized.

```
# Check if the dataset is standardized
is_standardized = True

```

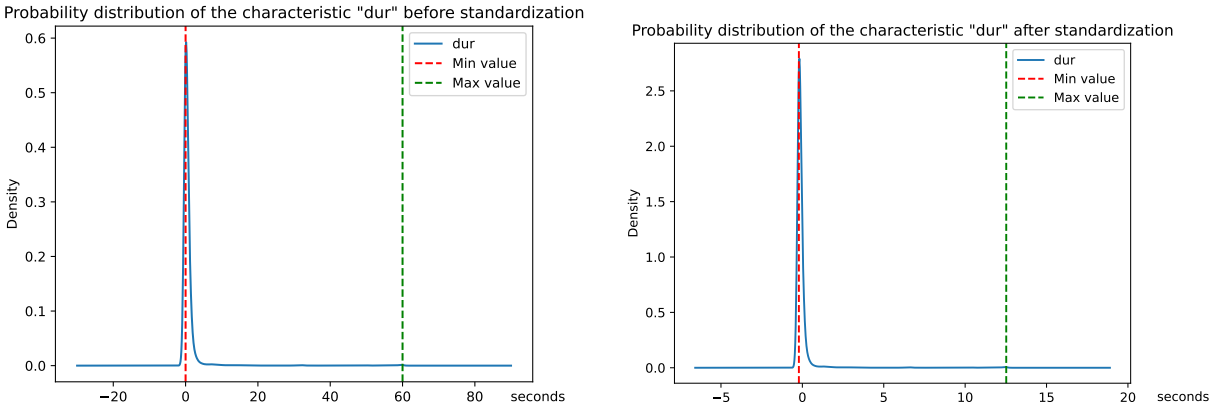
```
for col in numeric_cols:
    mean = unsw_dataset[col].mean()
    std = unsw_dataset[col].std()

    if abs(mean) > 0.001 or abs(std - 1) > 0.001:
        is_standardized = False
        break

if is_standardized:
    print("The dataset is standardized")
else:
    print("The dataset is not standardized") # The values do not have
        a mean of zero and do not have a standard deviation of one
```

As we can see, our dataset is not standardized; therefore, we will do it using the StandardScaler library.

```
from sklearn.preprocessing import StandardScaler

# We proceed to standardize
scaler = StandardScaler()
unsw_dataset[numeric_cols] = scaler.fit_transform(unsw_dataset[
    numeric_cols])

# Check if the dataset is standardized
is_standardized = True

for col in numeric_cols:
    mean = unsw_dataset[col].mean()
    std = unsw_dataset[col].std()

    if abs(mean) > 0.001 or abs(std - 1) > 0.001:
        is_standardized = False
        break

if is_standardized:
    print("The dataset is standardized") # The values have a mean of
        zero and a standard deviation of one
else:
    print("The dataset is not standardized")
```

Next, we look at a comparison of the standardized data with the unstandardized data using the duration feature (**dur**) as a sample via a density plot (see Fig. 4.3). A density plot shows the distribution of a numeric variable that uses a kernel density estimate to deliver the probability density function of the variable [229]. It was used because, before standardization (see Fig. 4.3a), it helps us know how the values of a feature are distributed.

It can be identified if the values tend to cluster in a specific range with multiple peaks or if the distribution is skewed; this is observed on the X-axis. However, the Y axis shows how the variable is distributed in terms of its density of probability; the higher the density is at a specific point on the Y axis, the greater the probability that the values of the variable are in that area. On the other hand, after standardization (see Fig. 4.3b), this type of graph will allow us to see how the distributions have been transformed and if they have acquired a more standard shape (with mean zero and standard deviation one).



(a) Before standardization

(b) After standardization

Figure 4.3: Probability distribution of the **dur** feature.

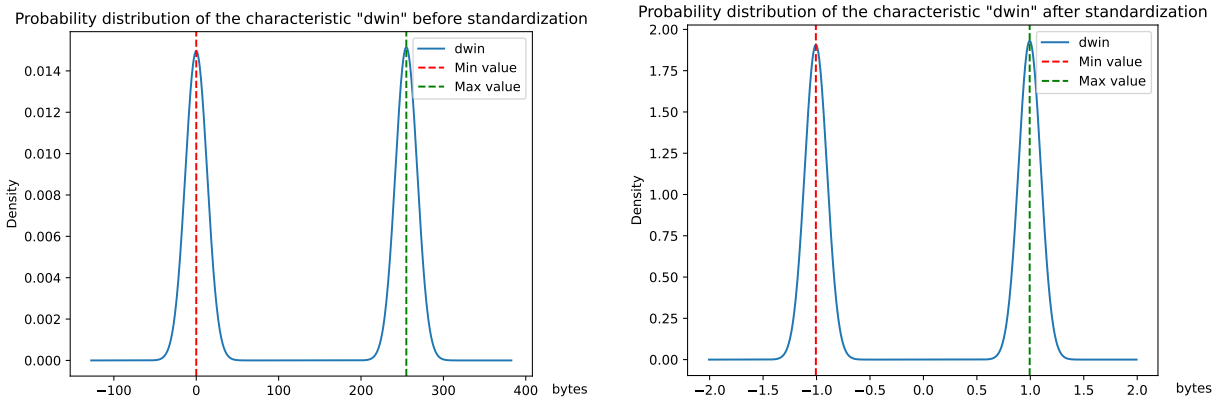
In Fig. 4.3a, it can be seen that the maximum value of this unstandardized feature is 60, and its minimum value is 0, in addition to the fact that most of the values are concentrated around this last point. However, values such as around 15 can be observed with a small peak or minimum concentration of values.

In Fig. 4.3b, it can be seen that the maximum value of this unstandardized feature is 12.52, and its minimum value is -0.21, in addition to the fact that most of the values are concentrated around this last point. However, values such as around one can be observed with a small peak or minimum concentration of values.

Another example may be that of the Destination window size (**dwin**) feature (see Fig. 4.4) or also called the Destination TCP window advertisement, which in the TCP (Transmission Control Protocol) field is an important component of the header of each segment that makes it up. Dwin represents the amount of data a receiver can accept before it needs the sender to stop and wait for more space to be freed in its receive buffer. It is a form of flow control to ensure that the sender does not overwhelm the receiver with more data than it can handle.

Fig. 4.4a shows the distribution probability of dwin data, where a minimum value of 0 and a maximum value of 255 are shown. At both points, the values are exact; they do not revolve around them but can only be the two values mentioned above. The value of 0 would mean that the receiver or the destination of the communication in TCP does not have space available in its receive buffer to accept more data at that moment. The receiver cannot receive more data and needs the sender to stop sending data temporarily. On the other hand, a value of 255 would mean the opposite, that the receiver has considerable space in its receive buffer available to receive data. On the other hand, in Fig. 4.4b, the

same graph of the dwin feature is observed, but with the standardized data; in this way, we observe a minimum value of -1 and a maximum value of 1.



(a) Before standardization

(b) After standardization

Figure 4.4: Probability distribution of the **dwin** feature.

Finally, in Fig. 4.5, the numerical features' density graphs are observed after standardizing their data.

PCA Application

After standardizing the data, we will apply PCA using the **sklearn** PCA library, which greatly assists us in all the subsequent steps after standardization that are shown in Fig. 2.9.

```
from sklearn.decomposition import PCA

pca = PCA()
pca.fit(unsw_dataset_standardized[numeric_cols])
principal_components = pca.transform(unsw_dataset_standardized[
    numeric_cols])

num_components = pca.n_components_
print("Number of main components:", num_components) #34
```

In this way, we obtain a total of 34 principal components; however, if we go back to what was said before about what the use of PCA means, we should have a lower value of principal components and not equal to that of the numerical features, then this could be telling us something about the position we occupy in terms of the relationship of our data.

To clearly visualize what it means that the number of principal components is equal to the number of numerical features, we will make a plot choosing the label feature. In Fig. 4.6, we have the PCA graph interpreted under the **label** feature; each drawn point represents a row of our dataset. Here, we can see how the existing classes in labels such as **Normal** and **Attack** overlap, thus demonstrating the non-linear nature of their data.

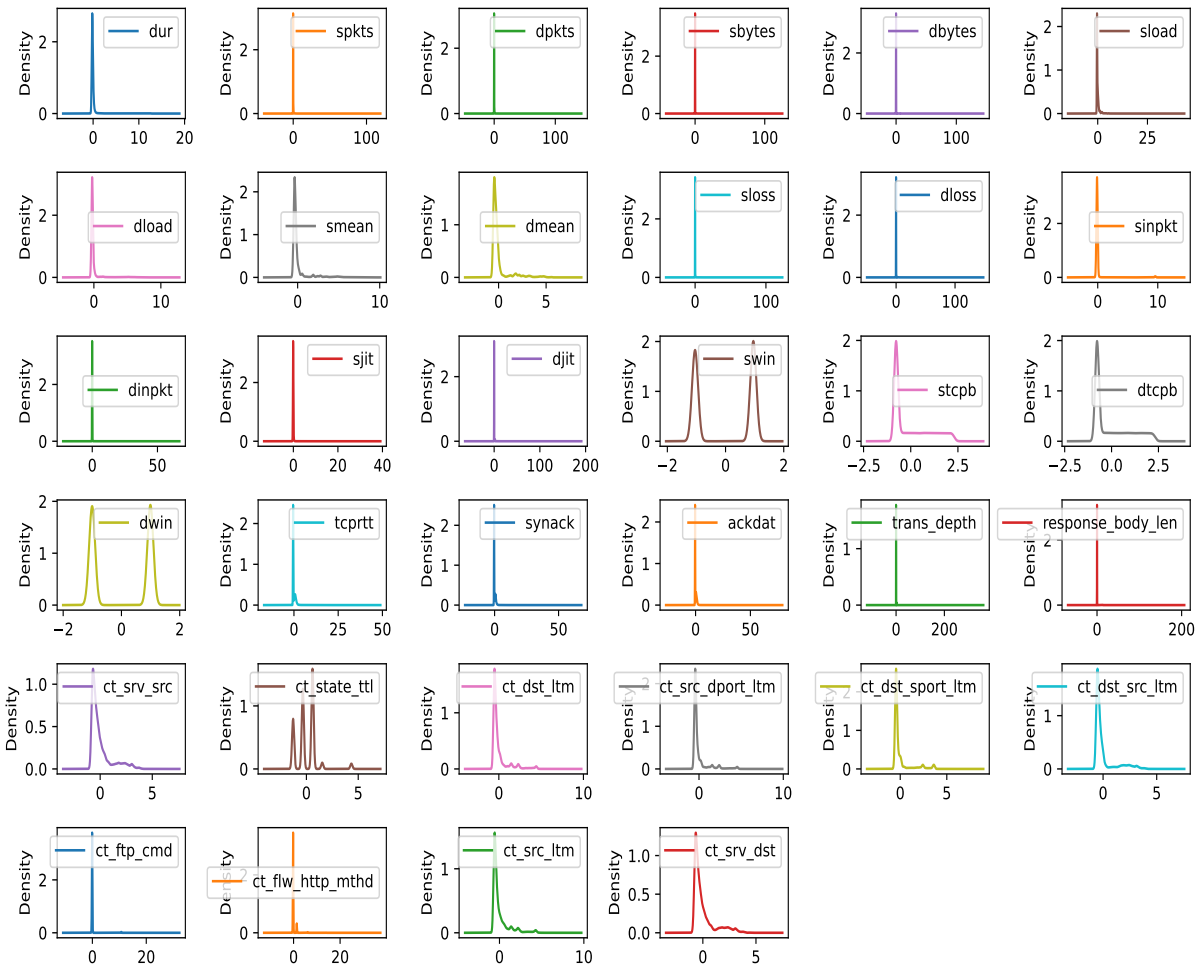


Figure 4.5: Probability distribution of all numeric features after standardization

Therefore, this investigation showed us that the UNSW-NB15 would not work with a linear feature projection algorithm such as PCA. To be sure of this hypothesis, in the next section, we will study the correlation of these data.

Data Linearity

As we observed in the previous section, we will study the linearity of the data from the UNSW-NB15 dataset due to the slightly inconsistent results obtained when applying PCA. To do this, we will use the Pearson correlation matrix, a method generally used as a primary check of the relationship between two variables. The correlation coefficient can be negative, positive, or zero and measures the strength of the linear relationship between two variables. Each type of correlation will be explained below:

- **Negative Correlation:** This means that when one variable increases, the other tends to decrease. The closer the value is to -1, the stronger the negative correlation.
- **Positive Correlation:** This means that when one variable increases, the other tends to increase. The closer the value is to 1, the stronger the positive correlation.

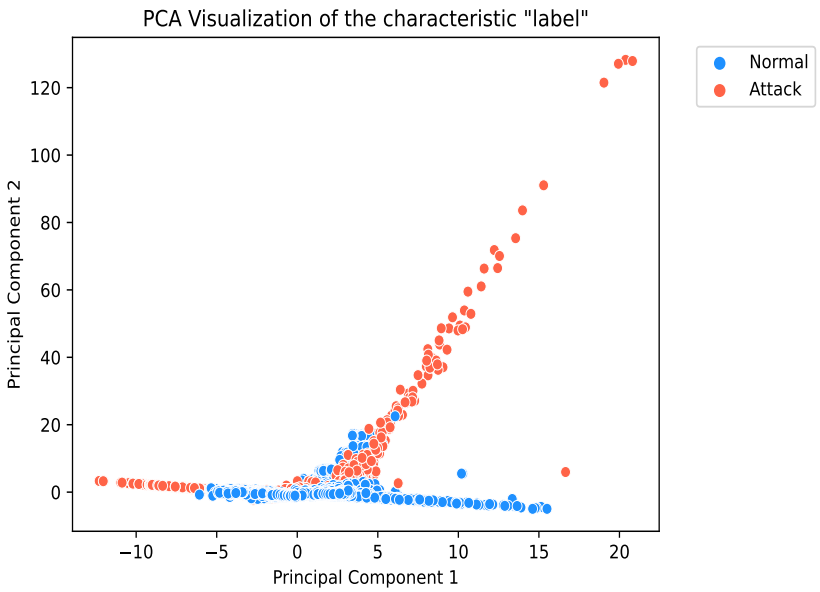


Figure 4.6: PCA applied to the data set of the study, for this case the **label** feature was used, the non-linearity of its data is observed

- **Zero Correlation:** This indicates no clear linear correlation between the two variables.

