People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

IBN KHALDOUN UNIVERSITY OF TIARET

Dissertation

Presented to:

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
DEPARTEMENT OF COMPUTER SCIENCE

in order to obtain the degree of :

MASTER

Specialty:

Artificial Intelligence and Digitalization

Presented by:

Benamer Belkacem Nawel

On the theme:

Alerts Correlation by attention mechanisms

Defended publicly on / /2025 in Tiaret in front the jury composed of:

Mr DAOUD Mohamed Amine	MCB Tiaret University	Chairman
Mr MOSTEFAOUI Sidahmed	MCA Tiaret University	Supervisor
Mr BOUGUESSA Aek	MAA Tiaret University	Examiner

2024-2025

Acknowledgements

In the name of Allah, the Most Gracious, the Most Merciful. Praise be to God who has granted me the strength, patience, knowledge, and success to complete this work. Praise is to Allah by whose grace good deeds are completed.

I am eternally indebted to my loving little family, my parents and my two brothers Abderrahman and Ali, for their unwavering support, encouragement, and patience throughout my academic journey. Furthermore, I would like to acknowledge the guidance and supervision of Dr. Mostefaoui SidAhmed, whose constructive criticism, valuable suggestions, and constant motivation greatly contributed to the success of this research project. I also extend my sincere thanks to the members of the jury for their time and attention to this work. Finally, I would like to thank all my family and friends for their prayers and best wishes for me to success my studies.

May Allah reward all those who have supported me in this endeavor.

Abstract

From the inception of intrusion detection systems to the present day, all IDSs have continuously generated generate large numbers of alerts, many of which are false positives. This leads to an overwhelming amount of data for security analysts, leading to missed detections or delayed detections of real threats. Our solution is an alert correlation framework with an attention mechanism that will produce lower false positives. Through this process, the model learns dependencies and contextual relationships between features of alerts, indicating that the model can better distinguish between malicious and benign traffic. Unlike the previous approaches, our approach does not rely on prior knowledge of specific attacks or predefined normal behavior, making it robust and adaptable to unknown or evolving threats. By modeling feature relationships contextually, the proposed method enhances detection accuracy while significantly reducing the number of false positives.

Keywords: Network Security, Intrusion Detection Systems, Alert Correlation, False positives, Deep Learning, Attention mechanism, Transformer.

ملخص

منذ نشأة أنظمة كشف التسلل وحتى يومنا هذا، ظلت هذه الأنظمة تُولّد عددًا كبيرًا من التنبيهات، ويُعدّ العديد منها إنذارات كاذبة. هذا يؤدي إلى تراكم كمّ هائل من البيانات على محللي الأمن، مما يسبب تأخيرًا أو فواتًا في اكتشاف التهديدات الحقيقية. نقترح في هذا العمل إطارًا لربط التنبيهات يعتمد على آلية الانتباه، يهدف إلى تقليل عدد الإنذارات الكاذبة. من خلال هذه الآلية، يتعلّم النموذج العلاقات والارتباطات السياقية بين خصائص التنبيهات، مما يمكّنه من التمييز بدقة أكبر بين الحركة الخبيثة والسليمة. بخلاف الأساليب التقليدية، لا يعتمد هذا النهج على معرفة مسبقة بأنواع الهجمات أو على سلوك طبيعي محدد، مما يجعله مرنًا وقادرًا على التكيف مع التهديدات الجديدة والمتغيّرة. ومن خلال فهم العلاقات بين الخصائص في سياقها، يُسهم الأسلوب المقترح في تحسين دقة الاكتشاف، ويُقلّل بدرجة كبيرة من عدد الإنذارات الكاذبة.

الكلمات المفتاحية: أمن الشبكات، أنظمة كشف التسلل، ربط التنبيهات، الإنذارات الكاذبة، التعلم العميق، آلية الانتباه، النموذج التحويلي.

Contents

List of Figures

List of Tables

Acronyms

IDS Intrusion detection system

DOS Denial of Service

U2R User to Root

R2L Remote to Local

IDPSs Intrusion Detection and Prevention Systems

IPS Intrusion Prevention System

NIDS Network based IDS

HIDS Host based IDS

IDMEF Intrusion Detection Message Exchange Format

ML Machine Learning

DL Deep Learning

ANN Artificial Neural Network

AI Artificial intelligence

MLP Multilayer perceptron

RNN Recurrent Neural Networks

KNN K-Nearest Neighbors

SVM Support vector machine

NLP Natural Language Processing

General Introduction

Context and Background

The explosive increase in the number of interconnected machines and the widespread use of the Internet in organizational environments has significantly expanded the attack surface of modern networks. The increase in the size and sophistication of a network leads to more frequent and advanced threats. A cyber threat refers to any potentially malicious action that aims to damage, disrupt, or gain unauthorized access to computer systems, networks, or data.

In response to these growing threats Intrusion Detection Systems (IDSs) have been developed as a key layer of defense. IDS is a security solution that monitors network traffic and system activities to detect suspicious behavior or potential security breaches. It alerts administrators about possible threats but does not take direct action to prevent them [6] .There are two main types of IDS approaches: misuse-based detection, which relies on known attack signatures, and anomaly-based detection, which identifies deviations from normal behavior [9]. Despite their usefulness, IDSs often suffer from a major drawback: the generation of a high number of false positives (benign activities incorrectly flagged as threats) [24]. In fact, it has been estimated that up to 99% of alerts [49].

These false positives overwhelm security analysts, making it difficult to identify real attacks in time. To alleviate this issue, alert correlation techniques have been proposed [48]. Alert correlation is a process that analyzes the alerts produced by one or more intrusion detection systems and provides a more succinct and high-level view of occurring or attempted intrusions, offering a more meaningful view of security events. It helps filter out irrelevant alerts and group related ones, improving the decision-making process for security teams.

Recent research has examined the use of machine learning (ML) and deep learning (DL) for improving alert correlation. These methods add the ability to learn from data and identify complex patterns, generally without heavy reliance on predetermined rules or attack signatures. However, many traditional techniques, and even first-generation DL techniques had a reasonable, but inadequate, response to false positive reduction when there was limited labeled data or unseen attack types.

The attention mechanism has become a staple of artificial intelligence recently due to its unique power to focus on the most relevant aspects of the input data. Attention is a deep

learning method that allows models to dynamically focus on the most relevant parts of the input data, as they are completing a task . Instead of processing all input information equally, attention assigns different weights to different components of the input data, allowing the model to focus on the most informative elements [43]. Attention mechanisms were first implemented in Natural Language Processing (NLP) and have been integrated into Transformers architecture boosting machine translation, text summarization, and question answering to stratospheric levels of performance. This success has catalyzed research in computer vision, speech recognition, and bioinformatics. The primary benefit attention brings is the ability to capture contextual dependencies in data irrespective of the distance or spatial arrangement. This is very useful for problems like alert correlation in intrusion detection, where the inter dependability of various features and events requires keen comprehension.

Problem Statement

Classic methods of detection and alert correlation often rely on set rules or predefined attack patterns [17]. As a result, they are inflexible, and do not generalize well to new or developmental forms of threat. Similarly, while deep learning has some success in learning patterns directly from data, many existing models continue to struggle to efficiently reduce false positives as they often rely on large labeled datasets and lack mechanisms to understand relationships between features in a nuanced way. In short, the question that motivates this thesis is:

"*How can we develop an effective deep learning approach, based on attention mechanisms, to reduce false positives in IDS alert correlation without requiring predefined attack signatures or normal traffic models?*"

Proposed Approach

To address the problem, we suggest using Transformer Encoder architecture with an attention mechanism. Unlike previous approaches, our method implements an attention mechanism to capture complex dependencies and relevance among features, allowing the model to separate true threats from benign events more reliably. Using the well-known NSL-KDD dataset, we show that this architecture improves alert correlation and decreases false positives remarkably, which enables more efficient intrusion detection.

Outline

This thesis consists of four chapters in addition to a general introduction and a general conclusion:

- **Chapter I: Network Security**

  We will introduce the basics of computer security and cover the field of intrusion detection in more detail.

- **Chapter II: Alert Correlation**

  An introductory exploration into the theoretical and practical aspects of alert correlation, with the primary objective of establishing a foundational understanding and offering a comprehensive perspective on its goals, challenges, and classification.

- **Chapter III: Machine Learning and Deep Learning**

  In this chapter, we introduce the basic principles of ML, covering artificial intelligence (AI), ML categories, and tasks. Furthermore, we discuss the foundational concepts of DL, including artificial neural networks (ANN), key concepts, and attention mechanism, and the transformers.

- **Chapter IV: Alert Correlation Using Attention Mechanism**

  We delve deeply into the practical implementation of transformer encoder for alert correlation. Firstly, it introduces the existing methods used in alert correlation, followed by the description of the proposed model. It then details the utilized datasets and baselines. Finally, a comparative analysis is conducted between the results of the baselines and the proposed TransformerEncoder model, accompanied by a comprehensive discussion of the obtained outcomes.

Chapter I

Network Security

I.1 Introduction

As businesses increasingly rely on digital networks, the risk of cyber threats grows. Computer security plays a pivotal role in protecting corporate networks from malicious intrusions. Security systems are designed to prevent attacks, identify vulnerabilities, and mitigate risks through proactive monitoring and detection. Understanding attackers' methods and intrusive activities is essential for developing effective defense strategies. By employing diagnostic and detection tools, organizations can enhance their security posture, identify threats in real time, and take necessary corrective actions to strengthen their overall cybersecurity framework.

This chapter explores the fundamental aspects of network security and cyber threats. It begins by examining network security, highlighting its role in protecting corporate information systems. The discussion then shifts to cyberattacks, detailing their stages and various classifications to provide a deeper understanding of attack methodologies.

To counter these threats, the chapter explores network security solutions, focusing on strategies and technologies used to defend against cyber intrusions. Finally, it delves into Intrusion Detection Systems (IDS), explaining their significance in monitoring network activities, detecting malicious actions, and enhancing overall cybersecurity.

I.2 Network Security

The network security is a crucial field that aims to protect the infrastructure, data, and communications of computer systems against threats, attacks, and unauthorized access. It focuses on establishing a secure environment for devices, applications, and users to operate safely and efficiently. In the context of computing, the British Standards Institute defines information security as the preservation of three fundamental properties [1]. When developing a secure network, these properties need to be carefully considered:

- **Confidentiality:** this involves avoiding illegal sharing of data by making sure that information is only available to authorized users
- **Integrity:** this involves avoiding illegal data change and guaranteeing the completeness and accuracy of the information.

- **Availability:** this involves preventing the general use of computer resources by guaranteeing that authorized users have constant access to information

Furthermore, other concepts are often considered to be part of the taxonomy of network security:

- **Authentication:** verify that the network's users are who they claim to be
- **Non-repudiation:** Ensure the user does not refute that he used the network
- **Access control:** Mechanisms that restrict access to resources only to authorized users, processes, or devices
- **Data confidentiality:** Ensuring that sensitive information is not disclosed to unauthorized individuals, entities, or processes
- **Protection Against Traffic Analysis (Sniffing):** Preventing attackers from analyzing network traffic to extract useful information (even if they can't read the data)

Definition 1. *(Threat)*

A threat is any potential danger that could exploit a weakness in a system and cause harm. It can be intentional, such as an attack by threat actors with malicious intent or, unintentional, like a system failure, human error, or natural disaster.

Definition 2. (*Vulnerability*)

Vulnerability is a weakness or flaw in a system that could be exploited by a threat.

Definition 3. (*Risk*)

Risk is the likelihood that a threat will exploit vulnerability and cause negative consequences.

I.3 Cyberattack

A cyberattack refers to any deliberate attempt to steal, expose, modify, disable, or destroy data, applications, or other assets by gaining unauthorized access to a network, computer system, or digital device. It can be classified under the following categories [2]:

- **Active attacks:** aim to manipulate system resources or impact their operation. The attacker's goal is to compromise integrity**,** availability**,** or both by actively interfering with the system or network. These attacks are often detectable but can cause damage.

- **Passive attacks:** aim to extract sensitive information from a system without affecting its resources. The attacker's goal is to steal information without being detected. These attacks focus on confidentiality and are harder to detect since they do not directly interfere with the system.

I.3.1 CyberAttack Stages

A typical attacker follows a structured sequence of events to infiltrate an organization's network and exfiltrate information, data, or trade secrets. This sequence often includes reconnaissance, weaponization, delivery, exploitation, installation, command and control, and actions on objectives [3]. Each step is designed to exploit specific vulnerabilities within the organization's security framework.

Having a comprehensive understanding of attackers and their methodologies is critical to building an effective defense. In the following we will analyze the details of each stage:

- **Reconnaissance:** During this stage the attackers gather information about the target organization to identify potential vulnerabilities, and try to find out which attack methods will be most effective against it, such as bribing an employee, e-mail attachments with viruses, decrypting Wi-Fi traffic, or some other phishing tactics.
- **Weaponization:** During this stage of a cyberattack, the attacker uses the information gathered in the reconnaissance phase to prepare a suitable payload for the attack. This can involve exploiting web applications or creating customized malware tailored to the target. A common tool used is a Remote Access Trojan (RAT).
- **Delivery:** During this stage of a cyberattack, attackers utilize realistic and efficient methods, such as emails, USB devices, or watering hole attacks, to transmit the selected payload. Delivery can be target-initiated, like opening a malicious PDF, or attacker-initiated, such as using SQL injection or compromising network services. This stage presents the first opportunity for defenders to mitigate attacks using tools like Intrusion Detection Systems (IDS)
- **Exploitation:** During this **s**tage of a cyberattack, the attacker's payload is triggered on the target system to compromise it and establish a foothold in the environment. For the exploit to be successful, it must match the target's operating system, software version, and patch status, while also evading antivirus and other security controls. Once executed, the payload connects to the attacker's Command and Control (C&C)

system, awaiting further instructions. This highlights the importance of keeping systems updated and implementing robust security measures to prevent exploitation.

- **Installation:** During this stage, the attacker ensures continued control over the compromised network by installing malware to enable further operations. With a foothold established in the system, attackers focus on maintaining access and escalating privileges. This privilege escalation allows them to access more secure data and restricted systems that require higher-level permissions. By gaining elevated access, attackers can expand their reach within the network and conduct more damaging activities, such as stealing sensitive information or disrupting critical operations.
- **Command & Control:** During this stage, the attacker establishes a Command and Control (C&C) channel using the installed malware, providing access to the victim's internal assets. The attacker gains full control over the compromised machine and uses the C&C channel to issue instructions, specify actions, and gather targeted information.
- **Act and Objectives:** The final stage of a cyberattack involves the attacker executing their ultimate goal, such as data exfiltration, destruction of critical infrastructure, defacing web properties, or engaging in extortion. Once the mission is accomplished, many attackers do not leave the environment entirely but instead maintain access for potential future operations. Meanwhile, the affected organization must deal with the aftermath, including addressing the attack’s repercussions, mitigating damage, and restoring normal operations. This stage often leaves a lasting impact on the organization’s reputation, finances, and operational continuity.

I.3.2 Attacks classification

Attacks can be classified based on different criteria, including their severity, the vulnerabilities they exploit, and the techniques used to execute them. Among the various classifications proposed. The attacks that affect a large number of computers in the world daily can be classified into four attacks [4], which will be detailed in the sections below:

Denial of Service (DOS)

A Denial of Service (DoS) attack is a type of cyberattack where a malicious actor attempts to render a computer, network, or service unavailable to its legitimate users by disrupting its normal operations. This is typically achieved by overwhelming the target with

an excessive volume of traffic or exploiting specific vulnerabilities to cause a crash or freeze [5].

There are several types of DoS attacks, each employing different techniques to achieve disruption. For example, a SYN Flood attack exploits the TCP handshake process, sending numerous connection requests without completing them, which ties up the target's resources. A Ping of Death attack sends oversized or malformed ICMP packets, causing the target to malfunction. Similarly, a UDP Flood attack overwhelms the target by sending large volumes of UDP packets to random ports, while an HTTP Flood attack targets web servers by sending a flood of seemingly legitimate HTTP requests, exhausting their capacity.

User to Root (U2R)

A User to Root (U2R) attack is a class of exploit where the attacker begins by gaining access to a regular user account on the target system, often through methods such as sniffing passwords or employing social engineering techniques. Once initial access is obtained, the attacker uses various techniques to exploit vulnerabilities within the system, with the ultimate goal of escalating their privileges to gain administrator or super-user access [4].

One of the most common types of U2R attacks is the buffer overflow attack, where the attacker takes advantage of improper memory handling to execute arbitrary code or gain control over the system. With administrative access, the attacker can alter configurations, install backdoors, steal sensitive data, or carry out other malicious actions, making U2R attacks particularly dangerous and impactful.

Remote to Local (R2L)

A Remote to Local (R2L) attack, also referred to as a Remote to User attack, occurs when an attacker who does not have an account on the target machine sends packets over a network to exploit vulnerabilities and gain unauthorized access. These attacks aim to escalate privileges from a remote connection to those of a local user. They are conceptually similar to User-to-Root (U2R) attacks but have a more modest ambition, focusing on gaining local user privileges rather than full administrative access.

Attackers achieve their goals through various methods, such as exploiting buffer overflows in network server software like IMAP, Sendmail, or named. or taking advantage of misconfigurations or weaknesses in the system's security policies.

Vulnerabilities in utilities such as xlock, guest, xnsnoop, phf, and Sendmail are common targets.

Probe

Probe or probe-response attacks represent a new threat to collaborative intrusion detection systems (IDS). These attacks are specifically designed so that the target detects and reports them with a recognizable "fingerprint" in the system's logs or alerts. The attacker then leverages the collaborative infrastructure to gather valuable information about the IDS's location and defensive capabilities based on the report. Probing attacks aim to identify vulnerabilities in the target system by executing carefully selected sequences of operations and observing the system's or IDS's response. In some cases, probing attacks are used to map the target's IP address space, often as a precursor to launching malware campaigns [4]. Scanning tools such as Satan, Saint, and Mscan allow even novice hackers to quickly scan and examine thousands of machines on a network, making these types of attacks increasingly accessible and dangerous.

I.4 Security Mechanisms

The widespread availability of attack tools and the abundance of information sources have significantly increased the risk of network intrusions. As a result, administrators are taking proactive measures to strengthen the security of their computer systems, ensuring better protection against potential threats and vulnerabilities. In the following we will briefly list and explain some common security tools.

I.4.1 Cryptography

Cryptography is a specialized branch of computer science and information security that employs mathematical techniques to secure communication and data storage. Its primary objective is to encode information, ensuring that only authorized parties can access and understand it. It ensures confidentiality, integrity, non-repudiation and authenticity of data. It is frequently used in various network applications such as messaging, remote connections, private networks and web servers. It can be categorized into two types:

- **Symmetric Cryptography:** Uses the same key for encryption and decryption. The sender uses the key to encrypt the plaintext and sends the cyphertext to the receiver. The receiver applies the same key to decrypt the message and recover the plaintext. It

is efficient for large amounts of data but requires secure key distribution. The common algorithms in this type of cryptography are:

- **AES (Advanced Encryption Standard):** is the most widely used symmetric encryption algorithm. It supports key sizes of 128, 192, and 256 bits, and is known for its high speed and strong security. AES is commonly used in VPNs, HTTPS, and Wi-Fi protection (WPA2/WPA3).
- **DES (Data Encryption Standard):** was one of the earliest symmetric encryption algorithms. It uses a 56-bit key, which is now considered too short to resist brute-force attacks. Due to this weakness, DES is no longer recommended for secure communication.

– **Asymmetric Cryptography:** Asymmetric encryption keeps data secure by using cryptographic algorithms to generate a pair of keys: a public key and a private key. Anyone can use the public key to encrypt data, but only those with the right private key can decrypt that data to read it. Suppose Alice want to send Bob a message. Alice encrypts some information using Bob's public key, Bob decrypt the ciphertext using his private key.

Public key cryptography algorithms that are in use today for key exchange or digital signatures include:

- **RSA (Rivest-Shamir-Adleman):** is the most established asymmetric algorithm. It uses a pair of public and private keys based on the difficulty of factoring large prime numbers. RSA is used for secure key exchange, digital signatures, and is widely deployed in SSL/TLS, SSH, and email encryption. Its security comes from the difficulty of factoring large integers that are the product of two large prime numbers. Multiplying these two numbers is easy but determining the original prime numbers from the total (or factoring) is considered infeasible due to the time it would take using even today's supercomputers

I.4.2 Firewall

A firewall is a security device that monitors and controls incoming and outgoing network traffic based on predetermined security rules. Its primary purpose is to protect an

internal network from external threats, such as malicious attacks and viruses, by acting as a barrier between the local network and the internet. There are mainly three types of Firewall:

- **Hardware Firewalls**: Physical devices installed between internal network and the internet.
- **Software Firewalls**: Programs installed on individual computers to regulate traffic at the machine level.
- **Cloud-based Firewalls**: It's a Firewall as a service in the cloud and requires no extra hardware or software installation.

Firewalls play a vital role in network security but do not provide complete protection. They are ineffective against internal threats, such as users unknowingly opening malicious files, and cannot prevent attacks that bypass security measures, highlighting the importance of a comprehensive security strategy that includes user training and other security measures.

I.4.3 Vulnerability Scanners

Vulnerability scanners play a crucial role in automating the detection of security weaknesses in computer systems. While they are often used by attackers to identify potential entry points, system administrators can leverage them to proactively detect and mitigate vulnerabilities.

However, vulnerability scanners have several limitations, primarily in three key areas: completeness, updating and accuracy. Indeed, despite the large number of vulnerabilities detected, today's scanners are unable to determine all possible weaknesses

I.4.4 Virtual Private Network (VPN)

A Virtual Private Network (VPN) enhances online privacy and security by encrypting user data and masking their IP address and location. Instead of connecting directly to the internet, users first connect to a secure VPN server, which then routes their traffic to its destination. This process helps protect sensitive information from cyber threats.

VPNs are particularly beneficial for remote workers, allowing them to securely access corporate networks even when using unsecured public Wi-Fi networks in places like coffee shops and airports. By encrypting internet traffic, VPNs prevent hackers from intercepting communications, reducing the risk of data breaches and unauthorized access.

Despite their advantages, VPNs have limitations, such as potential speed reductions and the inability to prevent all types of cyber threats. Therefore, they should be used alongside other security measures, such as multi-factor authentication (MFA) and endpoint protection, for a comprehensive cybersecurity strategy.

I.4.5 Intrusion Detection and Prevention Systems (IDPSs)

Intrusion Detection and Prevention Systems (IDPS) are essential components of modern cybersecurity strategies, offering real-time threat detection and response capabilities. Unlike traditional firewalls, which primarily control network traffic based on predefined rules, IDPS provides deeper visibility into network activity, identifying suspicious patterns and potential security breaches [6].

An **Intrusion Detection System (IDS)** is a security solution that monitors network traffic and system activities to detect suspicious behavior or potential security breaches. It alerts administrators about possible threats but does not take direct action to prevent them.

An **Intrusion Prevention System (IPS)**, on the other hand, extends the capabilities of IDS by not only detecting threats but also actively preventing them. It can block malicious traffic, terminate harmful connections, or reconfigure firewall rules to stop an attack before it causes harm.

I.5 Intrusion Detection Systems (IDS)

I.5.1 IDS definition

In information systems, intrusions can be defined as any activity that violates the system's security policy [7]. An intruder or attacker seeks to gain unauthorized access to information, cause harm, or engage in other malicious activities.

Intrusion detection refers to the process of monitoring and analyzing events occurring within a network or computer system to identify potential threats that could compromise security policies or standard security practices. As discussed earlier in Section 1.4.5, An Intrusion Detection System (IDS) is a combination of hardware and software components designed to automate this process. Figure 1.1 represents a generic architecture of intrusion detection systems. It is composed of: a sensor, an analyzer, a response module, and a repository.

A basic Intrusion Detection System (IDS) follows a clear architecture involving several key components [12]. First, the sensor monitors the network traffic or system activities, collecting data such as packets or logs. This data is then sent to the analyzer, which processes the information, searching for suspicious patterns or anomalies that may indicate an attack. If an attack is detected, the response module takes action, either by generating alerts, blocking the attack, or taking other corrective measures to mitigate the threat. Finally, the repository stores valuable data, including logs, historical alerts, and attack signatures, which can be used for further analysis, reporting, or auditing. This architecture enables the IDS to effectively monitor, detect, and respond to threats in a system or network environment.

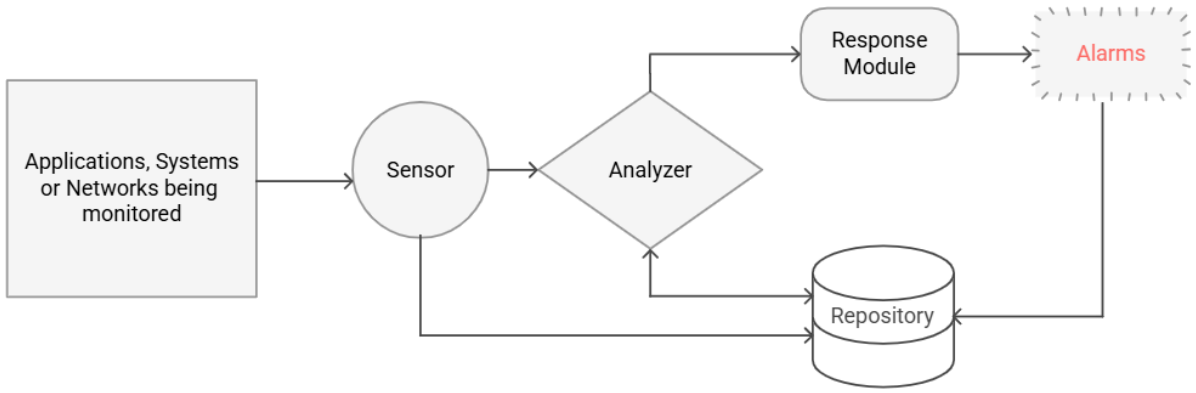

Figure 1.1 -Basic intrusion detection system architecture [12].

I.5.2 Roles of IDS

Intrusion Detection Systems (IDS) have become a critical component of an organization's IT security framework. Their main function is to identify and report security incidents. Organizations implement IDS for various security and operational reasons, including:

- Identifies unauthorized access, malware, and other suspicious activities within a network or system
- Continuously analyzes network packets and system logs for unusual patterns or anomalies
- Ensures compliance with security policies by flagging unauthorized actions or access attempts
- Uses anomaly detection to recognize deviations from normal user or system behavior

- Notifies administrators in real-time about potential threats for further investigation detect and respond to attacks, preventing intrusions before they can cause damage to the information system. However, achieving this level of proactive defense is challenging due to the increasing complexity and variety of cyber threats, as well as the emergence of new attack techniques driven by evolving technologies

I.5.3 IDS Taxonomy

Over the years, researchers have continuously advanced detection techniques and methodologies. Numerous studies have surveyed IDS frameworks or introduced taxonomies. In [5], the authors review various IDS techniques, and Figure 1.2 illustrates the proposed IDS taxonomy as outlined in [8].

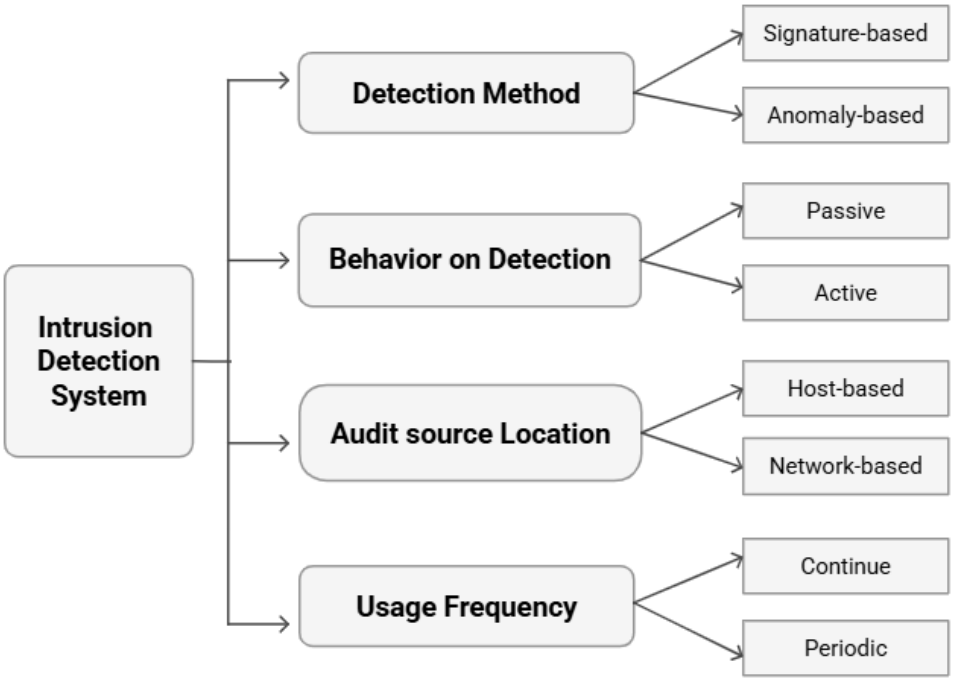

Figure 1.2 -Intrusion Detection system Taxonomy [8].

The classification of an Intrusion Detection System can be influenced by several factors, including the detection mechanisms employed (detection method), its response to detected threats (behavior on detection), the type of data being analyzed (audit source location), and the operational settings in which it functions (usage frequency):

- **Detection method:** defines the set of techniques used by intrusion detection systems in the detection process. There are two well-known approaches of intrusion detection [9]:

    - Signature-based detection: Operates by comparing system activity and audit data against a repository of known attack patterns. It continuously monitors events or sequences of events, identifying those that match predefined signatures stored in its database
    - Anomaly-based detection: Anomaly detection: is based on normal activity profile of a system (e.g., CPU usage, job execution time, system calls, etc.). Any action that significantly deviates from the normal behavior is considered intrusive

- **Behavior on detection:** describes the response of the intrusion detection system to an attack. The IDS can issue active responses that directly impact the attack source, as they can be restricted to passive responses that register the suspicious event. A list of active and passive expenses is presented in the table 1.1 [10]

Table 1.1 -List of common passive and active intrusion responses [10].

Passive	Active
Administrator notification: Generate alarm (through email, online/pager notification, ect.) Generate report (can contain information about an intrusion such as attack target, criticality, time, source IP/user account, description of suspicious packets, etc. as well as intrusion statistics for some period of time such as number of alarms from each IDS, attack targets grouped by IP, etc.) Other responses: enable additional IDS enable local/remote/ network activity logging enable intrusion analysis tools backup tampered with files Trace connection for information gathering purposes.	Host-based response actions: Deny full/selective access to file Delete tempered with file Allow to operate on fake file Restore tampered with file from backup Restrict user activity Disable user account Shutdown compromised service/host Restart suspicious process Terminate suspicious process Disable suspicious services Abort suspicious system calls Delay suspicious system calls Network-based response actions: Enable/disable additional fire wall rules restart targeted system Block suspicious incoming/outgoing network connection block ports/IP

	Trace connection to perform attacker isolation/quarantine create remote decoy

- **Audit source location:** it refers to where an IDS look for intrusive behavior. There are two categories:

  - **Host based IDS (HIDS):** monitors and processes activities within a software environment specific to a single host. It examines inbound and outbound communication traffic, analyzes system logs, controls access to system calls, and verifies file system integrity.
  - **Network based IDS (NIDS):** oversees the entire network environment by analyzing audit trails from multiple hosts and monitoring network traffic for intrusion signatures. It employs specialized sensors positioned at strategic locations within the network to conduct localized traffic analysis. Detected threats and suspicious activities are then reported to a centralized management console for further investigation and response.