Fig. 4.7 shows the Pearson correlation matrix; in this, we can see how few linear correlations exist between data; although there are positive (red hue) and negative (blue hue) correlations, the existence of no correlation, that is, values equal to 0 (gray hue), predominates.

Finally, after the analysis is done, we can perform some small calculations to obtain the total percentage of existing correlation within the matrix. For this, we will add the absolute values of all the correlation coefficients and then divide this sum by the total number of values in the matrix, giving us a value of **19.01%**.

```
1 total_corr_sum = correlation_matrix.abs().sum().sum()
2 total_values = correlation_matrix.shape[0] * correlation_matrix.shape
      [1]
3 total_corr_percentage = (total_corr_sum / total_values) * 100
4 print(f'Total Correlation Percentage: {total_corr_percentage:.2f}%')
      //19.01%
```

The value obtained is a very small percentage compared to 100% of our dataset. This suggests a relatively low correlation between the features, indicating they are not strongly interconnected. Additionally, this might imply the absence of a clear linear structure within the data, potentially revealing non-linear or intricate relationships among the features. Consequently, we will explore the utilization of t-SNE as a dimensionality reduction technique to visualize the data and uncover potential concealed clusters or patterns.

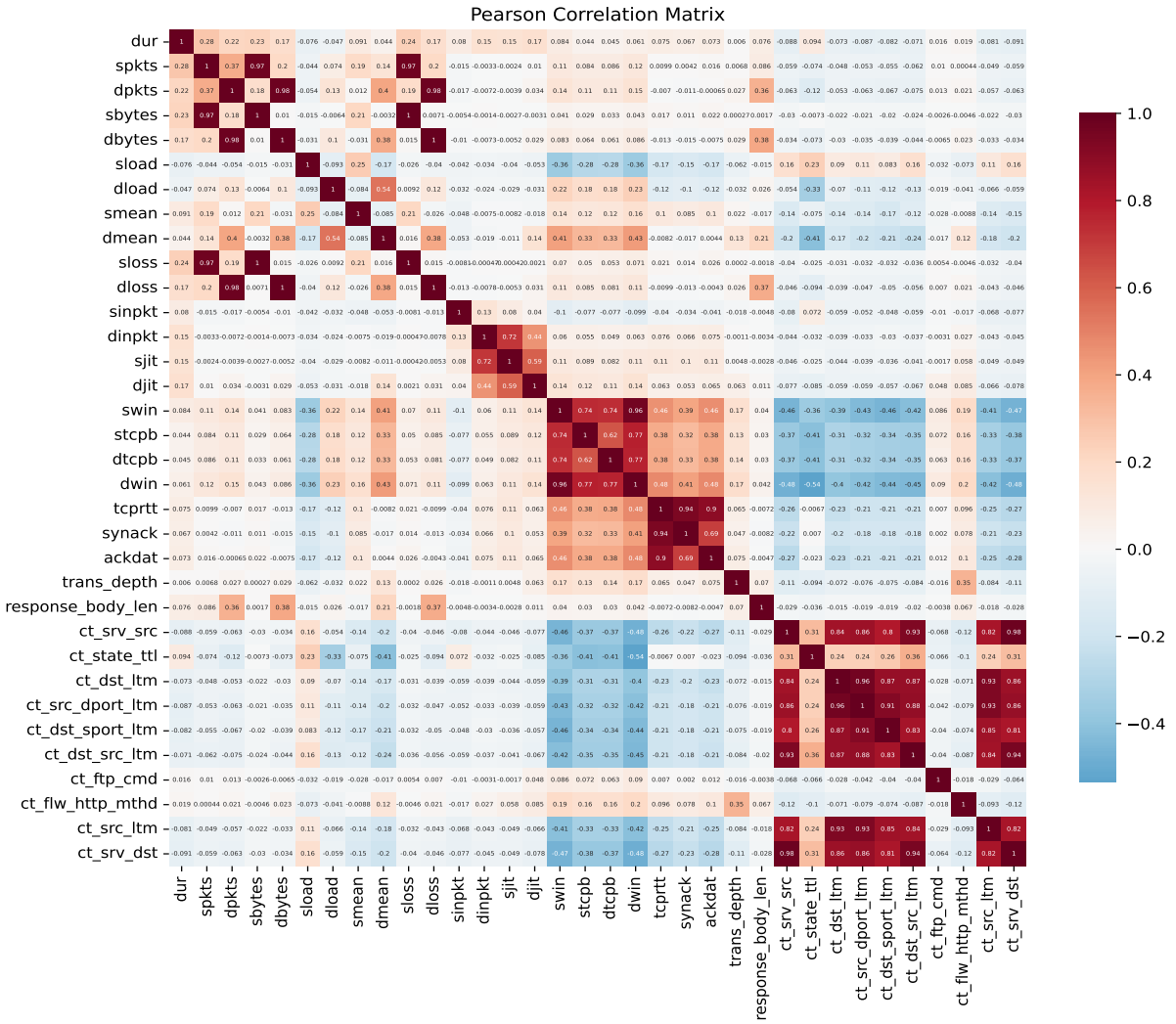


Figure 4.7: Pearson Correlation Matrix of the selected dataset

Preparation of the UNSW-NB15 Dataset for t-SNE

In this section, we will implement the t-distributed Stochastic Neighbor Embedding (t-SNE) technique, described in detail in Section. 2.8.4 of our theoretical framework. t-SNE is a technique used in data analysis to reduce the complexity of multidimensional data and allow its representation in a lower dimensional space. The main goal of t-SNE is to preserve the relationships between points in the original data set rather than simply capturing the total variability, which can be useful for identifying intricate clusters or patterns in the data.

Standardizing the data studied is one of the first steps for correctly applying the t-SNE algorithm. This is not strictly necessary. However, they can help ensure that features affect the similarity measure similarly. For this, we will follow the same steps performed in Section. 4.2.1.

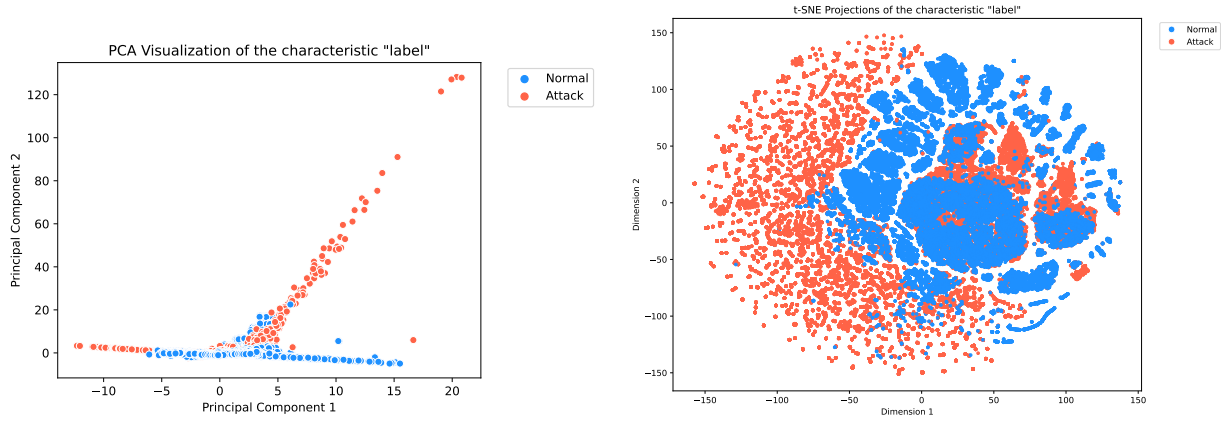
After this, we will perform the t-SNE on our data with the help of the sklearn.manifold library. We will apply the algorithm with a number of components equal to 2, which means that we will reduce our data to a two-dimensional or 2-dimensional space, and we will also

work under a seed or a random_state of 42; this number only helps us to have a reference when doing several repeated tests and be able to obtain the same results, we can skip it in case we do not want to repeat the tests.

```
from sklearn.manifold import TSNE

#apply t-SNE
tsne = TSNE(n_components=2, random_state=42)
#save generated projections in a variable
projections_tsne = tsne.fit_transform(unsw_dataset_standardized[
    numeric_cols])
```

Once the execution of our code is finished, we can graph our projections in 2 dimensions. Fig. 4.8b shows how the Normal and Attack classes look much more differentiated compared to Fig. 4.8a, the graph of the PCA projections. However, there is still some overlap; several of the attack logs tend to mimic the behavior of normal logs.

This overlap problem has already been mentioned by other investigations such as [230], which add that the UNSW-NB15 dataset suffers from the said problem, and in this work, the same overlap of classes can be observed in its graphs.



(a) PCA Projections of the **label** feature

(b) t-SNE Projections of the **label** feature

Figure 4.8: Comparison of the two types of projections based on **label** feature

Despite this, we obtain good and sufficient class differentiation to train our target model, the Isolation Forest. This model will be trained in the same way as the results of the two algorithms carried out, both with the PCA projections and with the t-SNE projections; we will do this to compare the results from the graphs that will be generated.

4.3 Isolation Forest

Once both the linear (PCA results) and non-linear (t-SNE results) projections have been obtained, we will go to train the Isolation Forest model with these projections; for this we will use the following small code:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.ensemble import IsolationForest

projections = np.load('/content/projections_type.npy')

isoforest_model = IsolationForest(contamination=n_contamination,
    random_state=42)
isoforest_model.fit(projections_type)
```

In the provided code snippet, we have a variable within the contamination value that is going to be modifiable since our goal is to be able to detect zero-day attacks within a dataset, which unfortunately does not have labels for this type of attack, a more qualitative and comparative study must be carried out. For this reason, several tests will be carried out using different contamination values. By default, we will use the one that the documentation itself mentions, that is, a value of 0.1 [231], then we will do it with values of 0.05, 0.25, and the maximum value allowed, which would be 0.5. Also, we have a variable called projections_type, which will indicate with which projection we will train our model. First, we will use the PCA projections and then the t-SNE; the respective methods will be shown below. Finally, it is important to mention that the value of random_state helps us to obtain the same results every time we train our model; we will set this fixed because what we need now is to change our contamination level since this is the most important parameter of the model since it tells us what percentage of outliers we think we have in our data [232].

4.3.1 Training with PCA Projections

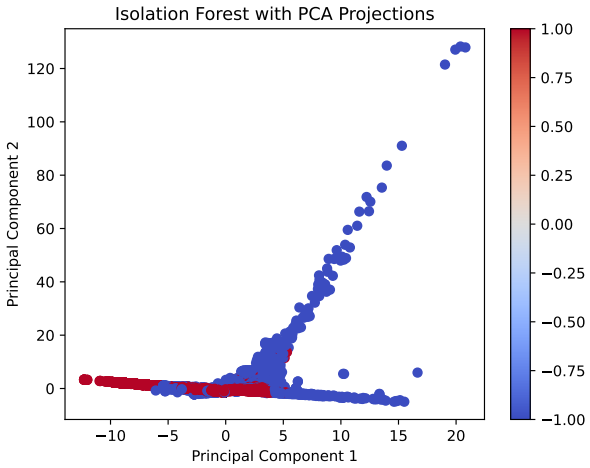
Next, we will apply the different PCA graphics and choose the different contamination values.

In Fig. 4.9a, 4.9b, 4.9c and 4.9d, we have the different training plots of our model with the contamination values chosen using the PCA projections. As we can see in the four graphs, there is a large percentage of anomalies, that is, deals with -1, even with little or a lot of contamination, which leads us to think that there is no good training model with this type, of projections or that it does not correctly differentiate between abnormal and normal data. Furthermore, the anomalies overlap in non-outliers, demonstrating once again the problem of class overlap existing in our dataset, and since they are projections of linear features, very little categorization is observed in these two types of values.

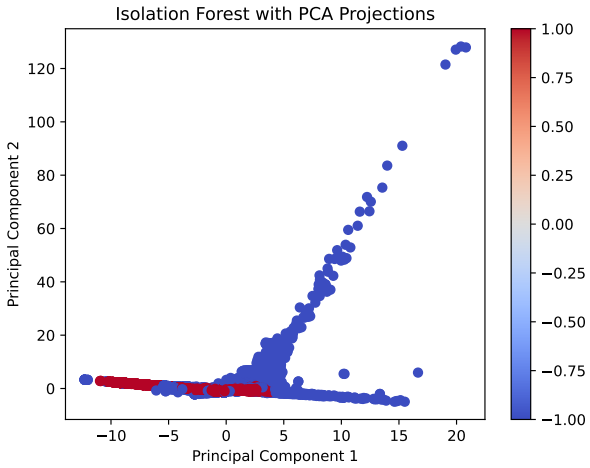
4.3.2 Training with t-SNE Projections

Next, we will apply the different t-SNE graphics and choose the different contamination values.

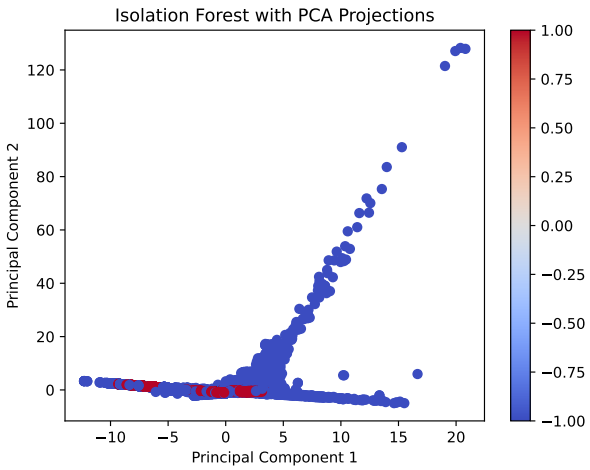
On the other hand, in Fig. 4.10a, 4.10b, 4.10c and 4.10d, we have the different training plots of our model with the contamination values chosen using the t-SNE projections. As we can see in the four graphs, there is an acceptable, logical, and differentiable percentage



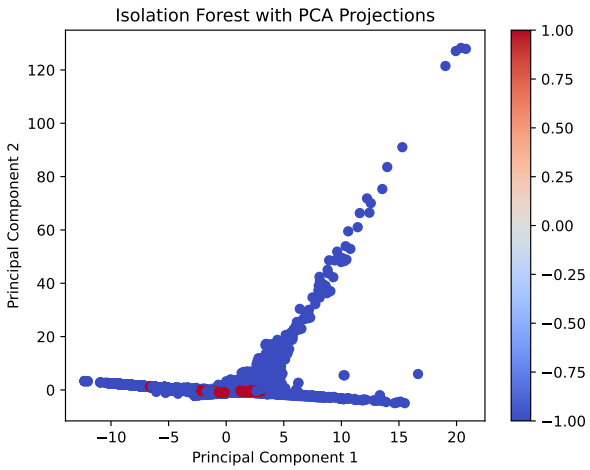
(a) Contamination value = 0.05



(b) Contamination value = 0.1



(c) Contamination value = 0.25

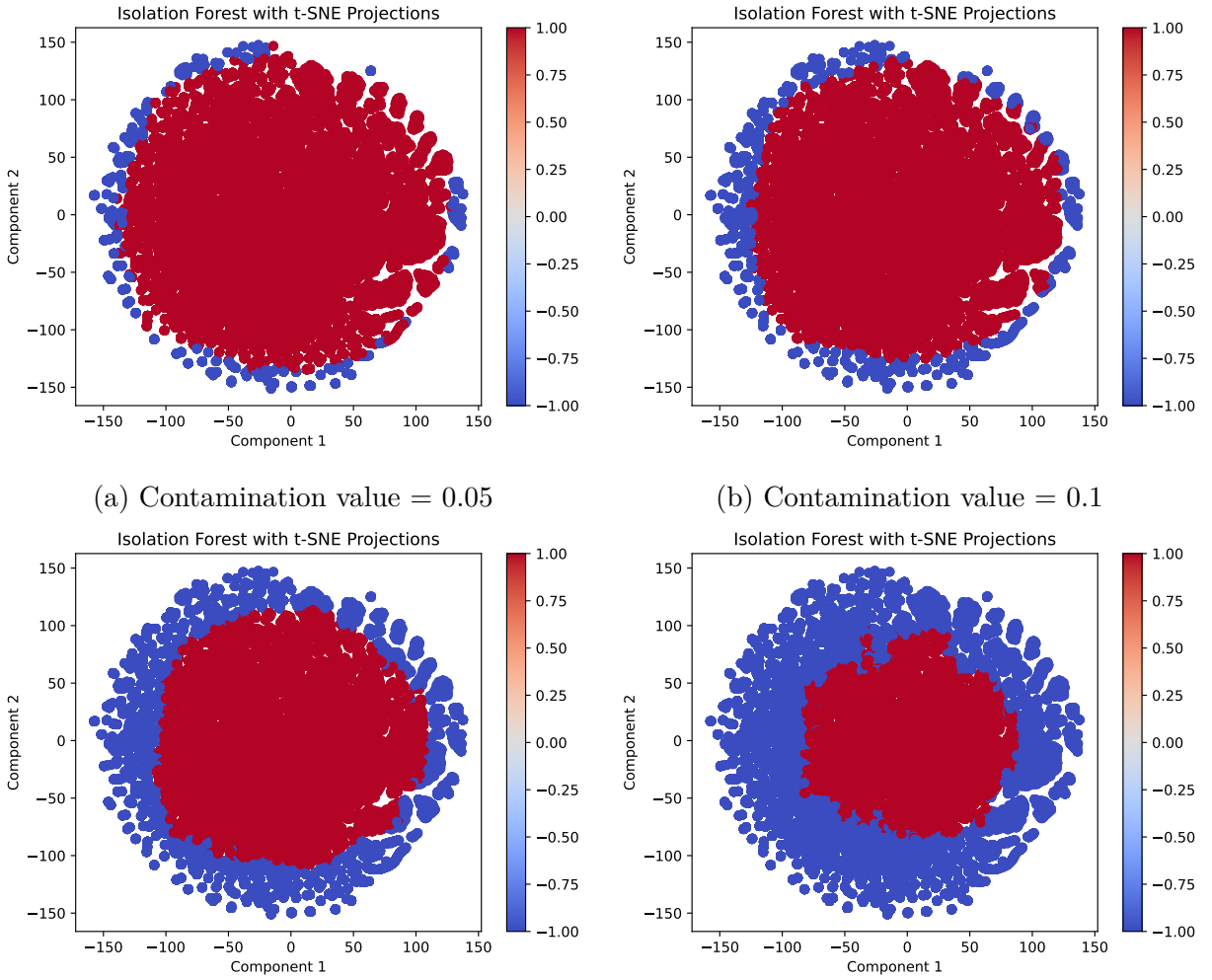


(d) Contamination value = 0.50

Figure 4.9: Isolation Forest trained on PCA projections with different contamination values.

of anomalies, which leads us to think there is a good model training with this type of nonlinear feature projection. Also, the anomalies do not overlap on the non-outliers, which shows that the model works much better if we treat our dataset nonlinearly, as we saw in the previous section.

Finally, since it is a very big challenge to be able to detect zero-day attacks with data sets not labeled with this type of attack, it will be decided to obtain the respective results and conclusions in the next section with arguments based on qualitative criteria rather than quantitative justifications, with the help of visualization graphs as we have been doing in this Methodology section.



(a) Contamination value = 0.05

(b) Contamination value = 0.1

(c) Contamination value = 0.25

(d) Contamination value = 0.50

Figure 4.10: Isolation Forest trained on t-SNE projections with different contamination values.

4.3.3 Filtering Anomalous and Non-abnormal Data

Once our model has been trained with both the linear and non-linear feature projections, we proceed to use the results obtained with the t-SNE; we use these results because, through visualization, we can observe a better decoupling between the normal and abnormal classes. Also, we will specifically use the ones that were generated with a contamination value of 0.05, although the default value is 0.1 as mentioned in [231]; we will use this because what we are trying to detect is totally unknown or very unknown attacks out of the normal.

First, we will calculate the percentage of anomalies and normal data in our data set after training. Then, we will analyze the comparison of some features and choose specific cases to discuss them in tables; this will be done in the next section, which is about results.

Chapter 5

Results and Discussion

This section exposes the anomalies found in our dataset and four different examples of feature comparisons that are part of the network traffic within the UNSW-NB15 dataset, which was used to train the Isolation Forest model. We will obtain the data from the model results obtained in the previous section. The analysis and corresponding discussion will be carried out using visualization graphs. These examples of network traffic were chosen randomly from all of their totals so that there is no bias when analyzing the model.

5.1 Anomalies found

After having trained the Isolation Forest model with a contamination level of 0.05, we found within our dataset a total of 4111 anomalous data, that is, anomalous traffic. In Fig. 5.1, we can see how of 100% of our dataset, 95.02% marked in blue belongs to normal data while 4.98%, marked in red, belongs to abnormal data, and in a certain way this can be considered a positive result of the model since this amount does not exactly belong to the traffic where all the existing types of attacks are found, this can be compared in a better way if we look at Fig. 4.1.

5.2 Dur vs. Spkts

We will take as a general test case the comparison between the "dur" and "spkts" features of the selected data set. Feature "dur" represents the duration of the network connection, indicating how long a particular connection or specific flow lasts. On the other hand, "spkts" means the count of packets from source to destination, that is, it is the number of packets transmitted from where the transmission begins to the place where it arrives. These two features are closely related since the duration of a network connection often plays an important role in determining the number of packets exchanged. Longer connections usually involve a greater number of packets, since more data is transferred over time. Analyzing the interaction between these two features can provide valuable information about network behavior in the context of its security.

In Fig. 5.2, we can see a comparison between these two features, in which normal and

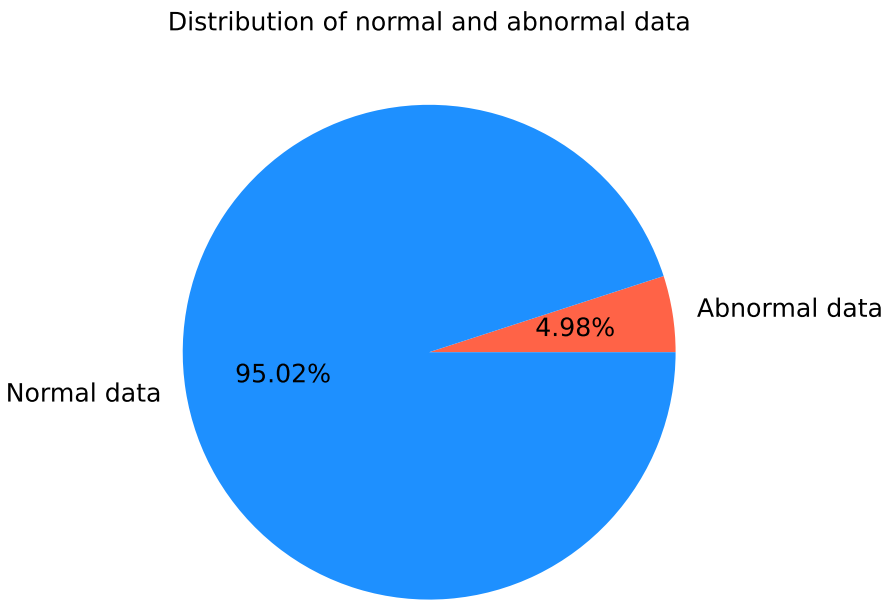


Figure 5.1: Distribution of the dataset after model training, its separation is made up of two classes, blue for normal data and red for data considered anomalous

abnormal data are shown. The "dur" feature is shown on the These data have two colors; the normal ones are represented with blue and the abnormal ones with red. On the other hand, it can be seen how the anomalous data are around the point (0,0) and the point (60, 0). Under the context of what was mentioned above, it can be mentioned that our model considers an anomaly when 0 are being sent. Packets with 0 duration or also when 0 packets are sent but the connection remains open for up to 60 seconds, which indicates that the session is still open but data packets are not explicitly being sent, but rather something else may be happening—strange situation.

In addition to this general analysis, we will do a more detailed one, choosing a specific case or transmission from our dataset. We will take into account extremely normal data and extremely anomalous one; we will see this in the next subsection.

5.2.1 Traffic with Id: 70537

We will discuss the network traffic with id: 70537; we chose this case because it has a "spkts" value of 2 and a "dur" value of 59.999989, approximating it would give a value of 60, thus giving the model considers it an anomaly. According to the features in Table. A1, all features that have a value different from 0 in this case were included in Table. 5.1 and Table. 5.2, the features that are not detailed in those tables, have a value equal to 0.

Taking into account the values of Table. 5.1 and 5.2, we will carry out the respective analysis given that the model classified it as an anomaly:

- **Low packet rate:** The packet rate (rate) is very low, with only 0.016667 packets per second. This is much slower than you would normally see on a typical network,

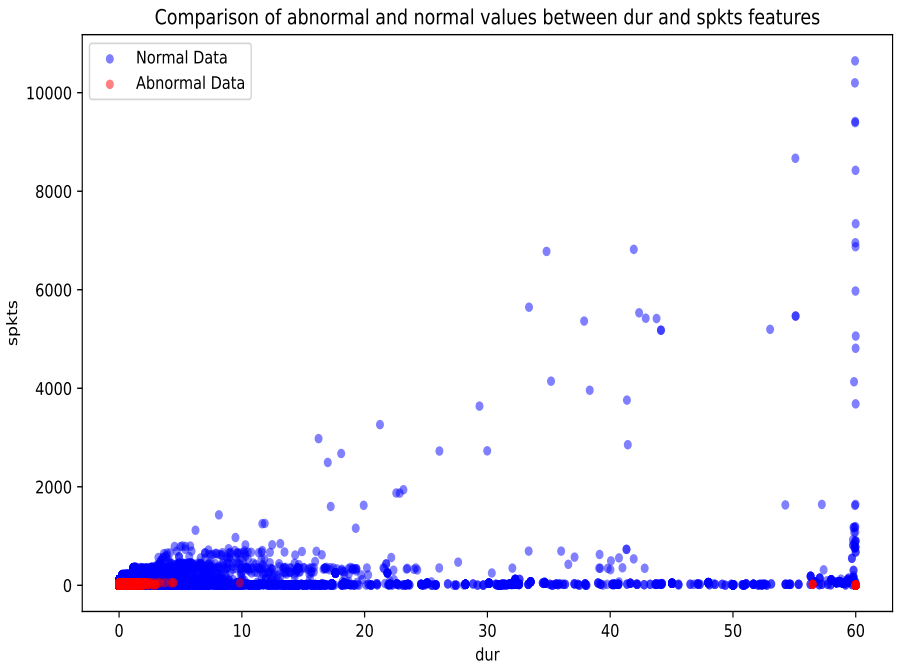


Figure 5.2: Distribution of the dataset after model training, its separation is made up of two classes, blue for normal data and red for data considered anomalous

Values of the traffic features with id: 70537.

Table 5.1: Part 1 of the values of traffic features with id: 70537

Feature	***Value***
id	70537
dur	59.999989
proto	arp
state	INT
spkts	2
sbytes	92
rate	0.016667
sload	6.133335
sinpkt	60000.704
smean	46

Table 5.2: Part 2 of the values of traffic features with id: 70537

Feature	***Value***
ct_srv_src	1
ct_state_ttl	2
ct_dst_ltm	1
ct_src_dport_ltm	1
ct_dst_sport_ltm	1
ct_dst_src_ltm	1
ct_src_ltm	1
ct_srv_dst	1
is_sm_ips_ports	1

which could be indicative of unusual activity.

- **Unusually High Next Packet Time:** The next packet time (sinpkt) is extremely high, at over 60,000 milliseconds (60 seconds). This suggests a long period of inac-

tivity between packets, which is uncommon on many networks.

- **Low average packet size:** The average packet size (smean) is only 46 bytes. This could be considered small compared to the typical packet size on a network.
- **Unique connection characteristics:** Several of the counters, such as ct srv src, ct state ttl, ct dst ltm, and others, have values of 1, suggesting that this connection has unique characteristics that differentiate it from other connections.

In summary, this connection could have been labeled as an anomaly due to multiple unusual characteristics, such as short duration, low packet rate, and extremely high packet sending time. The great inconsistency remains by having a transmission time of 60 seconds and only 2 sending packets; at this point, one could ask: What was happening in all that time if information was not being sent or received? Maybe it was an open door for another type of activity? For these reasons, our model takes it as an anomaly, and since it does not belong to any type of attack in the data set, it could be considered a new attack or a zero-day attack.

5.3 Sbytes vs Dbytes

As another general test and analysis case, we will use the comparison between the "sbytes" and "dbytes" features. "Sbytes" means the number of bytes from source to destination, that is, the total number of bytes transmitted from the source to the destination during a network connection, while "dbytes" refers to the count of bytes from destination to source, which indicates the total number of bytes transmitted in the opposite way. These two attributes are inherently intertwined, as together they represent the entire data transfer between two points in a network connection. By examining the relationship between "sbytes" and "dbytes", we can also find valuable information about data flow patterns and possible anomalies within network traffic. Abnormal variations or discrepancies between these byte counts may indicate unusual behavior.