- **Usage frequency (synchronization):** Synchronization refers to the time interval between the monitoring of events and their subsequent analysis.

  - **Periodic (Delayed synchronization):** Data is collected and analyzed at scheduled intervals rather than in real time. This method follows a "store and send" approach, commonly used in Host-Based Intrusion Detection Systems (HIDS) that scan system logs periodically.
  - **Continuous (Real-Time synchronization):** Data is processed as it is received, enabling instant detection and response. This approach is mainly used in Network-Based Intrusion Detection Systems (NIDS) to monitor live network traffic and take immediate action against threats.

I.5.4 Intrusion Detection Systems Issues

Intrusion Detection Systems (IDS) play a crucial role in identifying and analyzing potential security threats within networks and host systems. However, despite their effectiveness, IDS solutions come with several limitations that can impact their reliability

and performance. One of the most critical IDSs limitations are the alert representation format [11], false positive rate, and huge number of generated raw alerts, etc.

Alerts representation

One of the significant issues with Intrusion Detection Systems (IDS) is the lack of standardized alert representation, which creates significant challenges for security teams. Different IDS solutions, such as Snort and Corero, use proprietary alert formats, making it difficult to integrate and correlate alerts from multiple sources. This inconsistency in alert structures and severity levels complicates the interpretation of events and slows down incident response. Additionally, the high volume of alerts generated by IDSs, many of which are low-priority or redundant, can lead to alert fatigue, further hindering effective monitoring. Without a common format, automating responses and integrating alerts with other security tools becomes cumbersome, delaying threat mitigation and increasing operational complexity.

False Positives and False Negatives

Every Intrusion Detection System (IDS), regardless of its detection technique, is prone to generating both false positives and false negatives. A false negative occurs when an attack goes undetected, meaning no alert is generated, while a false positive happens when non-malicious behavior is incorrectly flagged as an attack. Although false negatives are critical for the vendor of the IDS, false positives pose a more significant issue from a user's perspective. A high number of false positives can obscure real threats, leading to poor decision-making in response. For example, legitimate users may be mistakenly blocked due to false positives, resulting in service disruptions and negatively impacting the user experience.

Duplicate Alerts

Duplicate alerts in Intrusion Detection Systems (IDS) occur when the same event or threat is reported multiple times, often leading to confusion and unnecessary action. This redundancy can stem from various factors, such as multiple IDS components detecting the same attack, misconfigured sensors, or overlapping detection rules. The presence of duplicate alerts exacerbates the problem of alert overload, where security teams are inundated with repetitive notifications, making it harder to prioritize and respond to actual threats effectively. In some cases, repeated alerts for the same event can lead to alert fatigue, where administrators become desensitized to the alerts, increasing the likelihood of missing critical incidents.

Information about Attacks

The incomplete or fragmented information provided by individual alerts generated by Intrusion Detection Systems (IDS). Alerts often contain partial data about ongoing attacks, which makes it challenging to fully understand the scope, nature, and progression of a threat. Without comprehensive context, security operators are unable to effectively assess the risk or take appropriate action to mitigate the attack. Multiple alerts, even if related to the same event, might not clearly indicate that they are part of a larger attack, leading to potential misinterpretation or delayed responses. This lack of contextual awareness and insufficient data from individual alerts creates the need for alert correlation, which aggregates and analyzes alerts from different sources to provide a holistic view of the attack and enable informed, timely decision-making.

I.6 Conclusion

This chapter has provided key insights into network security, detailing the stages and classifications of cyberattacks, security solutions, and the role of Intrusion Detection Systems (IDS) in mitigating threats. While IDS play a crucial role in identifying malicious activities, they are not without limitations, such as event observation challenges, human factors, and susceptibility to denial-of-service attacks. However, when integrated into comprehensive security architecture, IDS can significantly strengthen an organization's defense.

Despite their effectiveness, IDS often generate large volumes of alerts, making it difficult to distinguish real threats from false positives. This brings us to the next critical aspect of cybersecurity: alert correlation. In the following chapter, we will explore how correlating alerts from multiple security tools enhances threat detection, reduces false alarms, and provides a clearer understanding of cyberattacks in real time

.

Chapter II

Alert Correlation

II.1 Introduction

Intrusion detection tools provide the security analysts with huge amount of alerts. Managing and analyzing such tremendous number of alerts, which are often generated in different formats and contain a high false alarm rate is a challenging task for the security administrator. Alert Correlation seems to be one of the keys to the evolution of intrusion detection systems. Correlating alerts would reduce the volume of data quantities that are to be analyzed, hence improving threat detection and enabling a better view of a secured environment on an intrusion incident.

This chapter provides an overview of alert correlation, highlighting its importance in managing and analyzing security alerts effectively. It explores alert representation and its role in structuring alerts for efficient processing. Additionally, the objectives of alert correlation are outlined to clarify its significance in intrusion detection systems. A detailed definition and classification of alert correlation methods are presented, covering various approaches used in the field. Finally, the advantages and drawbacks of different correlation techniques are discussed, providing insights into their strengths and limitations.

II.2 Alerts Correlation

The term correlation has been used in many domains, such as science, biomedical, business, and others. Correlation refers to a process for establishing the relationships between two variables [13]. It is useful because it can show a predictive relationship that can be exploited in practice.

Alert correlation is a specialized form of correlation applied in cybersecurity, where it plays a key role in detecting, analyzing, and responding to threats more effectively. It is the analysis of alerts triggered by one or more IDSs in order to provide a synthetic and high-level view of interesting malicious events targeting the information system. We call an alert, a message sent by any component of an intrusion detection system when an attack or abnormal activity is detected [14].

II.3 Alert Representation

The diversity in the elements of any network presents a significant security and management challenge. Each security tool, such as firewalls, IDSs, antivirus software, VPNs,

and specialized security appliances, operates independently, often using proprietary formats and vendor-specific consoles. The same applies to IDS alerts, as various systems produce notifications in various formats for the same suspicious activity and therefore are not effective to function in tandem with one another. As a result, matching alerts from various sources is complicated, making effective threat detection and response difficult.

The challenge of interoperability among intrusion detection systems (IDSs) was identified early, in 1997 the US government's Defense Advanced Research Projects Agency (DARPA) initiated the Common Intrusion Detection Framework (CIDF) [7] research project. The developers of this project have set up a model allowing interoperability between the different components of an intrusion detection system. This effort was completed by the CISL (Common Intrusion Specification Language) which ensures representation and communication of data between these components. Despite the power of the CISL language, it has not been adopted by most manufacturers, which ended the project in 1999.

The failure of the CIDL motivated the creation of the IDWG working group in the IETF with the participation of several CIDF researchers. This group aimed to establish a standardized data format and procedures for information exchange among intrusion detection systems, response systems (such as CERT), and management consoles. Unlike the CIDF which remained a simple research project, the IDWG tried to propose standards for interoperability between IDSs. One of its key proposals was the **Intrusion Detection Message Exchange Format (IDMEF)** model.

II.3.1 IDMEF Model

The IDMEF standard, illustrated in Figure 2.1, is a data model created by the IETF to enable different intrusion tools to report potentially suspicious events in a unified format. IDMEF ensures syntactic and semantic interoperability of IDSs. Syntactic interoperability is tasked with processing data generated by different systems in a homogeneous manner. Semantic interoperability is tasked with guaranteeing that identical information, regardless of its source, conveys the same meaning. IDMEF uses DTD XML format in order to represent alerts and uses an object-oriented representation.

The top-level class of the IDMEF data model is IDMEF message. It is composed of two sub-classes: Alert and Heartbeat. The Alert class contains information about events detected by an analyzer. It primarily consists of nine main classes. Below, we outline the most significant ones used within the Alert class:

Analyzer: This component processes data collected by sensors to identify suspicious activity. Usually, the analyzer and the sensor are included in the same component in existing IDSs. In the alert class, the analyzer is identification information for the analyzer that originated the alert.

DetectTime: The time when the attack is detected.

Source: it contains information about the source of the attack leading up to the alert. It is composed of four sub-classes: Node, User, Process and Services.

Target: It contains information about the possible targets that are exploited by the detected attack. It is composed of five sub-classes: Node, User, Process, Service, and File.

Classification: This class provides the name of the alert or other information allowing the manager to determine what it is.

Assessment: informs the security administrator about the analyzer's assessment of the event.

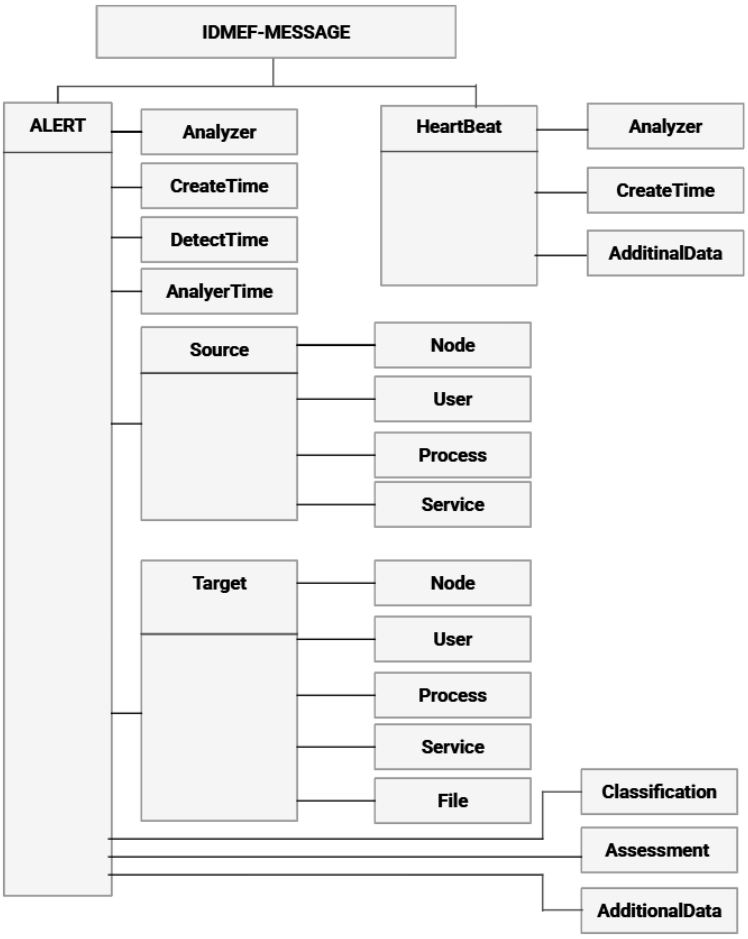

Figure 2.1 -IDMEF Model.

II.4 Alert correlation objectives

Here we identify three main objectives of alert correlation: reducing the volume of information to be processed by operators and analysts, increasing the quality of the diagnosis provided, and monitoring attacks over time.

II.4.1 Reduction in the Volume of Information

An intrusion detection probe can generate a high number of alerts in a short period, while the processing capacity of an operator is limited. So the first objective of correlation is to reduce the volume of alerts to facilitate the analysis task. This reduction is crucial, as certain alerts frequently occur due to the specific configuration and operation of the monitored information system. These alerts can be disabled in the configuration of the intrusion detection system, but it risks ignoring real attacks [16].

The objective of reducing the volume of information is broken down into four sub-objectives:

- **Redundancy elimination:** for the same event in the information source, several redundant alerts can be generated by one or more intrusion detection probes. In this case, the first function of an alert correlation system is to determine whether two alerts have been generated following the observation of the same event. Thus, the elimination of redundancy reduces the number of alerts to be processed.
- **Alert merging:** several alerts can be merged into a single one that contains all the information carried by these alerts. This method is especially effective for rapid sequences of events that IDS may separately alert and notify of, where each individual alert may contain a distinctive single precious event. Rather than dealing with each alert separately, merging enables alert notifications to be condensed into a single alert that contains important information.
- **Alert aggregation:** serves the purpose of lowering the alert volume. This is done by picking a portion of alerts that share a distinct characteristic and categorizing them depending on it. With aggregation, rather than gathering information about every solitary alert, it is more feasible to grasp the sequence of actions that make up the attack by focusing on commonalities such as the attacker's IP address, the attacked system, or even the type of attack itself. Such organization of alerts enables security.

officers to glean pertinent information concerning attack patterns instead of losing focus on a myriad amount of details

- **Attack scenarios Recognition:** also known as context correlation is an advanced approach that goes beyond individual alert analysis to identify multi-step attack patterns. Many sophisticated cyberattacks are not single actions but rather a sequence of intentional actions that follow a structured attack lifecycle. Rather than responding to all alerts in isolation, this function aims to identify patterns for known and unknown threats by linking together multiple distinct alerts over a certain period to determine whether an attack has taken place.

  This can be achieved in varying degrees through a plethora of diverse correlation techniques, developed over the past years. Some of the key methods include rule based correlation, statistical correlation, and so on. We will discuss these methods in section 2.4.

II.4.2 Improving the quality of diagnosis

Enhancing intrusion detection system (IDS) alert quality entails making the notifications more informative and helpful. IDS notifications frequently do not include the details necessary to completely comprehend the attack, such as whether the attack was successful or whether the system really was susceptible. For example, it's critical to determine if the attacker's actions truly hacked the system or if they were only doing fingerprinting to test the system for vulnerabilities. Knowing if the vulnerability has been repaired or is still there is also essential since some attacks may target problems that have previously been patched or may be the result of an error, such as hitting a system that isn't even susceptible. By using techniques like aggregation (grouping similar alerts) and synthesis (combining different alerts into a more comprehensive one), the system can provide a clearer picture, helping analysts better assess the severity and success of the attack.

II.4.3 Attack Tracking

An attack is rarely just a one-off action; it's usually made up of several smaller actions or sub-attacks that happen over time. Typically, an attacker will start by gathering information about the target system learning about its components, services, and potential weaknesses. With this information, they'll choose which targets to attack, modify their tactics based on the results, and move forward with their plan. Each of these sub-attacks tends to generate multiple

alerts in an intrusion detection system (IDS). The main goal of alert correlation is to connect these related alerts, helping to identify the different parts of the attack. This process needs to be both fast, so analysts can quickly group related alerts, and robust, meaning it should be able to withstand efforts by the attacker to hide their actions or changes that come from the environment. By correlating these alerts, security teams can track the attacker's moves, gather important details about the techniques they used, and understand what information the attack has gathered from the target system.

II.5 Alert Correlation Taxonomy

In the following, we present an overview of the different correlation methods proposed in the last decade, which can be divided into three categories based on their characteristics: 1) Similarity-based, 2) Knowledge-based and 3) Model-based [17]:

II.5.1 Similarity-based alert correlation

The similarity correlation methods aim at clustering alerts that have the closest similarity values between alert attributes. Most considered attributes are source and destination, IP address and ports, as well as timestamp. If an alert or meta-alert has needed similarity, each one of them is merged with the alert or meta-alert, and otherwise a new meta-alert is created.