In Fig. 5.3, we can see a comparison between these two features, in which normal and abnormal data are shown. The feature "dbytes" is shown on the X axis and the feature "sbytes" is shown on the Y axis. In the same way as the previous case, this data has two colors: the normal ones are represented with blue, and the abnormal ones with red. In addition, it can be observed how the anomalous data are around the point (0,0), under the context of what was mentioned above, it can be mentioned that our model considers an anomaly when 0 bytes of information are being sent and received in network traffic. However, at the same time, it can be considered that it would be correct since if we do not send bytes of information, we should not receive it either, but if we analyze Fig. 5.4, where the features "sbytes", "dbytes" and "dur" we can take into account that this traffic happens when the session time is around 60 seconds, so we can conclude that an anomaly is happening since it should not last long if there is no flow of bytes. It is important to mention that both "sbytes" and "dbytes" are under the scientific notation of 10^7.

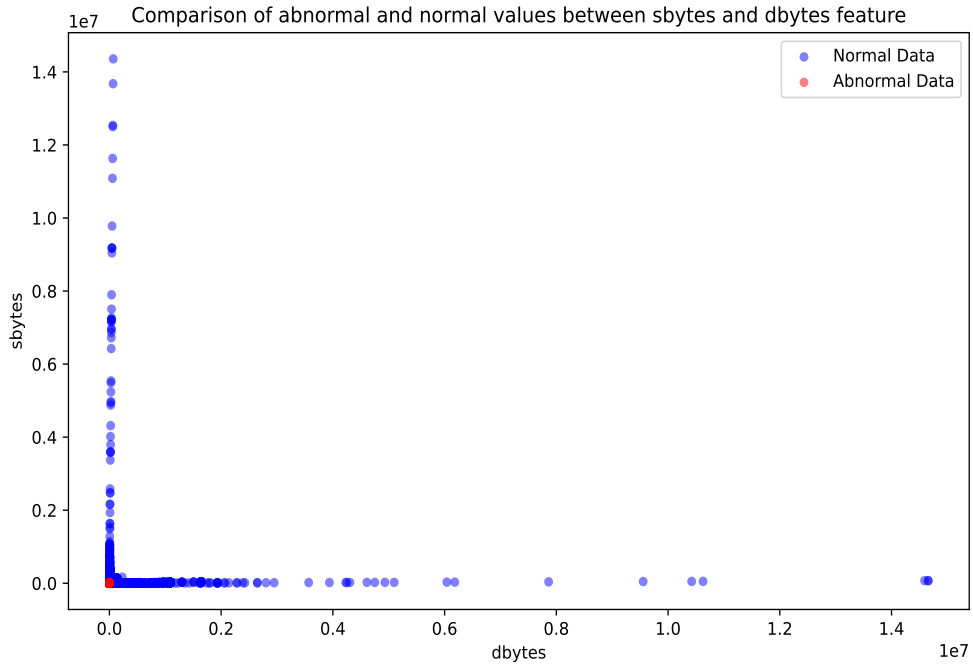


Figure 5.3: Distribution of "sbytes" and "dbytes" features after model training, its separation is made up of two classes, blue for normal data and red for data considered anomalous

5.3.1 Traffic with Id: 42

We will discuss the network traffic with id: 42; we choose this case because it has a "sbytes" value of 92, a "dbytes" value of 0 and a "dur" value of 59.995678, approximating it would give a value of 60, thus giving the model considers it an anomaly. According to the features in Table. A1, all features that have a value different from 0 in this case were included in Table. 5.3 and Table. 5.4, the features that are not detailed in those tables, have a value equal to 0.

Taking into account the values of the recently mentioned tables, we will analyze each of the features with their respective values below:

- **Unusually long duration:** The connection duration is almost 60 seconds (59.995678 seconds). This is longer than would normally be expected for an ARP connection, which is typically a quick MAC address resolution operation. A duration that was too long could have drawn attention to the model and therefore it could have been considered an atypical behavior.

- **High packet rate:** Although the packet rate is relatively low (0.016668 packets per second), it could be unusually high for an ARP connection. This could suggest anomalous or unexpected activity.

- **Extremely High Next Packet Time:** The next packet time (sinpkt) is exceptionally high, with a value of 59998.2 milliseconds. This indicates that there was a long period of inactivity between packets. In a typical ARP connection, packets are usually successive and fast, so this long idle time may be unusual.

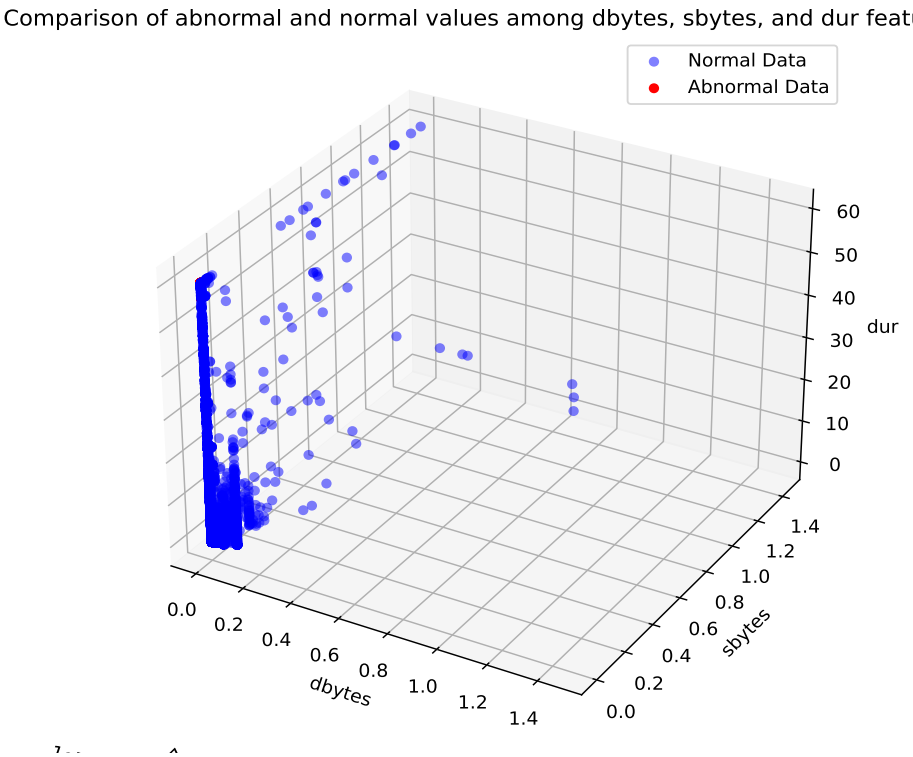


Figure 5.4: Distribution of "sbytes", "dbytes" and "dur" features after model training, its separation is made up of two classes, blue for normal data and red for data considered anomalous

Values of the traffic features with id: 42.

Table 5.3: Part 1 of the values of traffic features with id: 42

Feature	***Value***
id	42
dur	59.995678
proto	arp
state	INT
spkts	2
sbytes	92
rate	0.016668
sload	6.133775
sinpkt	59998.2
smean	46

Table 5.4: Part 2 of the values of traffic features with id: 42

Feature	***Value***
ct_srv_src	2
ct_state_ttl	2
ct_dst_ltm	2
ct_src_dport_ltm	2
ct_dst_sport_ltm	2
ct_dst_src_ltm	2
ct_src_ltm	2
ct_srv_dst	2
is_sm_ips_ports	1

5.4 Rate vs Spkts/Dpkts

On the other hand, we will use the comparison between the features “rates” and “spkts” or “dpkts”. The ”rate” feature indicates how many bits of data are transmitted or received per unit of time during a connection, and the “spkts” and “dpkts” features represent the total number of packets that are sent from the source and received at the destination respectively. These three features may be closely related since, in normal situations, an increase in the transfer rate should be correlated with an increase in the number of packets as more data is transmitted. However, if the transfer rate is extremely high compared to the number of packets, this could indicate an anomaly. It could suggest that someone is flooding the network with fake or malicious traffic.

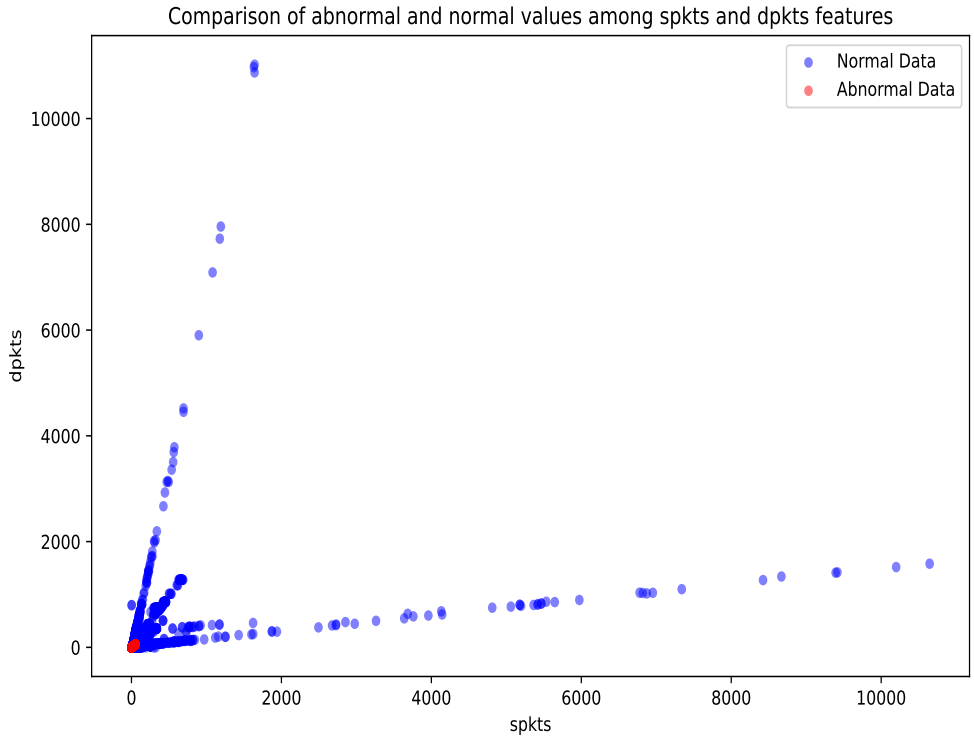


Figure 5.5: Distribution of “spkts” and “dpkts” features after model training, its separation is made up of two classes, blue for normal data and red for data considered anomalous

In Fig. 5.5, we can see a comparison between these two features, showing normal and abnormal data. The feature “spkts” is shown on the X-axis and the feature “dpkts” is shown on the Y-axis. In the same way as the previous case, this data has two colors: the normal ones are represented with blue, and the abnormal ones with red. Furthermore, it can be seen how the anomalous data is found around the point (0,0), which indicates that this network traffic is not sending packets from its origin and is not receiving them at its destination. However, at the same time, it can be considered that it would be correct since it may be a case in which information is not being sent, but, on the other hand, if we analyze Fig. 5.6, where the three aforementioned features appear, we can take into account that the case of sending and receiving around 0 to 100 information packets occurs with a transfer rate that is too high since the “rate” feature is on a scale of 10^6 and taking into

account the previous justification, this could be considered an anomaly since this traffic could be flooded with harmful information from other places.

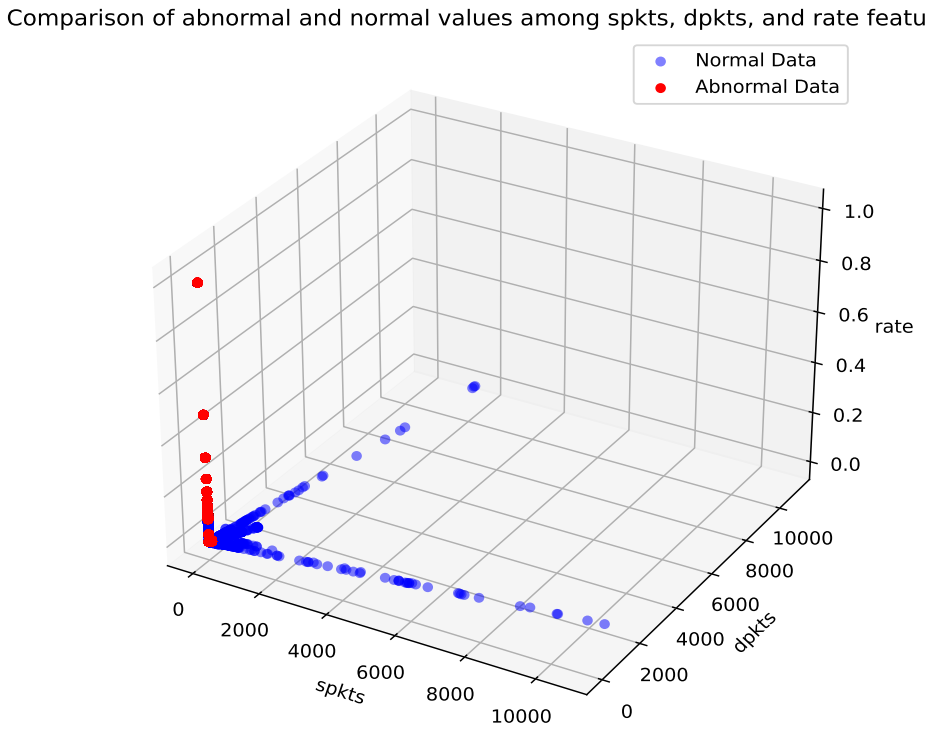


Figure 5.6: Distribution of “spkts”, “dpkts”, and “rate” features after model training, its separation is made up of two classes, blue for normal data and red for data considered anomalous

5.4.1 Traffic with Id: 1248

We will discuss the network traffic with id: 1248, we chose this case because observing the 3 features studied, they have values close to the anomalies observed in Fig. 5.4. The features of the dataset, whose values are different from 0, are displayed in Table. 5.5 and Table. 5.6, features that are not detailed have a value equal to 0.

Taking into account the values of the recently mentioned tables, we will analyze each of the features with their respective values below:

- **Extremely Short Duration:** The duration of the connection is only 5 microseconds (0.000005 seconds). This extremely short duration is unusual and could indicate that the connection was ephemeral or closed almost immediately after being established.
- **Unknown or Uncommon Protocol:** The “proto” feature has the value “trunk-1”, which does not correspond to a commonly recognized network protocol like TCP or UDP. An unknown or unusual protocol could raise suspicions of anomaly.

Values of the traffic features with id: 1248.

Table 5.5: Part 1 of the values of traffic features with id: 1248

Feature	***Value***
id	1248
dur	0.000005
proto	trunk-1
state	INT
spkts	2
sbytes	200
rate	200000.0051
sttl	254
sload	160000000
sinpkt	0.005

Table 5.6: Part 2 of the values of traffic features with id: 1248

Feature	***Value***
smean	100
ct_srv_src	6
ct_state_ttl	2
ct_dst_ltm	3
ct_src_dport_ltm	3
ct_dst_sport_ltm	3
ct_dst_src_ltm	6
ct_src_ltm	5
ct_srv_dst	6

- **Extremely High Packet Rate:** The packet rate (rate) is 200,000.0051 packets per second, which is very high. Such a high rate could be considered unusual and might be an indication of suspicious or unexpected activity.
- **Significant Service Load:** The 'sload' value is 160,000,000, indicating a very high service load. This could be considered anomalous compared to typical connections.
- **Very Short Next Packet Time:** The time between the current packet and the next one (sinpkt) is only five milliseconds, suggesting that packages were sent very rapidly, one after the other. This intense and rapid activity could be seen as unusual.

Taken together, these factors, including the extremely short duration, uncommon protocol, high packet rate, and significant service load, might have led the anomaly detection model to consider this connection as an anomaly. The model could have interpreted this traffic as unusual compared to the typical behavior of the network and therefore labeled this connection as an anomaly based on these unusual features.

In concluding this section, we have undertaken a qualitative analysis of how our model could detect zero-day attacks, driven by the pursuit of identifying anomalies. We have illustrated our approach with three distinct case studies, each featuring carefully selected network traffic samples from our dataset. It is worth noting that these examples were chosen to be both illustrative and comprehensible, ensuring that readers can easily grasp the concepts. Furthermore, to substantiate the efficacy of our approach, we employed 2D and 3D visualizations when necessary. These visual aids were instrumental in demonstrating how the algorithm developed in this study successfully aligned with our initial objectives.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This work proposed an unsupervised learning algorithm capable of detecting zero-day attacks based on anomalies within a dataset, whose data belongs to benign or malignant network traffic. This algorithm was carried out based on fundamental, theoretical, and practical aspects of different Machine Learning techniques, including the respective analysis and statistical methods, data dimensionality reductions, and concluding with the training of our main model based on anomaly detection, Isolation Forest. The most important thing about this proposal, unlike some recently published works, is the combination of techniques that were used, in addition to the use of an unsupervised learning model based on anomalies and the use of all the numerical characteristics existing in our dataset, if well we reduced the dimensionality of this, internally they were all maintained. Also, the results obtained from the training of our model were justified graphically, qualitatively, explicitly, and in detail in order to validate these results. It is important to mention the latter because compared to other works, only accuracy or some other known supervised learning metric is taken as justification, however, something very fundamental to consider is that zero-day attacks are not found in any open source dataset and much less with a label that tells us that it belongs to this type of attack, unless it has been generated internally or in a real laboratory or with some artificial intelligence tool, this is necessary to mention since, as mentioned at the beginning of this work, there is no record of zero-day attacks since these appear at the same time that the security of a network is being violated.

A comprehensive exploration of state-of-the-art machine learning algorithms and anomaly detection methodologies was presented. To manage this specific objective in the best way, robust cybersecurity concepts and the different prevention and detection methods that exist, which in turn have been improved with the help of ML, were included. A specific model was also highlighted, the Isolation Forest, an anomaly-based ML model, which would help us evaluate its suitability for predicting zero-day attacks by identifying behavioral patterns. This extensive study formed the foundational knowledge for subsequent phases of our work.

A robust algorithm designed for the detection of anomalies in network traffic was developed. We successfully created an algorithm that harmoniously integrates an anomaly-based

machine learning technique, such as Isolation Forest, and dimensionality reduction techniques, specifically feature projection for non-linear data such as t-SNE, which results in The result is an ML algorithm capable of detecting anomalies within network traffic.

To validate the effectiveness of our proposed algorithm, we conducted a meticulous and comprehensive evaluation. We employed qualitative, detailed, and graphical analyses to assess various properties within network traffic data. Initially, we utilized graphs to illustrate the behavior of PCA with our data, revealing non-linearity. To confirm this, we employed the Pearson Correlation Matrix, which led us to utilize the t-SNE algorithm for handling non-linear data. Subsequently, we employed various graphs to visualize the Isolation Forest model trained with the projections generated by t-SNE. These visualizations were performed at different contamination levels, enabling the distinction between anomalous and non-anomalous classes. These analyses provided compelling evidence of our algorithm's capability to detect anomalies.

6.2 Future Work

Furthermore, it is worth noting that the algorithm developed in this study can be considered a significant contribution to the evolution of Intrusion Detection Systems. By addressing the detection of zero-day attacks through the identification of anomalies in network traffic, our approach closely aligns with the nature of an IDS. An effective IDS not only relies on identifying known threat patterns but also must be capable of recognizing unusual behaviors that may indicate an intrusion. Considering that a zero-day attack is, by definition, unknown and lacks predefined signatures, our approach becomes a valuable resource for network security defense. In essence, the proposed algorithm not only detects anomalies but can also be viewed as the heart of an advanced IDS, capable of adapting to emerging and unknown threats.

Looking to the future, there are numerous exciting avenues for further exploration and development stemming from the foundations laid by this research. One notable direction involves the practical implementation of our algorithm in the form of IDS software. By doing so, our model could be seamlessly integrated into network infrastructures, always alert and prepared to analyze and respond to network traffic anomalies in real-time.

Expanding on this idea, the software could be designed with adaptability and scalability in mind. It could evolve alongside the ever-changing threat landscape, incorporating machine learning models that continually learn and refine their anomaly detection capabilities. This would allow the IDS not only to detect known anomalies but also to adapt and identify emerging threats that lack established patterns. Additionally, the system could be enriched with a comprehensive reporting and alerting mechanism, ensuring that network administrators are quickly informed of any potential intrusion, thus facilitating a rapid response to incidents.

On the other hand, as cybersecurity continues to evolve in this era, it is critical to consider ethical implications and data privacy concerns. Future work should delve deeper into ensuring that the algorithm aligns with best practices for data privacy, fairness, and transparency.

In conclusion, this work represents a significant step forward in the realm of network security and intrusion detection. The developed algorithm, which leverages unsupervised

learning and anomaly detection techniques, offers a novel approach to identifying zero-day attacks. By focusing on the recognition of anomalies within network traffic data, we break away from the limitations imposed by the absence of predefined signatures for zero-day attacks. This research not only contributes to the field of Intrusion Detection Systems but also emphasizes the importance of continuous adaptation and innovation in the face of evolving cyber threats. As our digital landscape becomes increasingly complex, the need for robust, adaptable, and proactive security measures is more critical than ever. The algorithm presented herein is a testament to our commitment to staying ahead of the curve, enhancing network security, and safeguarding the integrity of digital infrastructure. It is my hope that this work serves as a foundation for further advancements in the ever-evolving field of cybersecurity.

Bibliography

[1] H. Bangui, M. Ge, and B. Buhnova, "A hybrid machine learning model for intrusion detection in vanet," *Computing*, vol. 104, no. 3, pp. 503–531, 2022.

[2] K. Gupta. (2018) Applied dimensionality reduction techniques using python. [Online]. Available: https://www.learndatasci.com/tutorials/applied-dimensionality-reduction-techniques-using-python/

[3] D. Scientist, "Comprende el algoritmo t-sne en 3 pasos," https://datascientest.com/es/comprende-el-algoritmo-t-sne-en-3-pasos, 07 2023, accessed on July 27, 2023.

[4] H. Song, G. A. Fink, and S. Jeschke, *Security and privacy in cyber-physical systems: foundations, principles, and applications*. John Wiley & Sons, 2017.

[5] D. J. Brown, B. Suckow, and T. Wang, "A survey of intrusion detection systems," *Department of Computer Science, University of California, San Diego*, 2002.

[6] M. Zamani and M. Movahedi, "Machine learning techniques for intrusion detection," *arXiv preprint arXiv:1312.2177*, 2013.

[7] Cybriant, "Can traditional enterprise antivirus protect from unknown threats?" https://cybriant.com/enterprise-antivirus/, 2023, accessed on November 1, 2023.

[8] Kaspersky, "What is a zero-day exploit?" https://www.kaspersky.com/resource-center/definitions/what-is-a-zero-day-exploit, 2019.

[9] P. Patidar and H. Khandelwal, "Zero-day attack detection using machine learning techniques," *International Journal of Research and Analytical Reviews*, vol. 6, no. 1, pp. 1364–1367, 2019.

[10] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[11] G. Cibernos, "Tipos de seguridad informática, ¿cuáles son?" Apr 2022. [Online]. Available: https://www.grupocibernos.com/blog/tipos-de-seguridad-informatica-cuales-son

[12] M. E. Whitman and H. J. Mattord, *Principles of information security*. Cengage learning, 2021.

[13] C. P. Pfleeger, *Security in computing*. Prentice-Hall, Inc., 1988.

[14] F. J. García-Peñalvo, J. Cruz-Benito, M. T. Martín-Valdivia, R. Therón, and A. García-Holgado, "Ciberseguridad y bibliotecas: apuntes para una propuesta de formación sobre riesgo tecnológico en bibliotecas," *Métodos de Información*, vol. 10, no. 19, pp. 1–15, 2020.

[15] M. E. Whitman and H. J. Mattord, *Principles of Information Security*. Cengage Learning, 2018.

[16] M. Stamp, *Information Security: Principles and Practice*. John Wiley Sons, 2011.

[17] D. L. Hall, *Information Technology Auditing and Assurance*. Cengage Learning, 2015.

[18] C. Lewis and M. Knapp, *IT Auditing: Using Controls to Protect Information Assets*. McGraw-Hill Education, 2012.

[19] National Institute of Standards and Technology, "Computer Security Resource Center (CSRC)," https://csrc.nist.gov/, Último acceso: 21 de mayo del 2023 21:09.

[20] Cisco, "What is Cybersecurity?" https://www.cisco.com/c/en/us/products/security/what-is-cybersecurity.html, Último acceso: 21 de mayo del 2023 21:09.

[21] E. GDPR, "Article 32 gdpr - security of processing," https://gdpr-info.eu/art-32-gdpr/#:~:text=The%20controller%20and%20processor%20shall,Union%20or%20Member%20State%20law., Fecha de acceso: 21 de mayo del 2023 21:09.

[22] Computer Emergency Response Team Coordination Center (CERT/CC), "Incident Response," https://www.cert.org/incident-management/, Fecha de acceso: 21 de mayo del 2023 21:09.

[23] U. Online, "Top cyber security blogs and websites," https://onlinedegrees.sandiego.edu/top-cyber-security-blogs-websites/, 2021, accessed: July 19, 2023.

[24] C. Magazine, "Top 10 news sites for the cyber security industry revealed," August 2021. [Online]. Available: https://cybermagazine.com/top10/top-10-news-sites-cyber-security-industry-revealed

[25] T. Pearson, "What is cyber security?" Digital Guardian, 2018, accessed: July 19, 2023. [Online]. Available: https://digitalguardian.com/blog/what-cyber-security

[26] Y. Li and Q. Liu, "A comprehensive review study of cyber-attacks and cyber security; emerging trends and recent developments," *Energy Reports*, vol. 7, pp. 8176–8186, 2021.

[27] TechTarget, "What is cloud security and why is it important?" https://www.techtarget.com/searchsecurity/definition/cloud-security, 2022, accessed on May 22, 2023.

[28] CompTIA, "Network security basics: Definition, threats and solutions," https://www.comptia.org/content/guides/network-security-basics-definition-threats-and-solutions, 2019, accessed on May 22, 2023.

[29] Cisco, "What is network security?" https://www.cisco.com/c/en/us/products/security/what-is-network-security.html, 2022, accessed on May 22, 2023.

[30] IBM, "Ibm network security," https://www.ibm.com/topics/network-security, 2023, accessed on May 22, 2023.

[31] CSO Online, "What is application security? a process and tools for securing software," https://www.csoonline.com/article/3315700/what-is-application-security-a-process-and-tools-for-securing-software.html, 2018, accessed on May 22, 2023.

[32] TechTarget, "What is application security? a process and tools for securing software," https://www.techtarget.com/searchsoftwarequality/definition/application-security, 2018, accessed on May 22, 2023.

[33] Imperva, "Application security," https://www.imperva.com/learn/application-security/application-security/, 2023, accessed on May 22, 2023.

[34] CertStation, "Why application security is so important?" https://certstation.com/blog/application-security-important/, 2021, accessed on May 22, 2023.

[35] IBM, "Cloud security," https://www.ibm.com/topics/cloud-security, 2023, accessed on May 22, 2023.

[36] TechTarget, "Endpoint security," https://www.techtarget.com/searchsecurity/definition/endpoint-security, 2023, accessed on May 22, 2023.

[37] IBM, "Data security," https://www.ibm.com/topics/data-security, 2023, accessed on May 22, 2023.

[38] M. Riley and K. Mehrotra, "Cyberattacks are a grave threat to the global economy," 2023, accessed on May 22, 2023. [Online]. Available: https://www.bloomberg.com/professional/blog/cyberattacks-threat-global-economy/

[39] IFAC, "Cybersecurity is critical for all organizations – large and small," November 2019. [Online]. Available: https://www.ifac.org/knowledge-gateway/preparing-future-ready-professionals/discussion/cybersecurity-critical-all-organizations-large-and-small

[40] J. M. Borky, T. H. Bradley, J. M. Borky, and T. H. Bradley, "Protecting information with cybersecurity," *Effective Model-Based Systems Engineering*, pp. 345–404, 2019.

[41] Oodrive, "Los 10 principales tipos de ciberataques," https://www.oodrive.com/es/blog/seguridad/top-10-principales-tipos-de-ciberataques/, 2022, accedido el 29 de marzo de 2022.

[42] IBM, "¿qué es un ciberataque?" https://www.ibm.com/es-es/topics/cyber-attack, s.f., accedido el 18 de julio de 2023.

[43] S. K. Jha and S. S. Kumar, "Cybersecurity in the age of the internet of things: An assessment of the users' privacy and data security," in *Expert Clouds and Applications: Proceedings of ICOECA 2021*. Springer, 2022, pp. 49–56.

[44] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020.

[45] E. Skoudis and L. Zeltser, *Malware: Fighting malicious code*. Prentice Hall Professional, 2004.

[46] J. Hong, "The state of phishing attacks," *Communications of the ACM*, vol. 55, no. 1, pp. 74–81, 2012.

[47] Z. Alkhalil, C. Hewage, L. Nawaf, and I. Khan, "Phishing attacks: A recent comprehensive study and a new anatomy," *Frontiers in Computer Science*, vol. 3, p. 563060, 2021.

[48] L. Bilge and T. Dumitraş, "Before we knew it: an empirical study of zero-day attacks in the real world," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 833–844.

[49] "What is zero day? — preventing zero day attack," https://www.mimecast.com/content/zero-day-attack/, 2021.

[50] Mimecast, "What is zero day?" https://www.mimecast.com/content/zero-day-attack/, 2021.

[51] R. Kaur and M. Singh, "A survey on zero-day polymorphic worm detection techniques," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1520–1549, 2014.