Similarity-based correlation between attributes is a basic technique. However, which attributes should be compared and how to evaluate this comparison are the difficult tasks in this method. The probabilistic function as proposed in [18] is computed to evaluate similarities between alerts. This approach assigns probability weights to different attributes and calculates an overall similarity score ranging between 0 (no similarity) and 1 (exact match). Figure 2.2 shows an example of probabilistic similarity calculation for the attack

	INVALID	PRIVILEGE_VIOLATION	USER_SUBVERSION	DENIAL_OF_SERVICE	PROBE	ACCESS_VIOLATION	INTEGRITY VIOLATION	SYSTEM_ENV_CORRUPTION	USER_ENV_CORRUPTION	ASSET_DISTRESS	SUSPICIOUS_USAGE	CONNECTION_VIOLATION	BINARY_SUBVERSION	ACTION_LOGGED
INVALID	1	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.6
PRIVILEGE_VIOLATION	0.3	1	0.6	0.3	0.6	0.6	0.6	0.6	0.4	0.3	0.4	0.1	0.5	0.6
USER_SUBVERSION	0.3	0.6	1	0.3	0.6	0.5	0.5	0.4	0.6	0.3	0.4	0.1	0.5	0.6
DENIAL_OF_SERVICE	0.3	0.3	0.3	1	0.6	0.3	0.3	0.4	0.3	0.5	0.4	0.1	0.5	0.6
PROBE	0.3	0.2	0.2	0.3	1	0.7	0.3	0.3	0.3	0.3	0.4	0.8	0.3	0.6
ACCESS_VIOLATION	0.3	0.6	0.3	0.5	0.6	1	0.6	0.6	0.3	0.3	0.4	0.1	0.5	0.6
INTEGRITY VIOLATION	0.3	0.5	0.3	0.5	0.6	0.8	1	0.6	0.5	0.3	0.4	0.1	0.5	0.6
SYSTEM_ENV_CORRUPTION	0.3	0.5	0.3	0.5	0.6	0.6	0.6	1	0.6	0.3	0.4	0.1	0.5	0.6
USER_ENV_CORRUPTION	0.3	0.5	0.5	0.3	0.6	0.6	0.6	0.6	1	0.3	0.4	0.1	0.5	0.6
ASSET_DISTRESS	0.3	0.3	0.3	0.6	0.3	0.3	0.3	0.3	0.3	1	0.4	0.4	0.3	0.6
SUSPICIOUS_USAGE	0.3	0.3	0.5	0.3	0.5	0.6	0.5	0.6	0.5	0.3	1	0.1	0.3	0.6
CONNECTION_VIOLATION	0.3	0.1	0.1	0.3	0.8	0.3	0.3	0.3	0.3	0.5	0.4	1	0.3	0.6
BINARY_SUBVERSION	0.3	0.3	0.3	0.3	0.3	0.6	0.6	0.6	0.5	0.3	0.4	0.1	1	0.6
ACTION_LOGGED	0.3	0.3	0.3	0.3	0.6	0.5	0.3	0.3	0.3	0.3	0.4	0.3	0.3	1

Figure 2.2 –Similarity matrix between certain classes of attacks [18].

II.5.1.1 Rule-based Alert Correlation

Known as prerequisites and consequences, this mechanism can be seen as just another way of describing attack scenarios. When stating pre/post-conditions for distinct attacks, there is no need to know complete attack scenarios in advance and so do not have to insert huge amounts of correlation rules manually. It is enough to define the necessary conditions for a known attack and its potential outcomes.

Two alerts may be considered correlated when the postcondition of one action matches the precondition of another, thereby inducing some enactment between them. This relationship enables the dynamic linking of alerts and representation of multi-step attack scenarios. In contrast**,** rule-based alert correlation methods leverage the JIGSAW attack description language [22] to model attack conditions and their sequential steps. However, these approaches construct attack sequences only when all specified conditions are satisfied, making them vulnerable to incomplete attack reconstructions if certain alerts are missed. To address this limitation, the MIRADOR correlation method introduces a more flexible

approach that does not require full satisfaction of all attack conditions, allowing for more robust and adaptive alert correlation even when some attack steps are missing [23].

II.5.1.2 Scenario-based Alert Correlation

A single alert often represents a fundamental step in an attack scenario. The main purpose of this set of algorithms is to detect multi-step attacks by correlating individual alerts based on predefined sequences. However, their effectiveness relies on the existence of well-defined attack scenarios. The difficulty in this approach lies in the formulation of these scenarios, as it requires expert knowledge to accurately model attack sequences and continuously update them to adapt to evolving threats.

For the purpose of specifying attack scenarios, several modeling languages have been suggested. One of them is STATL which is based on the State Specification and Transition (STAT) [19]. In STATL, states are employed to specify security properties of a system whereas transitions are employed to model actions which constitute an attack.

As part of the Mirador project, two modeling languages, LAMBDA and AdeLe, were developed to enhance attack scenario description and intrusion detection. LAMBDA (Language to Model a Database for Detection of Attacks) is declarative language employed for specifying attacks as logical rules [20]. It aims at specifying attack circumstances more generally independently of the used attack mechanisms or attacked systems and thus constitutes a generalized intrusion detection approach. Similarly, AdeLe [21] was introduced as an attack description language with a purpose closely aligned with LAMBDA. It provides a formal attack specification to facilitate the configuration of intrusion detection systems (IDS). These languages play a crucial role in formalizing attack representation, improving the efficiency of scenario-based alert correlation, and enhancing the flexibility of IDS solutions.

II.5.2 Model-based Alert Correlation

Model-based alert correlation focuses on analyzing security alerts within their relevant context, enabling more accurate correlation and reducing false positives. One of the earliest formal approaches in this category was M2D2 [24], which introduced a structured method for representing sensor capabilities to determine whether an alert was a false positive or a legitimate security threat. This was further enhanced by the M4D4 data model [25], which incorporates contextual and topological information to improve alert correlation.

More precisely, the objective of the M4D4 model FIGURE 2.3, is to provide a formal representation of the different sources of information involved in intrusion detection such as: the monitored system (topology and software), vulnerabilities (CVE, Bugtraq, etc.), security tools (IDS, scanners, etc.), and observed events (attacks and vulnerabilities). This formalization improves the alert correlation process by taking into account as much information as possible.

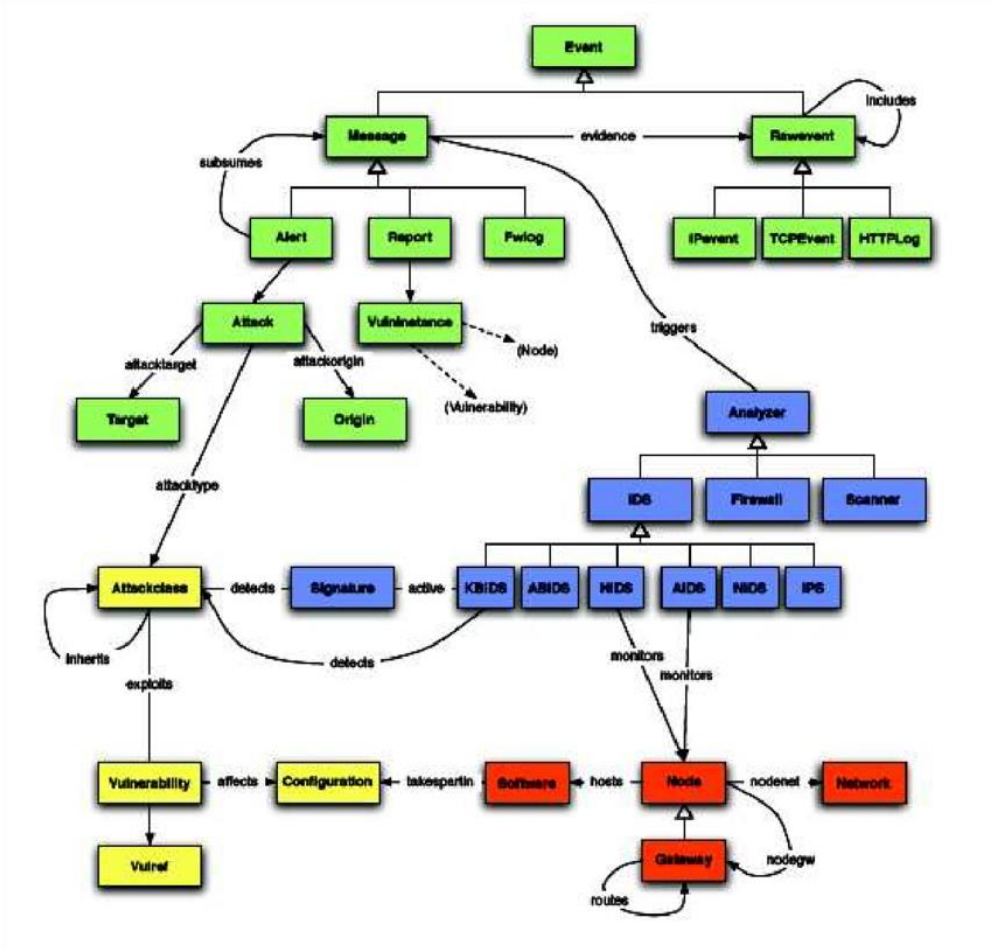

Figure 2.3 – M4D4 Model [25].

II.6 Comparison

Each alert correlation method has distinct advantages and limitations. Table 2.1 summarizes the advantages and drawbacks of each method [12]. Alert correlation methods are listed in the rows of the table, with advantages and drawbacks presented in the second and third columns, respectively. Examples of correlation techniques developed in the literature are provided in the last column

Table 2.1 -Advantages and drawbacks of alert correlation methods [12].

Alert correlation method	Advantages	Drawbacks	Examples
Similarity-based alert correlation	-easy to implement -no need for external knowledge	-unable to discover causality link between alert	Probabilistic Alert Correlation [18]
Rule-based alert correlation	-alerts are correlated based on their pre-defined causality link -dynamic attack scenario construction	-complex implementation and maintaining of inference rule -time-consuming to design inference rules - need for expert knowledge	Managing alerts in a multi-intrusion detection environment [23]
Scenario-based alert correlation	-alerts are correlated based on their pre-defined causality	-complex maintaining of defined scenarios -need for a prior knowledge on attack description	LAMBDA[20]
Model-based alert correlation	-consideration of context information and external knowledge	-complex to implement and maintain	M2D2 [24] and M4D4 [25]

II.7 Conclusion

Alert correlation is an important process in intrusion detection systems (IDSs), designed for the analysis and correlation of security alerts to most effectively identify possible attacks; it may arise out of the great diversity of sources of such alerts. The IDMEF standard was built within the framework of IETF to let the different intrusion tools report suspicious events in a unified format. Current methods include similarity-based techniques, grouping alerts with common features; rule-based, using preset if-then rules; scenario-based, linking alerts into sequences following known attack patterns; and lastly, model-based techniques, which use formal models and contextual knowledge to dynamically correlate alerts. Each of these methods offers its own strengths and limitations in addressing complex security challenges.

Chapter III

Machine Learning and Deep Learning

III.1 Introduction

Machine Learning (ML) and Deep Learning (DL) have emerged in the last few years as compelling techniques for data analyses and intelligent decision making in a plethora of applications ranging from healthcare to cybersecurity. In particular, these techniques provide the foundation to create adaptive systems that can learn from data, detect advanced patterns, and either predict or act, with minimal human intervention.

Machine Learning's application in cybersecurity, and particularly as a technique for alert correlation, has received significant interest. Alert correlation is the analysis of alerts triggered by one or more IDSs in order to provide a synthetic and high-level view of interesting malicious events targeting the information system. Because of the volume of alerts and their complexity, traditional rule-based alert correlation methods are often inadequate. Instead, machine learning provides a more intelligent, data-driven approach to automate this process by learning patterns of malicious behavior, reducing false positives, and enhancing the accuracy and efficiency of threat detection.

This chapter introduces the basic concepts of machine learning and deep learning, then provides a discussion of how machine learning techniques are applied to the alert correlation problem in a cybersecurity system.

III.2 Artificial Intelligence

Artificial intelligence (AI) is currently one of the fastest-growing fields of technology and already produces substantial revenue globally. AI is integrated in various domains, from healthcare and finance to manufacturing, education, agriculture, and cybersecurity. It's transforming how tasks are performed, decisions are made, and services are delivered; it enables automation and provides insights through data-driven analysis. However, there has been considerable confusion regarding a precise definition of AI [26]. A more recent and widely accepted definition describes AI as the capacity to “imitate intelligent human behavior” (Koket al., 2009).

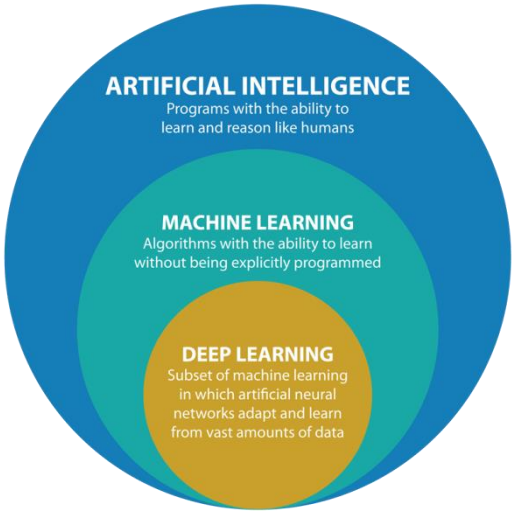

Figure 3.1 –The relationship between Artificial Intelligence, Machine Learning, and Deep Learning.

III.3 Machine learning

Machine Learning (ML) is a subfield of Artificial Intelligence (AI) (Figure 3.1) that focuses on enabling computers and systems to mimic human learning processes and improve performance over time through experience and exposure to more data. Rather than being explicitly programmed to carry out specific tasks, ML algorithms are trained on datasets to recognize patterns and make decisions or predictions. In other words, machine learning allows systems to automatically improve by learning from past data, making it a core component of many modern AI applications.

III.4 Types of machine learning problems

Machine learning is a fairly broad field, in the following we list the largest classes of problems it is interested in [27]:

III.4.1 Supervised learning

Supervised learning attempts to learn a model for predicting an output based on a known input (see Figure 3.2), consisting of a set of examples in which the correct output (label) is already known. Labels serve as a "teacher," who teaches and monitors the learning. The algorithm takes a training set made of input-output pairs, where the input is given as a feature vector and the output as a label or a value [27]. The objective is to find a mapping function that can generalize from the training data to accurately predict labels for new, unseen inputs.

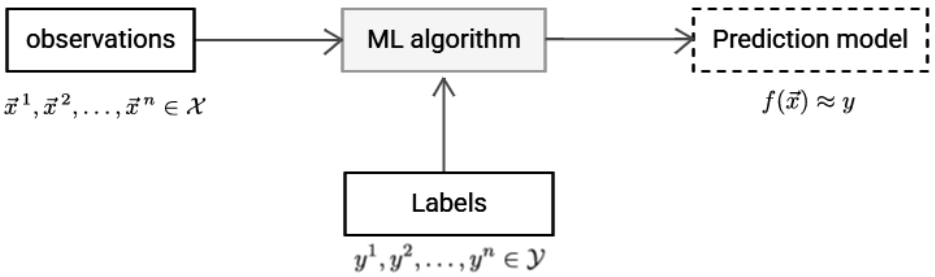

Figure 3.2 –Supervised learning.

III.4.1.1 Supervised learning tasks

Supervised learning tasks can be broadly divided into classification and regression problems:

- **Classification:** a supervised learning problem where the goal is to predict the class of new data points based on its features. The algorithm trains on labeled examples and the output is a function mapping input data to discrete output [27]. Common classification algorithms are linear classifiers, support vector machines (SVM), decision trees, k-nearest neighbors, and random forest.
- **Regression:** a supervised learning problem where the computer receives input data and predicts a continuous numerical value. The algorithm learns a function that maps an input vector of features to a real-valued output [27]. Linear regression, logistic regression, and polynomial regression are three examples of regression algorithms.