[52] D. Karig and R. Lee, "Remote denial of service attacks and countermeasures," *Princeton University Department of Electrical Engineering Technical Report CE-L2001-002*, vol. 17, 2001.

[53] S. Specht and R. Lee, "Taxonomies of distributed denial of service networks, attacks, tools and countermeasures," *CEL2003-03, Princeton University, Princeton, NJ, USA*, 2003.

[54] PortSwigger, "What is SQL injection? tutorial & examples — web security academy - PortSwigger," July. [Online]. Available: https://portswigger.net/web-security/sql-injection

[55] W. G. Halfond and A. Orso, "Amnesia: analysis and monitoring for neutralizing sql-injection attacks," in *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, 2005, pp. 174–183.

[56] N. Author, "A survey of SQL injection attack detection and prevention," *No Journal*, No Year. [Online]. Available: https://www.researchgate.net/publication/276496047_A_Survey_of_SQL_Injection_Attack_Detection_and_Prevention

[57] A. Mallik, "Man-in-the-middle-attack: Understanding in simple words," *Cyberspace: Jurnal Pendidikan Teknologi Informasi*, vol. 2, no. 2, pp. 109–134, 2019.

[58] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE communications surveys & tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.

[59] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1851–1877, 2019.

[60] P. Chen, L. Desmet, and C. Huygens, "A study on advanced persistent threats," in *Communications and Multimedia Security: 15th IFIP TC 6/TC 11 International Conference, CMS 2014, Aveiro, Portugal, September 25-26, 2014. Proceedings 15*. Springer, 2014, pp. 63–72.

[61] A. Khalid, A. Zainal, M. A. Maarof, and F. A. Ghaleb, "Advanced persistent threat detection: A survey," in *2021 3rd International Cyber Resilience Conference (CRC)*. IEEE, 2021, pp. 1–6.

[62] N. Author, "Operation aurora," January 2010. [Online]. Available: https://www.wired.com/2010/01/operation-aurora/

[63] B. Bencsáth, G. Pék, L. Buttyán, and M. Félegyházi, "Duqu: A stuxnet-like malware found in the wild," *CrySyS Lab Technical Report*, vol. 14, pp. 1–60, 2011.

[64] K. Lab, "Kaspersky lab identifies operation "red october," an advanced cyber-espionage campaign targeting diplomatic and government institutions worldwide," 2013, accedido el 19 de julio de 2023.

[65] Redacción, "Sony pictures hack: 7 secretos de hollywood que reveló el hackeo a sony pictures," *BBC News Mundo*, diciembre 2014, accedido el 19 de julio de 2023.

[66] J. Vukalović and D. Delija, "Advanced persistent threats-detection and defense," in *2015 38Th international convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE, 2015, pp. 1324–1330.

[67] P. Security, "Ataques de día cero: el equivalente cibernético de covid-19," agosto 2020, accedido el 19 de julio de 2023.

[68] A. Patel, Q. Qassim, and C. Wills, "A survey of intrusion detection and prevention systems," *Information Management & Computer Security*, vol. 18, no. 4, pp. 277–290, 2010.

[69] D. A. Bhosale and V. M. Mane, "Comparative study and analysis of network intrusion detection tools," in *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*. IEEE, 2015, pp. 312–315.

[70] T. Alsmadi and N. Alqudah, "A survey on malware detection techniques," in *2021 International Conference on Information Technology (ICIT)*. IEEE, 2021, pp. 371–376.

[71] GeeksforGeeks, "Intrusion prevention system (ips)," September 2021. [Online]. Available: https://www.geeksforgeeks.org/intrusion-prevention-system-ips/

[72] P. A. Networks, "What is an intrusion prevention system?" *Palo Alto Networks Cyberpedia*, 2020. [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-prevention-system-ips

[73] Z. Inayat, A. Gani, N. B. Anuar, M. K. Khan, and S. Anwar, "Intrusion response systems: Foundations, design, and challenges," *Journal of Network and Computer Applications*, vol. 62, pp. 53–74, 2016.

[74] G. Technology, "Ids vs ips: Go-to tools for modern security stacks," https://www.garlandtechnology.com/blog/ids-vs-ips-go-to-tools-for-modern-security-stacks, July 2023, accessed: July 19, 2023.

[75] R. G. Bace, *Intrusion detection*. Sams Publishing, 2000.

[76] G. González-Granadillo, S. González-Zarzosa, and R. Diaz, "Security information and event management (siem): analysis, trends, and usage in critical infrastructures," *Sensors*, vol. 21, no. 14, p. 4759, 2021.

[77] I. R. Chiadighikaobi and J. Abdullah, "Comparative study between signature-based & anomaly-based network intrusion detection system (sbnids & abnids)," *DOI: 10.13140/RG.2.2.25015.16808*, 2015.

[78] C.-Y. Ho, Y.-D. Lin, Y.-C. Lai, I.-W. Chen, F.-Y. Wang, and W.-H. Tai, "False positives and negatives from real traffic with intrusion detection/prevention systems," *International Journal of Future Computer and Communication*, vol. 1, no. 2, p. 87, 2012.

[79] M. Almseidin, M. Alzubi, S. Kovacs, and M. Alkasassbeh, "Evaluation of machine learning algorithms for intrusion detection system," in *2017 IEEE 15th international symposium on intelligent systems and informatics (SISY)*. IEEE, 2017, pp. 000 277–000 282.

[80] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," *Expert Systems with Applications*, vol. 141, p. 112963, 2020.

[81] WeLiveSecurity, "Ids, firewall and antivirus: what you need to have installed?" https://www.welivesecurity.com/2015/04/30/ids-firewall-antivirus-need-installed/, 2015, accessed: July 19, 2023.

[82] M. Abdel-Basset, A. Gamal, K. M. Sallam, I. Elgendi, K. Munasinghe, and A. Jamalipour, "An optimization model for appraising intrusion-detection systems for network security communications: Applications, challenges, and solutions," *Sensors*, vol. 22, no. 11, p. 4123, 2022.

[83] LinkedIn, "What are the common challenges and solutions for ids/ips?" https://www.linkedin.com/advice/0/what-common-challenges-solutions-idsips, 2023, accessed: July 19, 2023.

[84] S. Jin, J.-G. Chung, and Y. Xu, "Signature-based intrusion detection system (ids) for in-vehicle can bus network," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021, pp. 1–5.

[85] T. Sommestad, H. Holm, and D. Steinvall, "Variables influencing the effectiveness of signature-based network intrusion detection systems," *Information Security Journal: A Global Perspective*, vol. 31, no. 6, pp. 711–728, 2022.

[86] Accedian, "What is the difference between signature-based and behavior-based intrusion detection systems?" https://accedian.com/blog/what-is-the-difference-between-signature-based-and-behavior-based-ids/, 2020, accessed: July 19, 2023.

[87] Computerworld, "Behavioral rules vs. signatures: Which should you use?" https://www.computerworld.com/article/2581345/behavioral-rules-vs--signatures--which-should-you-use-.html, 2003, accessed: July 19, 2023.

[88] N-able, "Intrusion detection system (ids): Signature vs. anomaly-based," https://www.n-able.com/blog/intrusion-detection-system, 2021, accessed: July 19, 2023.

[89] LinkedIn, "What are the advantages and disadvantages of signature-based and anomaly-based nids?" https://www.linkedin.com/advice/0/what-advantages-disadvantages-signature-based, 2023, accessed: July 19, 2023.

[90] M. Bhavsar, K. Roy, J. Kelly, and O. Olusola, "Anomaly-based intrusion detection system for iot application," *Discover Internet of Things*, vol. 3, no. 1, p. 5, 2023.

[91] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *computers & security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[92] A. B. Mohamed, N. B. Idris, and B. Shanmugum, "A brief introduction to intrusion detection system," in *Trends in Intelligent Robotics, Automation, and Manufacturing: First International Conference, IRAM 2012, Kuala Lumpur, Malaysia, November 28-30, 2012. Proceedings*. Springer, 2012, pp. 263–271.

[93] G. Vigna and C. Kruegel, *Host-based intrusion detection*. na, 2006.

[94] S. Kumar, S. Gupta, and S. Arora, "Research trends in network-based intrusion detection systems: A review," *IEEE Access*, vol. 9, pp. 157 761–157 779, 2021.

[95] P. Mahalingam, "Intelligent network-based intrusion detection system (inids)," in *Advances in Computing and Information Technology: Proceedings of the Second International Conference on Advances in Computing and Information Technology (ACITY) July 13-15, 2012, Chennai, India-Volume 1*. Springer, 2012, pp. 1–9.

[96] Securitywing, "Host based ids vs network based ids," https://securitywing.com/host-based-ids-vs-network-based-ids/, 2012, accessed: July 19, 2023.

[97] 24hSolutions, "Advantages and disadvantages of hids," http://solutions24h.com/advantages-of-hids/, 2022, accessed: July 19, 2023.

[98] L. Consulting, "Host-based vs network-based intrusion detection system (ids)," https://logixconsulting.com/2022/05/03/host-based-vs-network-based-intrusion-detection-system-ids/, 2022, accessed: July 19, 2023.

[99] T. T. NEW, "Benefits of using a host-based intrusion detection system," https://www.tothenew.com/blog/benefits-of-using-a-host-based-intrusion-detection-system/, December 2016.

[100] S. M. Othman, N. T. Alsohybe, F. M. Ba-Alwi, and A. T. Zahary, "Survey on intrusion detection system types," *International Journal of Cyber-Security and Digital Forensics*, vol. 7, no. 4, pp. 444–463, 2018.

[101] A. N. Cahyo, A. K. Sari, and M. Riasetiawan, "Comparison of hybrid intrusion detection system," in *2020 12th International Conference on Information Technology and Electrical Engineering (ICITEE)*. IEEE, 2020, pp. 92–97.

[102] A. R. Chavez, C. F. Lai, N. Jacobs, S. Hossain-McKenzie, C. B. Jones, J. B. Johnson, and A. Summers, "Hybrid intrusion detection system design for distributed energy resources." Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2019.

[103] N. M. Sahar, S. Sari, N. Taujuddin *et al.*, "Intrusion-detection system based on hybrid models," in *IOP conference series: materials science and engineering*, vol. 917, no. 1. IOP Publishing, 2020, p. 012059.

[104] S. Khonde and V. Ulagamuthalvi, "Hybrid intrusion detection system using blockchain framework," *EURASIP Journal on Wireless Communications and Networking*, vol. 2022, no. 1, p. 58, 2022.

[105] P. Chauhan and N. Chandra, "A review on hybrid intrusion detection system using artificial immune system approaches," *International Journal of Computer Applications*, vol. 68, no. 20, 2013.

[106] Institute for Automation and Applied Informatics, "Hybrid intrusion detection system for smart grids," https://www.iai.kit.edu/english/464_4278.php.

[107] Spiceworks, "Ids vs ips: What's the difference?" https://www.spiceworks.com/it-security/network-security/articles/ids-vs-ips/amp/, June 2019.

[108] E. Alpaydin, *Introduction to machine learning*. MIT press, 2010.

[109] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[110] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[111] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[112] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[113] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[114] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[115] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection using data mining techniques: A systematic review," *IEEE transactions on dependable and secure computing*, vol. 13, no. 5, pp. 654–669, 2015.

[116] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 1985–1992.

[117] P. Domingos, "The master algorithm: How the quest for the ultimate learning machine will remake our world," *Basic Books*, 2015.

[118] J. Brownlee, "Types of learning in machine learning," https://machinelearningmastery.com/types-of-learning-in-machine-learning/, January 2019.

[119] A. Kumar, "Types of machine learning algorithms you should know," https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861, June 2020.

[120] T.-N. Hoang and D. Kim, "Detecting in-vehicle intrusion via semi-supervised learning-based convolutional adversarial autoencoders," *Vehicular Communications*, vol. 38, p. 100520, 2022.