III.4.2 Unsupervised learning

These are called unsupervised learning because unlike supervised learning above, there are no correct answers and there is no teacher, the data is not labeled, meaning there are no predefined output values. The goal is to model the underlying structure or distribution of the data in order to better understand it. This approach is often used to discover hidden patterns, groupings, or features within the data without explicit instruction [28].

III.4.2.1 Unsupervised learning tasks

Unsupervised learning models are utilized for three main tasks: clustering, association, and dimensionality reduction.

- **Clustering:** Clustering is a basic unsupervised learning method in machine learning, which partitions the data into groups (also called clusters) where the

data in one group is more similar (in some sense) to each other than to those in other clusters. In contrast to supervised learning, clustering does not require labeled training data; rather, it discovers the intrinsic structure of the data based on the similarities of features [29]. There are various clustering algorithms, each with different assumptions and mechanisms, the common ones are k-means, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), and Hierarchical clustering.

- **Dimensionality reduction**: Dimension reduction is essential in machine learning and data analysis to reduce input dimension or features in a data set while retaining its relevance. Such a procedure becomes particularly relevant in high-dimensional datasets [30], which include many possibly redundant, irrelevant, or noisy features. Dimensionality reduction can speed up the computation of nearest neighbors, is useful for an improved visualization, and helps to prevent overfitting. The common algorithms of dimension reduction are Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Non-negative Matrix Factorization (NMF).

- **Association rule learning:** Association rule learning (or association rule mining) is a widely used technique in unsupervised data analysis for discovering patterns and relationships in large datasets. It is a rule-based mining process that reveals useful relationships between diverse attributes. This method of analysis is frequently used in market basket analysis, where different products can be observed to be related through the purchasing behavior of the consumer [31]. Some common association rule learning algorithms are Apriori algorithm, FP-Growth algorithm, Eclat algorithm.

III.4.3 Semi-supervised learning

Semi-supervised machine learning is a combination of supervised and unsupervised machine learning methods, it involves learning from a dataset where only a portion of the data is labeled. The main advantage of this approach is that it reduces the need to label the entire training set, which is particularly useful when data collection is easy but labeling requires significant human effort [27].

For example in 1,000,000 medical X-ray images only 5,000 of them are labeled by radiologists. Semi-supervised learning can be used to extract useful patterns from the

unlabeled 995,000 images and improve the model's ability to detect diseases without needing to label all of them manually.

III.4.4 Reinforcement learning

Reinforcement learning is a reward/punishment-based learning technique where an agent interacts with its environment by taking actions and receiving feedback in the form of rewards. These rewards can be positive if the action is beneficial, or negative if it is not. In many cases, rewards are delayed, meaning they are received only after a sequence of actions such as in games like Go or chess, where the impact of a move may not be immediately apparent. The goal of reinforcement learning is to develop a policy, a strategy that guides the agent to consistently achieve the highest possible cumulative reward. This learning paradigm is widely used in areas such as game playing and robotics [32]**.**

III.5 Common Applications of Machine Learning

The field of machine learning is developing quickly and is changing many facets of our everyday lives. Even though we might not be aware of it, we frequently engage with machine learning systems whether it is when navigating with Google Maps using voice assistants like Siri and Alexa or depending on tailored suggestions on Netflix or Amazon. Because of its adaptability, machine learning can be used in many different domains and subdomains. Some of the uses are listed below:

- **Healthcare:** Machine learning and artificial intelligence (AI) are changing healthcare in many important ways. For example, machine learning tools can analyze patient data, such as medical histories, lab results, and radiological imaging, in order to provide reliable evidence about patterns of variation that can suggest early onset of illness, enabling faster and more accurate diagnoses. It enables researchers to extract valuable biological knowledge from large datasets that result in novel therapeutic discoveries. Additionally, by analyzing individual medical profiles, machine learning supports the development of personalized treatment plans, resulting in more effective and targeted healthcare interventions.

- **Marketing and recommendation systems:** Modern marketing along with recommendation systems relies heavily on machine learning through its power to deliver individualized interactions and automated forecasting processes. Modern marketing practices use ML to achieve customer segmentation while also predicting customer behavior, and estimating their lifetime value and setting prices and

conducting sentiment analysis which leads to better-targeted advertising that maximizes business results.

- **Autonomous Systems and Robotics**: Helps cars understand their surroundings and make driving decisions. Through a combination of computer vision, sensor data processing, and deep learning models. Additionally, machine learning algorithms help the car make decisions such as when to accelerate, brake, change lanes, or navigate intersections safely.

- **Natural Language Processing (NLP**): is a subfield of computer science and artificial intelligence (AI) that uses machine learning to enable computers and digital devices to recognize, understand and generate text and speech by combining computational linguistics, the rule-based modeling of human language together with statistical modeling.

- **Fraud detection**: Banks and credit card companies use machine learning to detect fraudulent transactions. By analyzing patterns of normal and abnormal behavior, they can flag suspicious activity in real-time.

III.6 Limitations of Machine Learning

Although machine learning has advanced considerably, numerous obstacles still need to be overcome to enhance its effectiveness and dependability:

- **Data quality and Quantity**: The predictive accuracy of machine learning can be seriously affected by poor quality data such as missing values, outliers, or training data that is biased. A good practice for data cleaning, preprocessing and methods of dealing with missing data is essential in obtaining valid results.

- **Interpretability and Explainability**: Interpretability is the understanding of how a model makes a decision, while explainability is clearly communicating why a model made a decision. Models with high complexity sometimes have little transparency, which could cause issues with accuracy in sensitive domains such as health care [33].

- **Bias and Fairness:** Training data can have systematic errors or be extremely underrepresented by one group or another, in which case machine learning systems can learn that poor quality and result in a discriminatory model affecting certain

groups in a substantive way. This raises ethical issues related to the existence of bias in machine learning [34].

III.7 Deep Learning

In recent decades, the area of machine learning (ML) has established many improvements in advanced learning algorithms and efficient preprocessing methods. One of the most notable developments is the evolution of artificial neural networks (ANNs) into increasingly deep architectures with enhanced learning capabilities commonly referred to as deep learning (DL).

III.7.1 Artificial Neural Network

The development of artificial neural networks (ANNs) dates back to the 1950s, when psychologists like Frank Rosenblatt aimed to simulate how the human brain processes information. Initially designed to mimic biological neural networks in mammals, their modern value lies not in biological realism but in their powerful ability to model complex, non-linear relationships. ANNs are highly adaptable due to their modular architecture, making them suitable for a wide range of machine learning tasks, including supervised, unsupervised, and reinforcement learning. These networks consist of interconnected artificial neurons, mathematical constructs that emulate the signal-transmitting behavior of biological neurons. Like synaptic connections in the brain, each connection carries a signal scaled by a weight that is adjusted during training. Neurons activate based on an input threshold defined by an activation function. A typical neural network consists of an input layer (e.g., a product image), one or more hidden layers (which learn the mapping from input to output), and an output layer (e.g., classifying the product) [35]. The first such model, Rosenblatt's 1957 perceptron [36], was a simple, single-layer network that laid the groundwork for what would evolve into deep learning.

III.7.1.1 Perceptron

The perceptron is the simplest artificial neural network architecture Figure 3.3. It performs binary classification that maps features to an output decision, classifying data into one of two categories.

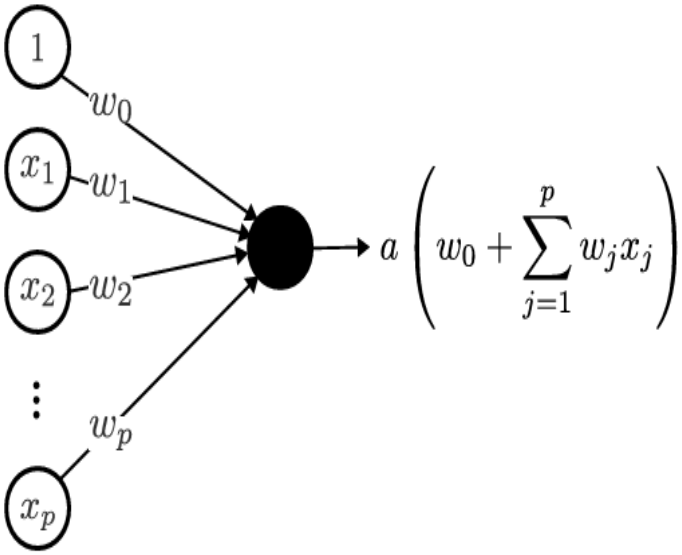

Figure 3.3 -Perceptron architecture.

The perceptron includes an input layer with p neurons, each representing an input variable $\vec{x} = (x_1, x_2, \dots, x_n)$. These neurons forward their input and a bias unit (which always outputs 1) to the next layer. The perceptron has one additional layer, just a single output neuron that connects to all inputs. This neuron computes a weighted sum $o(\vec{x})$ (equation 3.1) of the input values and applies an activation function *a* to produce the final output (equation 3.2), effectively forming the perceptron's decision rules [27, 35].

$$o(\vec{x}) = W_0 + \sum_{j=1}^{p} w_j x_j \tag{3.1}$$

$$f(\vec{x}) = a\big(o(\vec{x})\big) = a(W_0 + \sum_{j=1}^{p} w_j x_j) \tag{3.2}$$

III.7.1.2 Multilayer Perceptron

A multilayer perceptron is a neural network built by inserting intermediate layers between the input and output layers of a perceptron, referred to as hidden layers Figure 3.4 .Unlike the single-layer perceptron, which can only solve linearly separable problems, MLPs use non-linear activation functions [35]. It is a class of feed forward artificial neural networks, which each neuron in an intermediate layer or output layer receives as input the outputs of the neurons in the previous layer.

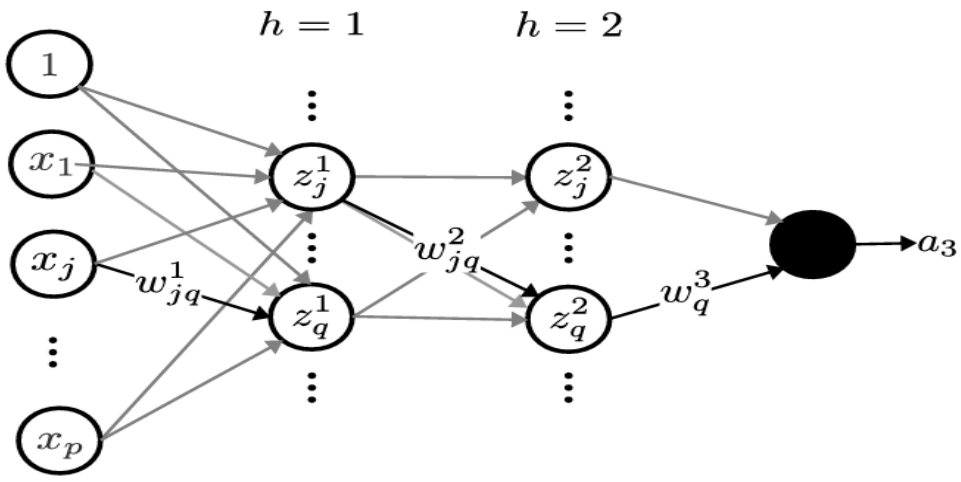

Figure 3.4 -Multi-Layer Perceptron Architecture.

III.7.2 Functioning of an artificial neural network

The process begins when an input vector (say, features of a data point) is fed into the input layer. This input vector $\vec{x}$ contains numerical values (like pixel intensities or product features) and is passed to the next hidden layers as shown in Figure 3.4.

Each neuron in the next layer (the first hidden layer *h=1*) receives all inputs from the previous layer. For each neuron it multiplies each input by a corresponding weight w_{jq}^1, then it adds a bias term w_0^1, this produces a weighted sum, denoted z_q^1 (equation 3.3) [37].

$$z_q^1 = w_0^1 + \sum_{j=1}^{p}(w_{jq}^1 x_j) \tag{3.3}$$

Note that w_{jq}^1 is the weight of the connection from neuron ***j*** of layer $h-1$ to neuron ***q*** of layer ***1***, and z_q^1 is the weighted sum of the **q**-th neuron of the first layer.

Activation function

This weighted sum z_q^1 is passed through an activation function *f* to introduce non-linearity**,** and we note $a_q^1(equation 3.4)$.

$$a_q^1 = f(z_q^1)$$

$$a_q^1 = f(w_0^1 + \sum_{j=1}^{p}(w_{jq}^1 x_j)) \tag{3.4}$$

Activation functions allow the network to learn complex patterns in data by enabling it to approximate non-linear functions. Commonly used activation functions include:

- **ReLU:** stands for Rectified Linear Unit is a non-linear activation function which is widely used in neural network, due to both simplicity of implementation and good performance on a variety of predictive tasks. Given an element x, the function is defined as the maximum of that element and 0 [38], It can be defined mathematically as:

$$\boldsymbol{ReLU}(\boldsymbol{x}) = max(0, x) \tag{3.5}$$

- **Sigmoid:** It is the most widely used activation function, it maps inputs whose values lie in the domain R to outputs that lie in the interval]0, 1[[38]. It can be defined as:

$$\boldsymbol{Sigmoid}(\boldsymbol{x}) = \frac{1}{1+e^{-x}} \tag{3.6}$$

- **Tanh:** stands for Hyperbolic Tangent, like the sigmoid function, the tanh function also squashes its inputs, transforming them into elements of the interval]-1,1[[39]. It's defined as:

$$\boldsymbol{tanh}(\boldsymbol{x}) = \frac{1-e^{-2x}}{1+e^{-2x}} \tag{3.7}$$

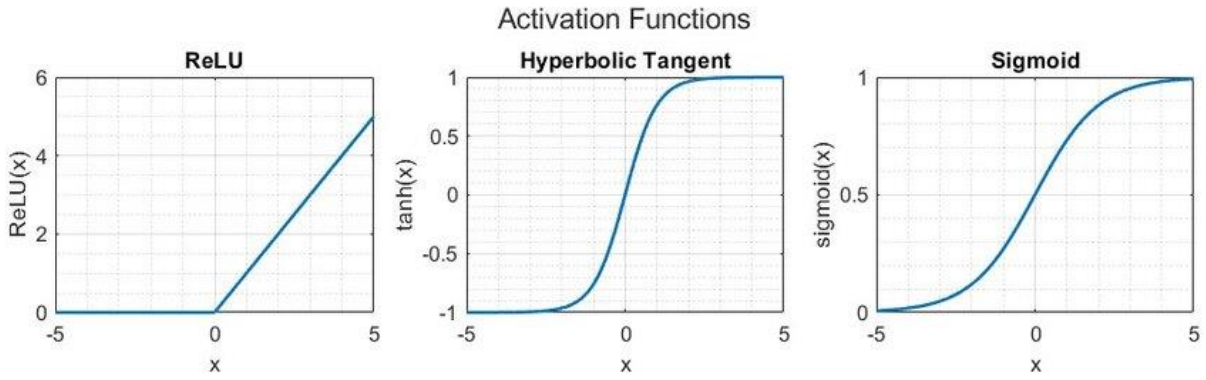

Figure 3.5 -Curves of the sigmoid, Tanh, and ReLU activation functions.

FeedForward Through Hidden Layer

The outputs of the first hidden layer are passed forward to the next layer. Each hidden layer performs the same sequence of operations: it takes the previous layer's output, applies a weighted sum with a bias (equation 3.8), and passes the result through an activation function (equation 3.9). This continues through all hidden layers until the output layer is reached [37].

$$z_q^h = w_0^h + \sum_{j=1}^{p}(w_{jq}^h a_j^{h-1}) \tag{3.8}$$

$$a_q^h = f(z_q^h) \tag{3.9}$$

Output Layer

The output layer is the final layer in the neural network which produces the final prediction or result $\hat{y}$. The number of neurons in this layer depends on the nature of the problem. For example if it is a binary classification, the output might be a single value between 0 and 1 using a sigmoid activation function (equation 3.6). For multi-class classification, the final layer of a Multilayer Perceptron contains one neuron for each class. Each neuron in this layer produces a score that is transformed into a probability using the Softmax activation function (equation 3.10) [35]. The Softmax function ensures that the outputs are positive and that the sum of all output probabilities is 1, effectively forming a probability distribution over the classes. The predicted class is typically the one with the highest probability.

$$\hat{y}_j = \frac{e^{a_j}}{\sum_j e^{a_j}} \tag{3.10}$$

Loss function

A loss function is a mathematical tool used to evaluate the performance of a machine learning model .It is used during training to optimize the model's parameters (weights and biases). It measures the difference between the predicted and expected outputs of the model, and the goal of training is to minimize this difference [35, 40]. In the following some common loss functions for regression and classification:

- **loss function for regression**:
  - **Mean Square Error (MSE)**: It measures the average of the squares of the differences between predicted values and actual values [40].

$$MSE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.11}$$

  - **Mean Absolute Error (MAE)**: It measures the average of the absolute differences between the predicted values and the true values [40].

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{3.12}$$

- **loss function for classification**:
    - **Binary cross-entropy Loss**: In binary classification, we use the binary cross-entropy loss, also known as log loss. It measures the dissimilarity between the predicted probability of a class and the true class label [27, 40].

$$BinaryCrossEntropy(y, \hat{y}) = -ylog(\hat{y}) - (1-y)\log(1-\hat{y}) \quad (3.13)$$

    - **Categorical Cross-Entropy Loss**: In multi class classification, we use the categorical cross-entropy loss, also known as multi class log loss. It measures the dissimilarity between the predicted probability distribution and the true distribution [40].

$$CategoricalCrossEntropy(y, \hat{y}) = \frac{-1}{n}\sum_{i=1}^{n}\sum_{j=1}^{C} y_{i,j}\log(y_{i,j}) \quad (3.14)$$

Backpropagation and Optimization

Backpropagation optimizes the network weights and biases by computing the gradient of the loss function with respect to the network parameters using the chain rule, starting from the output layer and working backward to the input layer [27,41](equation 3.15).

$$\frac{\partial L(y,\hat{y})}{\partial w_{jq}^{h}} = \frac{\partial L(y,\hat{y})}{\partial a_{q}^{h}} \cdot \frac{\partial a_{q}^{h}}{\partial z_{q}^{h}} \cdot \frac{\partial z_{q}^{h}}{\partial w_{jq}^{h}} \quad (3.15)$$

Once the gradients are computed, an optimization algorithm updates the parameters to minimize the loss function. Gradient descent is a common method for adjusting network parameters (equation 3.16) where the learning rate determines the step size of parameter updates [41].However, several variants exist that can enhance performance, Batch Gradient Descent computes gradients using the entire training dataset, Stochastic Gradient Descent (SGD) updates parameters using a single sample at a time, and Mini-Batch Gradient Descent finds a balance by updating using small batches of data. More advanced optimizers like Adam, RMSprop, and Adagrad enhance training by adapting learning rates or incorporating momentum, often leading to faster convergence and better performance on complex problems.

$$W_{jq}^{h} = w_{jq}^{h} - \eta \frac{\partial L(y,\hat{y})}{\partial w_{jq}^{h}} \quad (3.16)$$

III.8 Attention Mechanism

Attention has become one of the strongest concepts in deep learning. Inspired by the human ability to selectively focus on relevant information when processing large volumes of data. Attention mechanisms have been integrated into a wide range of neural network architectures [42].

The significance of attention mechanisms was greatly amplified with the introduction of the Transformer model by Vaswani et al. [43]. Unlike the older recurrent or convolution-based architectures, the Transformer completely discards recurrence and relies on self-attention to efficiently capture all input and output token dependencies. It is this architectural shift that has enabled radical breakthroughs in the area of natural language processing (NLP) and made it possible for models to better comprehend relationships among sequences of context. As a result, Transformers have now become the foundation of numerous state-of-the-art models in a broad variety of tasks, not only NLP, but computer vision, time series prediction, and even simple regression tasks, firmly cementing themselves as a pillar of modern deep learning.

III.8.1 Transformer

The Transformer is the first transduction model that relies entirely on self-attention mechanisms to compute representations of both its input and output and is not in need of sequence-aligned recurrent (RNNs) or convolutional computations. The Transformer is a sequence-to-sequence model (Figure 3.6), with the decoder and encoder both built as stacks of the identical blocks. The encoder block has a multi-head self-attention block and position-wise feed-forward network (FFN) enriched with residual connections and layer normalization for the stable training of deeper models. The decoder block takes this architecture forward with the addition of a cross-attention block (between self-attention and FFN) for attending to the output of the encoder while its own self-attention gets masked so that positions do not attend to future tokens, preserving autoregressive properties [43,44]. This attention-alone model supports parallel processing, more accurate modeling of long-distance dependencies, and scaling that has revolutionized tasks from machine translation to generative AI.

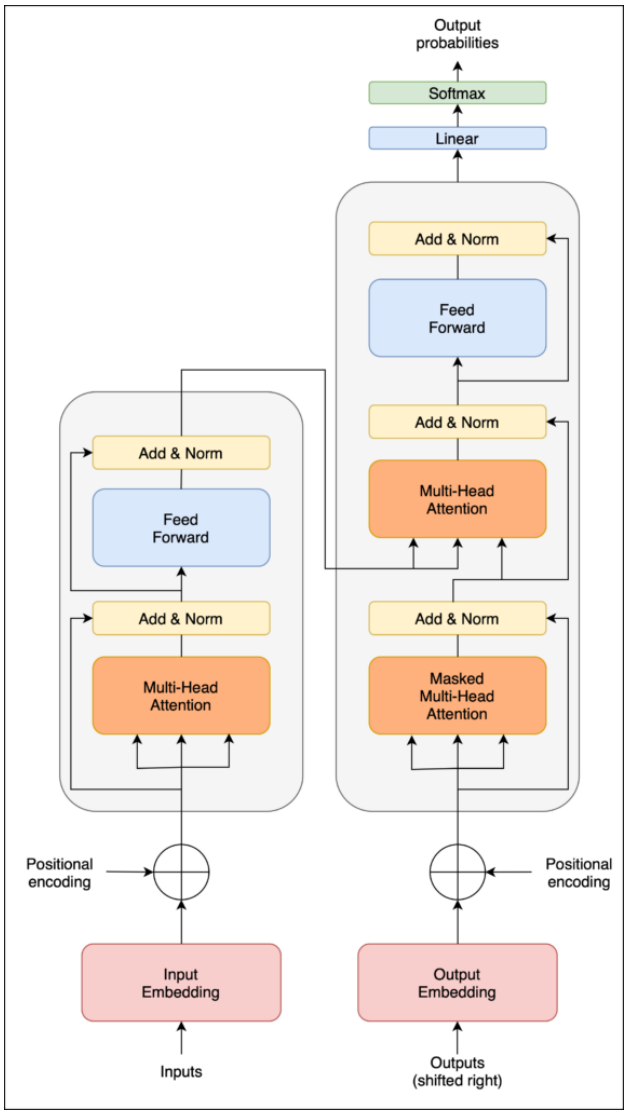

Figure 3.6 -The Transformer Architecture.

III.8.2 Self-Attention

Self-attention enables a model to dynamically weigh the relevance of all input tokens when processing each element in a sequence. Given an input, it projects three learned vectors per token: Query (Q), Key (K), and Value (V). The attention score between tokens is computed as the scaled dot product of Q and K, where higher scores indicate stronger relevance. These scores are normalized via Softmax to create attention weights (summing to 1), which then scale the Values (V) to produce a context-aware output (equation 3.17).

This allows each token to integrate information from the most relevant parts of the sequence [42, 43].

$$Attention(Q,K,V) = softmax\left(\frac{\mathrm{Q}K^T}{\sqrt{d_K}}\right)V \qquad (3.17)$$

Here the d_K is the dimension of the keys, and scaling by $\sqrt{d_K}$ prevents gradient instability from large dot products.

III.8.3 Multi-Head Attention

Multi-head attention conducts multiple self-attention "heads" in parallel with various learned projections of Q, K, and V. By dividing attention in various representation subspaces, the model can jointly capture different relationships (e.g., syntax, semantics, or long-range dependencies). The final representation is produced by concatenating all the heads' outputs and projecting them linearly (equation 3.18), enabling richer context modeling than single-head attention [43].

$$MultiHead(Q,K,V) = Concat(head_1,\ldots\ldots,head_h)W^O \quad (3.18)$$

$$\text{Where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Here, h is the number of heads. W_i^Q, W_i^K,W_i^V Are learned head-specific projections, and W^O combines the outputs.

III.8.4 Transformers Usage

The Transformer architecture can be adapted into three primary configurations, each tailored for different tasks [44]:

- **Encoder-Only Transformer:** Consists only of the encoder blocks from the original transformer architecture shown in Figure 3.6. It's often used for Natural Language Understanding (NLU) tasks like text classification. Some common examples of this architecture are BERT, RoBERTa, and DistilBERT.
- **Decoder-inly Transformer:** Uses only the decoder blocks. Instead of attending to encoder output, it uses causal/self-attention (masked) to prevent attending to future tokens. It's used for sequence generation like language modeling. Some common examples of this architecture are GPT series, LLaMA.
- **Encoder-Decoder Transformer:** Both encoder and decoder are used as explained in section 3.8.1. It's typically used for machine translation, summarization. Some examples of this architecture are T5 (Test-To-Text Transfer Transformer), and BART(Bidirectional and Auto-Regressive Transformers).

III.9 Conclusion

This chapter discussed fundamental concepts in machine learning and deep learning, and their respective roles in allowing systems to learn from data and improve over time. It described the utilization of traditional machine learning algorithms as tools for pattern recognition and decision-making, and the power of deep learning algorithms to victories abstractions using layered architectures based on neural mechanisms, thereby allowing them to deal with varied and complex unstructured inputs.

Aside from machine learning and its applications, it provided an introduction to the Transformer architecture and attention mechanism, underscoring their role in capturing relative relationships in the presented data. Because of these advances, attention mechanisms became central to many of the state-of-the-art models, particularly ones that require a comprehensive understanding of sequences or events that are interrelated.

Finally, the chapter discussed how these concepts and techniques have been implemented in the practical domain of cybersecurity, concentrating on the particular sub-domain of alert correlation. The fact that modern detection systems present massive volumes of alerts means traditional manual or rule-based approaches will not stand for long. In this regard, machine learning, represents a more flexible and efficient model to correlate alerts, recognize malicious activity patterns, and prioritize threats.

Chapter IV

Alert Correlation Using Attention Mechanism

IV.1 Introduction

One of the most critical challenges faced by Intrusion Detection Systems (IDS) today is the high rate of false positives alerts that mistakenly indicate security issues, and unnecessarily require security analyst attention. Studies have shown that up to 99% of IDS alerts may be unrelated to actual security issues [47]. The challenge is to reduce the number of false positives and improve the quality of alerts.

In this chapter we present our approach for false positive reduction using attention mechanism, which capturing dependencies and relevance across the input features, allowing the model to distinguish between malicious and benign traffic more effectively. Unlike traditional techniques, our approach does not rely on prior knowledge of specific attacks or normal behavior, making it suitable for identifying unknown or evolving threats with greater accuracy.

IV.2 Alert correlation

To address the problem of the false positive generated from the IDS, some efforts have been made both in academic and industrial communities referred to as Alert Correlation. Alert correlation is a process that analyzes the alerts produced by one or more intrusion detection systems and provides a more succinct and high-level view of occurring or attempted intrusions.

In [48], alert correlation is regarded as too complex to be effectively addressed in a single phase. Instead, it is more appropriately viewed as a framework composed of several interrelated components. In their work, the authors identify six key components: Normalization, Aggregation, Correlation, False Alert Reduction, Attack Strategy Analysis, and Prioritization. For each of these components, different techniques from various disciplines have been applied to enhance the overall effectiveness of the correlation process. In this work, we specifically focus on the False Alert Reduction component, aiming to minimize the number of false positives and improve the reliability of the alerts generated by intrusion detection systems.

IV.3 Related work

Several studies have explored alert correlation techniques to address the false alert reduction issue. In [49], the authors proposed a classification approach for reducing false positive alerts. The classification system uses three main sources of information: the network topology, the alert context (meaning similar alerts related to the current one), and the alert semantics, which refers to evidence based on the vulnerabilities of the target system. They obtained significant features from the input data to trained the classification model. For the classification task, they used a cost-sensitive version of Ripper rule learner, which provided a confidence score for both classes, whether the alert were true or false. During continual operation, the system will engage the security administrator for feedback whenever it is unsure, or not confident, and this feedback is used as labeled data to help the system learn and adapt to new or unseen representations or situations, which improves the classification performance and reduces the number of false positives over time.

Ikram ST, Cherukuri AK [50] proposes a hybrid intrusion detection model by integrating the principal component analysis (PCA) and support vector machine (SVM). The novelty of the paper is the optimization of kernel parameters of the SVM classifier using automatic parameter selection technique. This technique optimizes the punishment factor (C) and kernel parameter gamma (γ), thereby improving the accuracy of the classifier and reducing the training and testing time. The experimental results obtained on the NSL KDD.

The study in [51] explores the use of deep learning models to optimize anomaly-based IDS. By training on large datasets, the system improves its ability to generalize, thereby reducing false positives. The research demonstrates that deep learning can outperform traditional models in minimizing false alerts.

Paper [52] explores various feature selection methods applied to different classification algorithms across multiple datasets with the goal of reducing false positives in intrusion detection. By tailoring the feature selection techniques to each classifier and dataset, the study aims to enhance detection performance and improve the overall accuracy of identifying true threats by the proposed approach of cross correlation feature selection and apply it to a classifier.

IV.4 Architecture (Methodology)

The proposed model is based on a modified Transformer encoder architecture, tailored for tabular data Figure 4.1. The model consists of four main layers: Embedding layer, Positional Encoding Layer, Transformer Encoder Layer, and output layer.

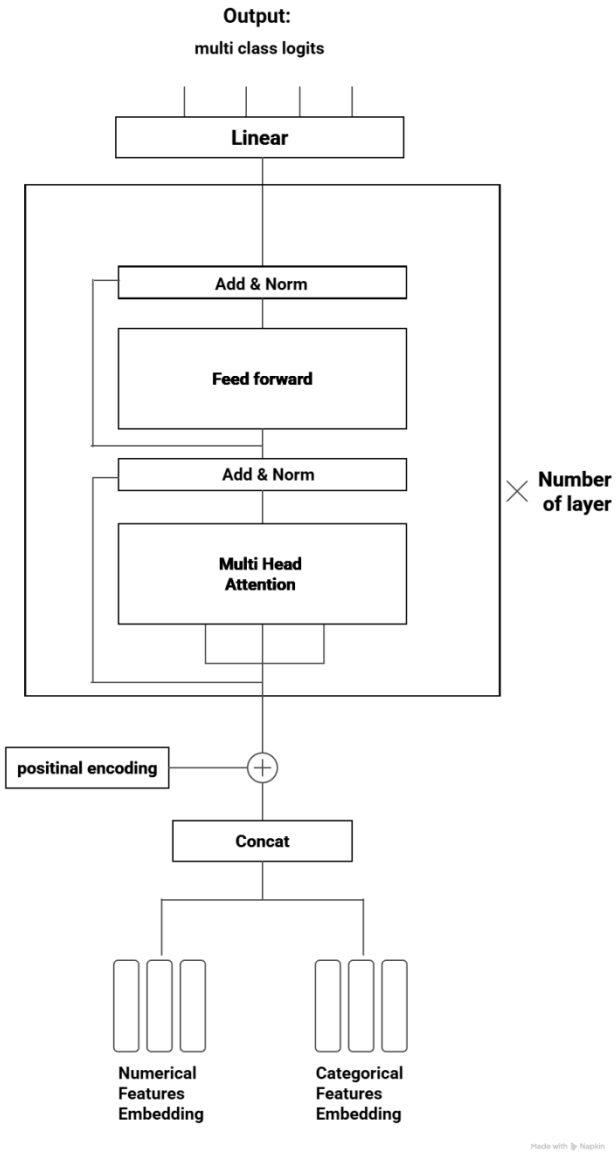

Figure 4.1 –The architecture of our model.