[121] TechTarget, "Logistic regression," https://www.techtarget.com/searchbusinessanalytics/definition/logistic-regression.

[122] G. Seif, "A guide to decision trees for machine learning and data science," https://towardsdatascience.com/a-guide-to-decision-trees-for-machine-learning-and-data-science-fe2607241956, November 2018.

[123] QuantStart, "Beginner's guide to decision trees for supervised machine learning," https://www.quantstart.com/articles/Beginners-Guide-to-Decision-Trees-for-Supervised-Machine-Learning/.

[124] C. Validated, "Why are decision trees considered supervised learning?" https://stats.stackexchange.com/questions/380069/why-are-decision-trees-considered-supervised-learning, December 2018.

[125] B. In, "Random forest algorithm: How it works and why it matters," https://builtin.com/data-science/random-forest-algorithm, May 2021.

[126] Simplilearn, "Random forest algorithm," https://www.simplilearn.com/tutorials/machine-learning-tutorial/random-forest-algorithm.

[127] A. Gupta, "Understanding random forest," https://towardsdatascience.com/understanding-random-forest-58381e0602d2, July 2018.

[128] S. Mishra, "Unsupervised learning and data clustering," https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a, May 2017.

[129] JavaTpoint, "Unsupervised machine learning," https://www.javatpoint.com/unsupervised-machine-learning.

[130] V. Kotu and B. Deshpande, *Data science: concepts and practice*. Morgan Kaufmann, 2018.

[131] B. In, "Unsupervised learning," https://builtin.com/machine-learning/unsupervised-learning.

[132] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine learning*, vol. 109, no. 2, pp. 373–440, 2020.

[133] AltexSoft, "Semi-supervised learning: Definition, algorithms, and applications," https://www.altexsoft.com/blog/semi-supervised-learning/, June 2021.

[134] X. J. Chen, L. M. Collins, and B. O. Mainsah, "Language model-guided classifier adaptation for brain-computer interfaces for communication," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2022, pp. 1642–1647.

[135] M. Boschini, P. Buzzega, L. Bonicelli, A. Porrello, and S. Calderara, "Continual semi-supervised learning through contrastive interpolation consistency," *Pattern Recognition Letters*, vol. 162, pp. 9–14, 2022.

[136] B. Kolosnjaji, "Machine learning for anomaly detection under constraints," Ph.D. dissertation, Technische Universität München, 2019.

[137] A. Networks, "Anomaly detection," https://avinetworks.com/glossary/anomaly-detection/.

[138] P. Ntambu and S. A. Adeshina, "Machine learning-based anomalies detection in cloud virtual machine resource usage," in *2021 1st International Conference on Multidisciplinary Engineering and Applied Science (ICMEAS)*. IEEE, 2021, pp. 1–6.

[139] A. Adnan, A. Muhammed, A. A. Abd Ghani, A. Abdullah, and F. Hakim, "An intrusion detection system for the internet of things based on machine learning: Review and challenges," *Symmetry*, vol. 13, no. 6, p. 1011, 2021.

[140] A. B. Nassif, M. A. Talib, Q. Nasir, and F. M. Dakalbab, "Machine learning for anomaly detection: A systematic review," *Ieee Access*, vol. 9, pp. 78 658–78 700, 2021.

[141] J. Brownlee, "Anomaly detection with isolation forest and kernel density estimation," https://machinelearningmastery.com/anomaly-detection-with-isolation-forest-and-kernel-density-estimation/, January 2022.

[142] ——, "Isolation forest — auto anomaly detection with python," https://towardsdatascience.com/isolation-forest-auto-anomaly-detection-with-python-e7a8559d4562, September 2022.

[143] ——, "How to perform anomaly detection with the isolation forest algorithm," https://towardsdatascience.com/how-to-perform-anomaly-detection-with-the-isolation-forest-algorithm-e8c8372520bc, November 2021.

[144] M. Weber, S. Klug, E. Sax, and B. Zimmer, "Embedded hybrid anomaly detection for automotive can communication," in *9th European Congress on Embedded Real Time Software and Systems (ERTS 2018)*, 2018.

[145] M. Alam, "Support vector machine (svm) for anomaly detection," https://towardsdatascience.com/support-vector-machine-svm-for-anomaly-detection-73a8d676c331, October 2020.

[146] C. Bigoni and J. S. Hesthaven, "Simulation-based anomaly detection and damage localization: an application to structural health monitoring," *Computer Methods in Applied Mechanics and Engineering*, vol. 363, p. 112896, 2020.

[147] C. Wang, Y. Sun, S. Lv, C. Wang, H. Liu, and B. Wang, "Intrusion detection system based on one-class support vector machine and gaussian mixture model," *Electronics*, vol. 12, no. 4, p. 930, 2023.

[148] G. Huang, J. Chen, and L. Liu, "One-class svm model-based tunnel personnel safety detection technology," *Applied Sciences*, vol. 13, no. 3, p. 1734, 2023.

[149] W. Zhang, L. Du, L. Li, X. Zhang, and H. Liu, "Infinite bayesian one-class support vector machine based on dirichlet process mixture clustering," *Pattern Recognition*, vol. 78, pp. 56–78, 2018.

[150] GeeksforGeeks, "Introduction to dimensionality reduction," https://www.geeksforgeeks.org/dimensionality-reduction/, May 2023.

[151] I. Jolliffe, "Principal component analysis (pp. 1094-1096)," *Springer Berlin Heidelberg. RESUME SELİN DEĞİRMECİ Marmara University, Goztepe Campus ProQuest Number: ProQuest). Copyright of the Dissertation is held by the Author. All Rights Reserved*, vol. 28243034, p. 28243034, 2011.

[152] A. B. Coleman, "Feature extraction using dimensionality reduction techniques: Capturing the human perspective," 2015.

[153] KDnuggets, "Dimensionality reduction techniques in data science," https://www.kdnuggets.com/2022/09/dimensionality-reduction-techniques-data-science.html, September 2022.

[154] C. O. S. Sorzano, J. Vargas, and A. P. Montano, "A survey of dimensionality reduction techniques," *arXiv preprint arXiv:1403.2877*, 2014.

[155] U. M. Khaire and R. Dhanalakshmi, "Stability of feature selection algorithm: A review," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 4, pp. 1060–1073, 2022.

[156] M. Kuhn, K. Johnson, M. Kuhn, and K. Johnson, "An introduction to feature selection," *Applied predictive modeling*, pp. 487–519, 2013.

[157] T. Boyle, "Feature selection and dimensionality reduction," https://towardsdatascience.com/feature-selection-and-dimensionality-reduction-f488d1a035de, March 2019.

[158] A. Desarda, "Getting data ready for modelling: Feature engineering, feature selection, dimension reduction (part two)," https://towardsdatascience.com/getting-data-ready-for-modelling-feature-engineering-feature-selection-dimension-reduction-39c December 2018.

[159] M. Dawood, "Feature selection and dimensionality reduction techniques," https://www.linkedin.com/pulse/feature-selection-dimensionality-reduction-techniques-muhammad-dawood, June 2023.

[160] R. Kohavi and G. John, "Wrappers for feature subset selection, artificial intelligence, vol. 97, no. 1-2," *1997*, vol. 273324, 1997.

[161] G. Borboudakis and I. Tsamardinos, "Forward-backward selection with early dropping," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 276–314, 2019.

[162] M. L. Bermingham, R. Pong-Wong, A. Spiliopoulou, C. Hayward, I. Rudan, H. Campbell, A. F. Wright, J. F. Wilson, F. Agakov, P. Navarro *et al.*, "Application of high-dimensional feature selection: evaluation for genomic prediction in man," *Scientific reports*, vol. 5, no. 1, p. 10312, 2015.

[163] J. Laborda and S. Ryoo, "Feature selection in a credit scoring model. mathematics 9, 746," 2021.

[164] R. Panthong and A. Srivihok, "Wrapper feature subset selection for dimension reduction based on ensemble learning algorithm," *Procedia Computer Science*, vol. 72, pp. 162–169, 2015.

[165] Y. Bouchlaghem, Y. Akhiat, and S. Amjad, "Feature selection: a review and comparative study," in *E3S Web of Conferences*, vol. 351. EDP Sciences, 2022, p. 01046.

[166] J. Miao and L. Niu, "A survey on feature selection," *Procedia computer science*, vol. 91, pp. 919–926, 2016.

[167] M. B. Imani, M. R. Keyvanpour, and R. Azmi, "A novel embedded feature selection method: a comparative study in the application of text categorization," *Applied Artificial Intelligence*, vol. 27, no. 5, pp. 408–427, 2013.

[168] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Icml*, vol. 97, no. 412-420. Nashville, TN, USA, 1997, p. 35.

[169] DataAspirant, "Feature selection methods in machine learning," https://dataaspirant.com/feature-selection-methods-machine-learning/.

[170] KDnuggets, "Feature selection – all you ever wanted to know," https://www.kdnuggets.com/2021/06/feature-selection-overview.html, June 2021.

[171] S. Biswas, M. Bordoloi, and B. Purkayastha, "Review on feature selection and classification using neuro-fuzzy approaches," *International Journal of Applied Evolutionary Computation (IJAEC)*, vol. 7, no. 4, pp. 28–44, 2016.

[172] W. Jia, M. Sun, J. Lian, and S. Hou, "Feature dimensionality reduction: a review," *Complex & Intelligent Systems*, vol. 8, no. 3, pp. 2663–2693, 2022.

[173] I. Jolliffe, "Principal component analysis: Wiley online library," *Google Scholar*, 2005.

[174] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction.* Springer, 2009, vol. 2.

[175] B. In, "Step-by-step explanation of principal component analysis," https://builtin.com/data-science/step-step-explanation-principal-component-analysis, July 2023.

[176] M. Miller, "The basics: Principal component analysis," https://towardsdatascience.com/the-basics-principal-component-analysis-83c270f1a73c, January 2020.

[177] LinkedIn, "Challenges and limitations of using pca for data analysis," https://www.linkedin.com/advice/0/what-some-challenges-limitations-using-principal, n.d.

[178] Keboola, "Pca and machine learning: A match made in heaven," 09 2021. [Online]. Available: https://www.keboola.com/blog/pca-machine-learning

[179] C. de Datos, "Análisis de componentes principales (pca)," https://cienciadedatos.net/documentos/35_principal_component_analysis, 03 2022, accessed on July 27, 2023.

[180] A. Tharwat, T. Gaber, A. Ibrahim, and A. E. Hassanien, "Linear discriminant analysis: A detailed tutorial," *AI communications*, vol. 30, no. 2, pp. 169–190, 2017.

[181] D. Sachin *et al.*, "Dimensionality reduction and classification through pca and lda," *International journal of computer Applications*, vol. 122, no. 17, 2015.

[182] R. N. Khushaba, A. Al-Ani, A. Al-Jumaily, and H. T. Nguyen, "A hybrid nonlinear-discriminant analysis feature projection technique," in *AI 2008: Advances in Artificial Intelligence: 21st Australasian Joint Conference on Artificial Intelligence Auckland, New Zealand, December 1-5, 2008. Proceedings 21.* Springer, 2008, pp. 544–550.

[183] J. Teenbaum, D. Silva, and J. Langford, "global geometric framework for nonlinear dimensionality reduction [j]," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[184] Y. Yang, H. Sun, J. Gong, Y. Du, and D. Yu, "Interpretable dimensionality reduction by feature preserving manifold approximation and projection," *arXiv preprint arXiv:2211.09321*, 2022.

[185] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[186] B. Dharamsotu, K. S. Rani, S. A. Moiz, and C. R. Rao, "k-nn sampling for visualization of dynamic data using lion-tsne," in *2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC).* IEEE, 2019, pp. 63–72.

[187] J. Díaz-Verdejo, J. Muñoz-Calle, A. Estepa Alonso, R. Estepa Alonso, and G. Madinabeitia, "On the detection capabilities of signature-based intrusion detection systems in the context of web attacks," *Applied Sciences*, vol. 12, no. 2, p. 852, 2022.

[188] S. Jacobsen, "Attacking artificial intelligence: Ai's security vulnerability and what we can do about it," *IT Professional*, vol. 20, no. 6, pp. 20–26, 2018.

[189] "Snort," https://www.snort.org/, accessed: October 30, 2023.

[190] M. Sarhan, S. Layeghy, M. Gallagher, and M. Portmann, "From zero-shot machine learning to zero-day attack detection," *International Journal of Information Security*, pp. 1–13, 2023.

[191] N. Peppes, T. Alexakis, E. Adamopoulou, and K. Demestichas, "The effectiveness of zero-day attacks data samples generated via gans on deep learning classifiers," *Sensors*, vol. 23, no. 2, p. 900, 2023.

[192] I. Mbona and J. H. Eloff, "Detecting zero-day intrusion attacks using semi-supervised machine learning approaches," *IEEE Access*, vol. 10, pp. 69 822–69 838, 2022.

[193] P. Pitre, A. Gandhi, V. Konde, R. Adhao, and V. Pachghare, "An intrusion detection system for zero-day attacks to reduce false positive rates," in *2022 International Conference for Advancement in Technology (ICONAT).* IEEE, 2022, pp. 1–6.

[194] S. Ali, S. U. Rehman, A. Imran, G. Adeem, Z. Iqbal, and K.-I. Kim, "Comparative evaluation of ai-based techniques for zero-day attacks detection," *Electronics*, vol. 11, no. 23, p. 3934, 2022.

[195] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, and O. Jogunola, "Federated deep learning for zero-day botnet attack detection in iot-edge devices," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3930–3944, 2021.

[196] B. Drozdenko and M. Powell, "Utilizing deep learning techniques to detect zero day exploits in network traffic flows," in *2022 IEEE 13th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 2022, pp. 0163–0172.

[197] M. Soltani, B. Ousat, M. J. Siavoshani, and A. H. Jahangir, "An adaptable deep learning-based intrusion detection system to zero-day attacks," *arXiv preprint arXiv:2108.09199*, 2021.

[198] S.-J. Bu and S.-B. Cho, "Integrating deep learning with first-order logic programmed constraints for zero-day phishing attack detection," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 2685–2689.

[199] T. Zoppi, A. Ceccarelli, and A. Bondavalli, "Unsupervised algorithms to detect zero-day attacks: Strategy and application," *Ieee Access*, vol. 9, pp. 90 603–90 615, 2021.

[200] H. Hindy, R. Atkinson, C. Tachtatzis, J.-N. Colin, E. Bayne, and X. Bellekens, "Utilising deep learning techniques for effective zero-day attack detection," *Electronics*, vol. 9, no. 10, p. 1684, 2020.

[201] N. Sameera and M. Shashi, "Deep transductive transfer learning framework for zero-day attack detection," *ICT Express*, vol. 6, no. 4, pp. 361–367, 2020.

[202] R. Tang, Z. Yang, Z. Li, W. Meng, H. Wang, Q. Li, Y. Sun, D. Pei, T. Wei, Y. Xu *et al.*, "Zerowall: Detecting zero-day web attacks through encoder-decoder recurrent neural networks," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2479–2488.

[203] A. Blaise, M. Bouet, V. Conan, and S. Secci, "Detection of zero-day attacks: An unsupervised port-based approach," *Computer Networks*, vol. 180, p. 107391, 2020.

[204] H. Wang, H. Sayadi, G. Kolhe, A. Sasan, S. Rafatirad, and H. Homayoun, "Phased-guard: Multi-phase machine learning framework for detection and identification of zero-day microarchitectural side-channel attacks," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*. IEEE, 2020, pp. 648–655.

[205] Q. Zhou and D. Pezaros, "Evaluation of machine learning classifiers for zero-day intrusion detection–an analysis on cic-aws-2018 dataset," *arXiv preprint arXiv:1905.03685*, 2019.

[206] F. Abri, S. Siami-Namini, M. A. Khanghah, F. M. Soltani, and A. S. Namin, "Can machine/deep learning classifiers detect zero-day malware with high accuracy?" in *2019 IEEE international conference on big data (Big Data)*. IEEE, 2019, pp. 3252–3259.

[207] K. Radhakrishnan, R. R. Menon, and H. V. Nath, "A survey of zero-day malware attacks and its detection methodology," in *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*. IEEE, 2019, pp. 533–539.

[208] Z. Hu, P. Chen, M. Zhu, and P. Liu, "Reinforcement learning for adaptive cyber defense against zero-day attacks," *Adversarial and Uncertain Reasoning for Adaptive Cyber Defense: Control-and Game-Theoretic Approaches to Cyber Security*, pp. 54–93, 2019.

[209] J.-Y. Kim, S.-J. Bu, and S.-B. Cho, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," *Information Sciences*, vol. 460, pp. 83–102, 2018.

[210] W. Jung, S. Kim, and S. Choi, "Poster: deep learning for zero-day flash malware detection," in *36th IEEE symposium on security and privacy*, vol. 10, 2015, pp. 2 809 695–2 817 880.

[211] H. Holm, "Signature based intrusion detection for zero-day attacks:(not) a closed chapter?" in *2014 47th Hawaii international conference on system sciences*. IEEE, 2014, pp. 4895–4904.

[212] P. V. Amoli, T. Hamalainen, G. David, M. Zolotukhin, and M. Mirzamohammad, "Unsupervised network intrusion detection systems for zero-day fast-spreading attacks and botnets," *JDCTA (International Journal of Digital Content Technology and its Applications*, vol. 10, no. 2, pp. 1–13, 2016.

[213] Y. Guo, "A review of machine learning-based zero-day attack detection: Challenges and future directions," *Computer Communications*, 2022.

[214] K. Cup, "Kdd cup 1999 data," https://www.kdd.org/kdd-cup/view/kdd-cup-1999/Data, 1999, accessed: July 01, 2023.

[215] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, 2009, pp. 1–6.

[216] U. of New Brunswick, "NSL-KDD data set," https://www.unb.ca/cic/datasets/nsl.html, 2009, [Online; accessed 01-July-2023].

[217] Y. Tang, L. Gu, and L. Wang, "Deep stacking network for intrusion detection," *Sensors*, vol. 22, no. 1, p. 25, 2021.

[218] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 4, pp. 262–294, 2000.

[219] S. Choudhary and N. Kesswani, "Analysis of kdd-cup'99, nsl-kdd and unsw-nb15 datasets using deep learning in iot," *Procedia Computer Science*, vol. 167, pp. 1561–1573, 2020.

[220] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set," *Information Security Journal: A Global Perspective*, vol. 25, no. 1-3, pp. 18–31, 2016.

[221] R. Panigrahi and S. Borah, "A detailed analysis of cicids2017 dataset for designing intrusion detection systems," *International Journal of Engineering & Technology*, vol. 7, no. 3.24, pp. 479–482, 2018.

[222] G. Yang, X. Chen, Y. Zhou, and C. Yu, "Dualsc: Automatic generation and summarization of shellcode via transformer and dual learning," in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2022, pp. 361–372.

[223] M. Xu, J. Guo, H. Yuan, and X. Yang, "Zero-trust security authentication based on spa and endogenous security architecture," *Electronics*, vol. 12, no. 4, p. 782, 2023.

[224] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.

[225] S. Bagui, E. Kalaimannan, S. Bagui, D. Nandi, and A. Pinto, "Using machine learning techniques to identify rare cyber-attacks on the unsw-nb15 dataset," *Security and Privacy*, vol. 2, no. 6, p. e91, 2019.

[226] Cybersecurity and Infrastructure Security Agency. (2021) Understanding denial-of-service attacks. [Online]. Available: https://www.cisa.gov/news-events/news/understanding-denial-service-attacks

[227] R. Pujari, "Network attack detection and classification using machine learning models based on unsw-nb15 data-set," *Medium*, 2020.

[228] B. Bonev, *Feature selection based on information theory*. Universidad de Alicante, 2010.

[229] D. V. Catalogue, "Density plot," https://datavizcatalogue.com/methods/density_plot.html, 2023, accessed: October 4, 2023.

[230] Z. Zoghi and G. Serpen, "Unsw-nb15 computer security dataset: Analysis through visualization," *arXiv preprint arXiv:2101.05067*, 2021.

[231] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Isolationforest," https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html, 2021, accessed: September 4, 2023.

[232] K. Sadaf and J. Sultana, "Intrusion detection based on autoencoder and isolation forest in fog computing," *IEEE Access*, vol. 8, pp. 167 059–167 068, 2020.

Appendices

Table A1: Description of the features of the UNSW-NB15 dataset

Id	Name	Type	Feature	Description
1	srcip	numerical	Flow	Source IP address
2	sport	numerical	Flow	Source port
3	dstip	numerical	Flow	Destination IP address
4	dsport	numerical	Flow	Destination port
5	proto	categorical	Flow	Network protocol used to establish the connection
6	state	numerical	Basic	Represents the connection state (e.g., established, closed, reset)
7	dur	numerical	Basic	Duration of the connection
8	sbytes	numerical	Basic	Total number of bytes sent by the source device in the connection
9	dbytes	numerical	Basic	Total number of bytes received by the destination device in the connection
10	sttl	numerical	Basic	Source time-to-live value set in the IP header of the source device
11	dttl	numerical	Basic	Destination time-to-live value set in the IP header of the destination device
12	sloss	numerical	Basic	Number of lost packets in the source direction of the connection
13	dloss	numerical	Basic	Number of lost packets in the destination direction of the connection
14	service	categorical	Basic	Service type or protocol used in the connection
15	sload	numerical	Basic	Source load or amount of traffic originating from the source device
16	dload	numerical	Basic	Destination load or amount of traffic received by the destination device
17	spkts	numerical	Basic	Number of packets sent by the source device in the connection

18	dpkts	numerical	Basic	Number of packets received by the destination device in the connection
19	swin	numerical	Content	Source window size
20	dwin	numerical	Content	Destination window size
21	stcpb	numerical	Content	Source cumulative bytes
22	dtcpb	numerical	Content	Destination cumulative bytes
23	smeans	numerical	Content	Source mean packet size
24	dmeans	numerical	Content	Destination mean packet size
25	trans_depth	numerical	Content	Transmission depth in the connection
26	res_bdy_len	numerical	Content	Response body length in the connection
27	sjit	numerical	Time	Source jitter
28	djit	numerical	Time	Destination jitter
29	stime	numerical	Time	Source time
30	ltime	numerical	Time	Last time
31	sintpkt	numerical	Time	Source interpacket arrival time
32	dintpkt	numerical	Time	Destination interpacket arrival time
33	tcprtt	numerical	Time	TCP round-trip time
34	synack	numerical	Time	SYN-ACK response time
35	ackdat	numerical	Time	ACK-data response time
36	is_sm_ips_ports	numerical	Additional Generated	Indicates whether the source and destination IP addresses and ports are the same in the connection
37	ct_state_ttl	numerical	Additional Generated	Represents the number of connections with the same source IP address, destination IP address, and TTL value
38	ct_flw_http_mthd	numerical	Additional Generated	Denotes the number of flows that have the same source and destination IP addresses, and the same HTTP method

39	is_ftp_login	numerical	Additional Generated	Indicates whether an FTP login was attempted in the connection
40	ct_ftp_cmd	numerical	Additional Generated	Represents the number of FTP commands in the connection
41	ct_srv_src	numerical	Additional Generated	Indicates the number of connections to the same service and source IP address
42	ct_srv_dst	numerical	Additional Generated	Denotes the number of connections to the same service and destination IP address
43	ct_dst_ltm	numerical	Additional Generated	No. of records of the same dstip (3) in 100 records according to the ltime (26)
44	ct_src_ltm	numerical	Additional Generated	No. of records of the srcip (1) in 100 records according to the ltime (26)
45	ct_src_dport_ltm	numerical	Additional Generated	No of records of the same srcip (1) and the dsport (4) in 100 records according to the ltime (26)
46	ct_dst_sport_ltm	numerical	Additional Generated	No of records of the same dstip (3) and the sport (2) in 100 records according to the ltime (26)
47	ct_dst_src_ltm	numerical	Additional Generated	No of records of the same srcip (1) and the dstip (3) in in 100 records according to the ltime (26)
48	attack_cat	numerical	Labelled	Specifies the specific category or type of attack associated with the network connection if it is classified as an attack
49	label	categorical	Labelled	Represents the ground truth label for the network connection, indicating whether it is classified as a normal or an attack