IV.4.1 Embedding Layer

The input features are divided into two types: categorical and numerical. Each type undergoes a separate embedding process. Categorical features are embedded using learnable embedding layers (see Listing 4.1), while numerical features are transformed using dense (linear) projections (see Listing 4.2).

```python
class CategoricalEmbedding(nn.Module):
    def __init__(self, cat_features: dict, embed_size: int):
        super().__init__()
        self.cat_embeds = nn.ModuleDict({
            feat: nn.Embedding(n_cat, embed_size)
            for feat, n_cat in cat_features.items()
        })
        self.feature_names = list(cat_features.keys())

    def forward(self, x_cat: torch.Tensor):
        embeddings = torch.stack([
            self.cat_embeds[feat](x_cat[:, i])
            for i, feat in enumerate(self.feature_names)
        ], dim=1)
        return embeddings
```

Listing 4.1 Categorical Embedding Module in PyTorch.

```python
class NumericalEmbedding(nn.Module):
    def __init__(self, num_features, embed_dim):
        super().__init__()
        self.proj = nn.ModuleList([
            nn.Linear(1, embed_dim) for _ in range(num_features)
        ])

    def forward(self, x):
        # x shape: [batch_size, num_features]
        out = [proj(x[:, i:i+1]) for i, proj in enumerate(self.proj)]
        return torch.stack(out, dim=1)
```

Listing 4.2 –Numerical Embedding Module in PyTorch.

The output of both the categorical and numerical embedding classes is a matrix of shape *(batch_size, num_features, embed_size)*. These two embeddings are then concatenated along the feature dimension to produce a unified embedding matrix.

IV.4.2 Positional Encoding Layer

Positional encoding is a technique used in Transformers to provide positional information to the model by adding position-dependent signals to word embeddings, allowing the model to incorporate the order of words in the input sequence. There are many variants of positional encoding. The original Transformer uses sine and cosine_functions for positional encoding [43], as shown in (equation 4.1):

$$PE_{pos,i} = \begin{cases} \sin\left(pos/10000^{2i/d_{model}}\right), & if\ i\ is\ even \\ \cos\left(pos/10000^{2i/d_{model}}\right), & if\ i\ is\ odd \end{cases} \quad (4.1)$$

Where *pos* is the position index in the sequence, i is the dimension index of the embedding vector, and d_{model} is the total dimension of the model's embedding space.

For our tabular data, positional encoding is added to incorporate information about feature positions, which is crucial in the absence of sequential data. The combined embeddings from embedding layer is subsequently passed through a positional encoding layer to incorporate information about feature positions (see Listing 4.3).

```
class PositionalEncoding(nn.Module):
    def __init__(self, d_model: int, dropout: float = 0.1, max_len: int = 5000):
        super().__init__()
        self.dropout = nn.Dropout(p=dropout)
        position = torch.arange(max_len).unsqueeze(1)
        div_term = torch.exp(torch.arange(0, d_model, 2) * (-math.log(10000.0) / d_model))
        pe = torch.zeros(max_len, 1, d_model)
        pe[:, 0, 0::2] = torch.sin(position * div_term)
        pe[:, 0, 1::2] = torch.cos(position * div_term)
        self.register_buffer('pe', pe)
    def forward(self, x: Tensor) -> Tensor:
        x = x + self.pe[:x.size(1)].transpose(0, 1)
        return self.dropout(x)
```

Listing 4.3 – Positional Encoding Module in PyTorch.

IV.4.3 Transformer Encoder Layer

In the context of tabular data modeling, the Transformer encoder is especially important for capturing complex interactions and dependence among different features. Unlike linear models that consider each feature independently or impose fixed interaction schema, the Transformer encoder takes advantage of self-attention mechanism (mentioned in section 3.8) which adaptively measures the relationships between all features, no matter its position or type (categorical or continuous). This would allow the model to discover which features are most important in combination for the prediction.

After embedding the features (categorical and numerical) into a common vector space, and pass it to the position encoder and the result is added to the original embeddings. The model feeds these embeddings into the encoder, which is composed of several key layers:

- **The Multi-Head Self-Attention layer:** allows each feature to attend to all other features, capturing dependencies and relevance across the input.
- **Feed-Forward Neural Network (FFN)**: that applies non-linear transformations independently to each feature vector, helping the model learn complex feature patterns.

- **Layer Normalization** and **residual (skip) connections:** are applied around both the attention and feed-forward blocks to improve training stability and gradient flow.

```python
class TransformerEncoderLayer(nn.Module):
    def __init__(self, d_model, num_heads, d_ff, dropout=0.1):
        super(TransformerEncoderLayer, self).__init__()
        self.self_attn = MultiHeadAttention(d_model, num_heads)
        self.ffn = FeedForwardNetwork(d_model, d_ff)
        self.norm1 = nn.LayerNorm(d_model)
        self.norm2 = nn.LayerNorm(d_model)
        self.dropout = nn.Dropout(dropout)

    def forward(self, x):
        # Self-Attention with Residual Connection
        x = x + self.dropout(self.self_attn(x, x, x))
        x = self.norm1(x)

        # Feedforward with Residual Connection
        x = x + self.dropout(self.ffn(x))
        x = self.norm2(x)

        return x
```

Listing 4.4 –Transformer Encoder Module in PyTorch.

IV.4.4 Output Layer

The output of the Transformer encoder is passed through a pooling mechanism, typically by computing the mean across the feature dimension, to produce a fixed-size vector. This flattened representation is then passed through a final dense (linear) layer for multi-class classification.

IV.5 DataSet

The NSL-KDD (Network Security Layer Knowledge Discovery Database) is an improved version of the KDDCUP'99 Dataset [1]. The KDDCUP'99 is a benchmark dataset in cybersecurity, created for the Third International Knowledge Discovery and Data Mining Tools competition [46] by the Defense Advanced Research Project Agency (DARPA), in 1999. The main objective of the competition was to build a network intrusion detector capable of differentiating between malicious connections (intrusions or attacks) and legitimate ones.

The dataset provides a standardized collection of audit data, which includes a wide variety of intrusions simulated in a military network environment.

The KDDCUP'99 dataset, despite being widely used for intrusion detection research, suffers from significant issues such as redundant records, class imbalance, and biased learning that make it hard for the model to perform and generalize well. Its drawbacks were worked upon through an upgraded version of the NSL-KDD dataset [45], which has unrolled its duplicates, balances the class distribution, and provides a more definitive basis for evaluating intrusion detection systems.

The NSL-KDD dataset has most of the inherent properties of its parent dataset. It is available in multiple versions, with KDDTrain+ and KDDTest+ containing 125,912 and 22,544 instances, respectively. For instance, every data point is labeled as either normal or an attack, with each attack being a DoS, U2R, R2L, or a Probe as shown in Table 4.1.

Table 4.1 -Detailed instances in the dataset.

NSL-KDD	Total instance	Normal	DoS	Probe	R2L	U2R
KDDTrain	125,912	67343	45927	11656	995	52
KDDTest	22,544	9711	7460	2885	2421	67

The dataset also comprises 41 features (3 nominal, 4 binary and 34 continuous), grouped in to basic, content, and Traffic features:

- **Basic attributes:** describe the basic information of a connection include duration, source, destination hosts, port number, and status flags.
- **Content attributes:** are drawn from the payload data of packets in network, such as lines like failed connection attempts and access frequency to control files.
- **Traffic attributes:** are derived from statistical analysis, for example, the number of connections to a common destination within a strange time window.

IV.6 Dataset Preprocessing

We concatenate the training and testing datasets to ensure consistent preprocessing. Specifically, we apply normalization to the numerical features and label encoding to the categorical features, which are subsequently used for embedding. The dataset contains no missing values and 610 duplicate records, which were dropped during preprocessing. It originally includes 30 traffic classes as shown in table 4.2, which are mapped into five broader categories: four main attack types Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R), and one class representing normal traffic, then apply label encoding to the target feature. Finally the dataset is split into training, testing, and validation sets.

Table 4.2 -Attack Traffic classes in dataset.

Attack Class	Attack Type
DoS	back, land ,neptune, pod, smurf, teardrop, mailbomb, processtable, udpstorm, apache2, worm
Probe	satan, iPsweep, nmap, portsweep, mscan, saint
R2L	guess_password, ftp_write, imap, phf, multihop, warezmaster, xlock, xsnoop, snmpguess, snmpgetattack, httptunnel, sendmail, named
U2R	buffer_overflow, loadmodule, rootkit, rerl , sqlattack, xterm, ps

IV.7 Training

The model was implemented and trained in Google Colab using an NVIDIA T4 GPU, with PyTorch version 2.6.0+cu124.

IV.7.1 Libraries Used

IV.7.1.1 Pandas

Pandas[1] is a Python library (see Figure 4.2) that provides powerful and flexible data structures and tools for data analysis, manipulation, and cleaning. Pandas is built on top of numpy, which is a Python library that supports large, multi-dimensional arrays and matrices. It offers a wide range of features including handling missing data, grouping and aggregating, merging and joining datasets, reshaping and pivoting data, slicing and indexing, and reading

[1] https://pandas.pydata.org/

from or writing to various file formats. Due to its versatility and ease of use, Pandas is widely adopted in data science, machine learning, and statistical analysis**.**

Figure 4.2 -Pandas library for Python.

IV.7.1.2 Pytorch

PyTorch[2] is an open-source deep learning framework developed by Facebook's AI Research lab, used for applications such as computer vision and natural language processing. PyTorch (see Figure 4.3) is built on the Torch library and uses dynamic computation graphs, which allow users to change the network architecture during runtime, making it particularly useful for research and experimentation.

Figure 4.3 -Pytorch library logo.

IV.7.1.3 Scikit-learn

Scikit-learn[3] (usually imported as sklearn) is a powerful Python machine learning library which is free to use. Scikit-learn is built on NumPy, SciPy, and matplotlib, and provides simple, efficient tools for classification, regression, clustering, dimensionality reduction, model selection, and preprocessing. Scikit-learn (see Figure 4.4) implements many of the popular algorithms you may have seen such as Support Vector Machines (SVM), Random Forests, k-Nearest Neighbors (k-NN), and logistic regression. Scikit-learn is also known for its clean and consistent API, making it ideal for both beginners and advanced users.

[2] https://pytorch.org/
[3] https://scikit-learn.org/

Figure 4.4 -Sklearn library logo.

IV.7.2 Hyperparameters

We train the model using the following hyper parameters:

- We set the embedding size to 128, the number of multi-head attention heads to 8, the dimension of hidden layer of the feed forward to 2048, and the number of transformer encoder layers to 2
- We used 60 epochs while the model was kept for 15 epochs
- We used a batch size of 16

IV.7.3 Loss function

Our problem is a multi-class classification task; therefore, we use the multi-class cross-entropy loss function (equation 3.14). It directly measures the difference between the true class and the model's predicted probability distribution. It encourages the model to assign high probabilities to the correct class and penalizes confident wrong predictions more heavily, leading to better and faster learning.

IV.7.4 Optimization

We optimized the model using the Adam optimization with a learning rate of 0.0001, a $\beta 1$ of 0.9, and a $\beta 2$ of 0.999, which provides adaptive learning rates for each parameter and accelerates convergence through momentum-based updates.

IV.8 Results and Discussion

IV.8.1 Evaluation Metrics

- **Confusion matrix:** A table that visualizes the performance of the classification model by showing the counts of true positives (TP), true negatives (TN), false positives (FP),

and false negatives (FN). It helps in understanding the types of errors the model makes.

- **Accuracy:** accuracy refers to the percentage of correct predictions made by a model. It is the ratio of correctly predicted instances to the total number of instances (equation 4.2).

$$Acc = \frac{TP+TN}{TP+TN+FP+FN} \quad (4.2).$$

- **Precision:** Indicates how many of the instances predicted as positive are actually positive (equation 4.3).

$$Precision = \frac{TP}{TP+FP} \quad (4.3)$$

- **Recall:** known also as sensitivity, it Measures how well the model identifies all actual positive instances (equation 4.4).

$$Recall = \frac{TP}{TP+FN} \quad (4.4)$$

- **F1 score:** known also as F-mesure, F-score. It is the harmonic mean of precision and recall (equation 4.5).

$$F1score = 2.\frac{Precision.Recall}{Precision+Recall} \quad (4.5)$$

IV.8.2 Baselines

We compare our model with the following baselines:

- **PCA and SVM** [50] Hybrid model by integrating the principal component analysis (PCA) dimension reduction and (SVM) for classification in NSL-KDD dataset.

- **LSTM (Long Short-Term Memory)** networks for false positive reduction in anomaly detection [51]. LSTM models can learn the temporal dependencies and contextual patterns of normal vs. abnormal behavior, allowing them to understand sequences leading to true attacks**,** and ignore harmless out-of-pattern behaviors that resemble attacks
- **Feature Selection Based on Cross-Correlation for the Intrusion Detection** [52] explores various feature selection methods applied to different classification algorithms across multiple datasets with the goal of reducing false positives in intrusion detection. The author compares his proposed method (CCFS for feature selection and different classifier) with other existing methods (MIFS, CFS with

different classifier) to evaluate its effectiveness in improving classification performance.

IV.8.3 Result

Figure 4.5 presents the training and validation accuracy and loss curves across epochs. The figure demonstrates how the model's performance evolved during training. Both training and validation accuracy steadily improved over epochs, with the training accuracy slightly outperforming the validation accuracy. Similarly, the validation loss remained slightly higher than the training loss. This mild discrepancy is typical and suggests the model is generalizing well without significant overfitting.

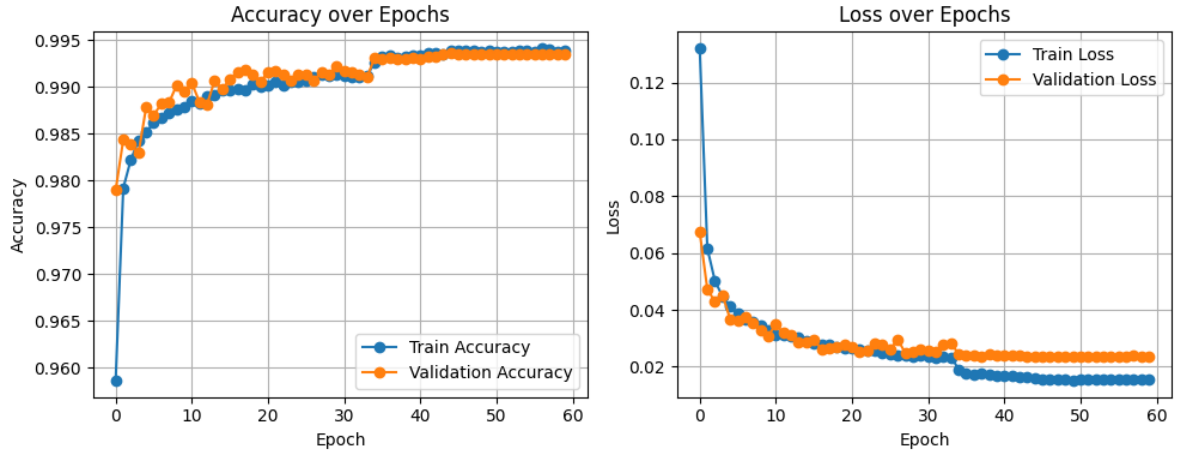

Figure 4.5 -Training and Validation Accuracy and Loss curves across epochs.

The confusion matrix in Figure 4.6 describes how the model performed on the test dataset. The matrix breaks down the model's classification success by number of true positives, true negatives, false positives, and false negatives, for each of the classes in the model. The confusion matrix provides an initial impression of how successful the model is at distinguishing between classes and show where the model is strong and weak, as well any class predictions that may be imbalanced.

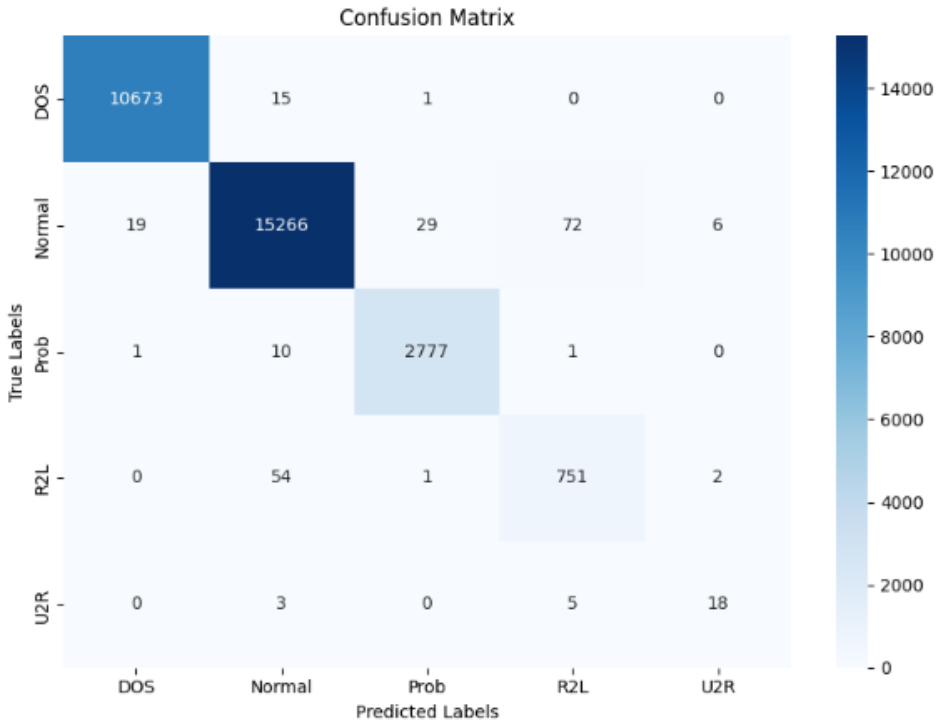

Figure 4.6 -Confusion matrix on the test dataset.

We compared our model's performance with the methods cited in [52], which were also evaluated against the model proposed in that paper. This comparison allows us to assess how our approach performs relative to both existing techniques and the baseline established in [52]. We also compared our model with the approach proposed in [51], as well as the other methods evaluated in that paper [50]. This helps provide a broader perspective on how our model performs against various existing techniques. The result of MIFS, CFS, and CCFS methods for DT, SVM, KNN, NB classifiers on NSL-KDD with 20 features, along with our model using all 41 features, are presented in Table 4.3. Furthermore, the performance of LSTM, PCA with SVM methods and our method is shown in Table 4.4.

Table 4.3-Comparison of the MIFS, CFA, and CCFS methods for DT, SVM, KNN, and NB classifiers on the NSL-KDD dataset with 20 features, and our method with 41 features.

Method	Accuracy	Precision	Recall	F1 score
MIFS+DT	91.21	93.52	93.45	93.49
MIFS+SVM	87.80	88.17	87.21	87.69
MIFS+KNN	89.79	89.66	89.61	89.64
MIFS+NB	87.35	89.27	89.27	89.27
CFS+DT	93.12	94.82	94.82	94.82
CFS+SVM	90.15	91.18	91.18	91.18

CFS+KNN	91.50	92.46	92.46	92.46
CFS+NB	89.24	90.34	91.18	90.76
CCFS+DT	93.47	94.87	95.79	95.33
CCFS+SVM	90.69	92.39	92.12	92.25
CCFS+KNN	93.09	94.39	94.35	93.37
CCFS+NB	90.87	92.27	92.27	92.27

Table 4.4 -Comparison of the LSTM, PCA+SVM methods and our method on the NSL-KDD dataset with 41 features.

Method	Accuracy
LSTM	96.7
PCA+SVM	87
TransformerEncoder	99.3

IV.8.4 Discusion

Our proposed TransformerEncoder method has a slightly better performance than the compared methods shown in Tables 4.3 and 4.4. One likely reason for the improved results is the attention mechanism, which captures dependencies between features, something that traditional models, which often process features independently, fail to address. This ability to model feature interactions more effectively leads to more accurate classification. The percentage improvement of our model over the other methods is summarized in Table 4.5.

Table 4.5 -Percentage improvement of our model over the other methods.

Method	Accuracy	Precision	Recall	F1 score
MIFS+DT	8.09	5.78	5.85	5.81
MIFS+SVM	11.5	11.13	12.09	11.61
MIFS+KNN	9.51	9.64	9.69	9.66
MIFS+NB	11.95	10.03	10.03	10.03
CFS+DT	6.18	4.48	4.48	4.48
CFS+SVM	9.15	8.12	8.12	8.12
CFS+KNN	7.8	6.84	6.84	6.84
CFS+NB	10.06	8.96	8.12	8.54
CCFS+DT	5.83	4.43	3.51	3.97

CCFS+SVM	8.61	6.91	7.18	7.05
CCFS+KNN	6.21	4.91	4.95	5.93
CCFS+NB	8.43	7.03	7.03	7.03
LSTM	2.6	/	/	/
PCA+SVM	12.3	/	/	/

IV.9 Conclusion

In this work, we introduced an attention-based method that utilizes a Transformer Encoder to reduce false positives through alert correlation. By leveraging the self-attention mechanism to allow our model to identify and learn complex relationships among features, our model can classify alerts more accurately. Rather than treating the features as independent of one another, we learn the contextual relationships among alerts to improve her ability to separate true and false alerts. Our experimental results indicate that our method performs competitively when compared to existing methods, highlighting its potential as a robust solution for enhancing the reliability of IDSs through improved alert correlation and false positive reduction.

General Conclusion

In this paper, we introduced an intelligent alert correlation framework aimed to decrease false positives in intrusion detection systems. The proposed model employs an attention mechanism to learn dependencies and contextual relations of alert features which allows it to better distinguish between benign and malicious network activity. In contrast to traditional approaches, our method does not require prior knowledge of specific attacks as well as taking on new attacks.

The experimental results on the NSL-KDD dataset have shown us that although we obtain a reduction in non-related alerts, we also increase the accuracy for intrusion detection which aids the security analyst in identifying threats and helps them deal with lesser alerts. Future work could be directed to enhance the quality of the dataset by addressing class imbalance, which is one of the primary ways to increase detection performance. Creating a balanced and representative dataset will allow the model to learn about types of attacks more effectively and generalize better across all types of attack.

Additionally, in a real-world situation, attackers often employ complex, coordinated strategies involving a sequence of individual actions known as "attack scenarios" or "attack plans". While most of these actions are individually signaled by intrusion detection systems, the dependencies and temporal relationships between them often go undetected.

In this consideration, attention mechanisms can be further studied to identify both. sequenced dependencies and contextual relationships between alerts, in which allows the model not only to correlate individual alerts, but also to predict more elaborate multi-step attack strategies. This would greatly outstretch the capabilities of the system to detect elaborate, emergent threats and provide recommendations earlier and more accurately into potential intrusion.

Bibliography

[1] Priyank S, Kreena M, Shikha S, “Network Security”, International Journal of Scientific and Research Publications, Vol. 3, Issue 8, August 2013

[2] Shailja P, “Modern network security: issues and challenges”, International Journal of Engineering Science and Technology, Vol. 3 No. 5 May 2011

[3] Faisal A, “THE ANATOMY OF A CYBER ATTACK: DISSECTING THE CYBER KILL CHAIN (CKC)”, Scientific and Practical Cyber Security Journal (SPCSJ),

[4] Ali .A, Sulaiman .A, Mohamed Z, Denial-of-Service, Probing, User to Root (U2R) & Remote to User (R2U) Attack Detection using Hidden Markov Models, International Journal of Computer and Information Technology Vol.07-issue 05, September 2018

[5] Merouane M, “reseaux Bayesiens pour detection d’intrusions dans un reseau informatique”, PhD thesis, 2010

[6] Karen S, Perer M, Guide to Intrudion Detection and Prevention Systems (IDPS), February 2007

[7] Tayeb Kenza,Modéle graphique probabilistes pour la correlation d’alertes en detection d’intrusions, PhD thesis, 2011

[8] H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion detection systems. Computer Networks, Elseiver, pages 805–822, 1999

[9] A. Ghorbani, W. Lu, and M. Tavallaee, Network Intrusion Detection and Prevention, ser. Advances in Informatio Security. Springer, 2010, vol. 47

[10] N. Stakhanova, S. Basu, J. Wong, A taxonomy of intrusion response system, International Journal of Information and Computer Security, January 2007

[11] P. M. Mell, “An Overview of Issues in Testing Intrusion Detection Systems, NISTIR-7007,” 2003.

[12] Ben Mustapha Y, Corrélation d’alertes : un outil plus efficace d’aide à la décision pour répondre aux intrusions,PhD thesis, 2015.

[13] Saeed Salah, Gabriel Maciá-Fernández, Jesús E. Díaz-Verdejo,A model-based survey of alert correlation techniques,Computer Networks,Volume 57, Issue 5,2013,Pages 1289-1317,ISSN 1389-1286.

[14] Pouget Fabien, Dacier Marc, Alert Correlation: Review of the state of the art. EURECOM, Technical Report, (2003)

[15] H. Debar, D. Curry, and B. Feinstein, “The Intrusion Detection Message Exchange Format (IDMEF),” http://tools.ietf.org/pdf/rfc4765.pdf, 2007.

[16] H. Debar, B Morin, F. Cuppens, F. Autrel, L. Mé, B. Vivinis, S. Benferhat, M. Ducassé, R. Ortalo, Détection d'intrusion: correlation d'alertes, 2004

[17] S. Al-Mamory, H. Zhang, A survey on IDS alerts processing techniques. In: Proceeding of the 6th WSEAS International Conference on Information Security and Privacy (ISP), pp. 69-78 (2007)

[18] A. Valdes and K. Skinner, Probabilistic Alert Correlation, in Recent Advances in Intrusion Detection, ser. Lecture Notes in Computer Science, W. Lee, L. Mé, and A. Wespi, Eds., vol. 2212. Springer Berlin Heidelberg, 2001, pp. 54–68.

[19] K. Ilgun, R. A. Kemmerer, and Ph. A. Porras. State transition analysis : A rule-based intrusion detection approach. IEEE Transactions on Software Engineering, 21 :181–199, 1995.

[20] F. Cuppens and R. Ortalo, "LAMBDA: A Language to Model a Database for Detection of Attacks ," Recent Advances in Intrusion Detection, vol. 1907, pp. 197–216, 2000.

[21] C. Michel and L. Mé. Adele : An attack description language for knowledge-based intrusion detection. In In Proceedings of the 16th International Conference on Information Security , 2001

[22] S. J. Templeton and K. Levitt, "A requires/provides model for computer attacks," in Proceedings of the 2000 Workshop on New Security Paradigms, ser. NSPW '00. ACM, Ireland, 2000, pp. 31–38.

[23] F. Cuppens, "Managing alerts in a multi-intrusion detection environment," in Proceedings of the 17th Annual Computer Security Applications Conference, ser. ACSAC '01. IEEE Computer Society, December Washington, DC, USA, 2001.

[24] B. Morin, L. Mé, H. Debar, and M. Ducassé, M2D2: A formal data model for ids alert correlation, in In Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002), 2002

[25] B. Morin, L. Mé, H. Debar, M. Duccassé, M4D4: a Logical Framework to Support Alert Correlation in Intrusion Detection, Information Fusion, vol. 10, pp. 285–299, 2009

[26] Michela Pellicelli, Managing the supply chain: technologies for digitalization solutions, The Digital Transformation of Supply Chain Management, pp 101-152 (2023).

[27] chloé-agathe azencott, introduction au machine learning, Paris [France] : Dunod, 2018.

[28] Batta Mahesh, Machine Learning Algorithms - A Review, International Journal of Science and Research (IJSR) ISSN: 2319-7064, 2018.

[29] Jain, A. K., Murty, M. N., & Flynn, P. J. Data clustering**:** A review. *ACM Computing Surveys (CSUR)*, 31(3), 264–323(1999).

[30] L.J.P. van der Maaten , E.O. Postma, H.J. van den Herik, Dimensionality Reduction: A Comparative Review,2008

[31] Agrawal, R., Imieliński, T., & Swami, A**,** Mining association rules between sets of items in large databases, (1993).

[32] Rachna Vaish, U.D. Dwivedi, Saurabh Tewari, S.M. Tripathi, Machine learning applications in power system fault diagnosis: Research advancements and perspectives, Engineering Applications of Artificial Intelligence,Volume 106, 2021,

[33] P. Jonathon Phillips et al. Four Principles of Explainable Artificial Intelligence. en. 2021.

[34] Ninareh Mehrabi et al. "A Survey on Bias and Fairness in Machine Learning". In: ACM Comput. Surv. 54.6 (2021).

[35] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016

[36] Rosenblatt, F. **,** *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*. *Psychological Review*, 65(6), 386–408,(1958)

[37] Bishop, C. M, *Pattern Recognition and Machine Learning*, Springer, (2006).

[38] Siddharth Sharma, Simone Sharma,Anidhya Athaiya, activation functions in neural networks, International Journal of Engineering Applied Sciences and Technology, 2020.

[39] B.L. Kalman and S.C. Kwasny. Why tanh: choosing a sigmoidal function, IJCNN International Joint Conference on Neural Networks. Vol. 4. 1992.

[40] Terven, Juan, Cordova-Esparza, Diana-Margarita, Ramirez-Pedraza, AlfonzoChávez , Edgar, Loss Functions and Metrics in Deep Learning. A Review,2023.

[41] Li, Mingfeng, Comprehensive Review of Backpropagation Neural Networks, academic Journal of Science and Technology, 2024

[42] Zhaoyang Niu, Guoqiang Zhong, Hui Yu, A review on the attention mechanism of deep learning, 2021.

[43]Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, Attention Is All You Need, 2017.

[44] Tianyang Lin, Yuxin Wang, Xiangyang Liu, Xipeng Qiu,A survey of transformers,

AI Open,Volume 3,2022.

[45] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani "A Detailed Analysis of the KDD CUP 99 Data Set", Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009)

[46] KDD Cup 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/

[47] Stefan Axelsson. Understanding Intrusion Detection Through Visualization. PhD thesis, Chalmers University of Technology, 2005.

[48] Reza Sadoddin, Ali Ghorbani, Alert Correlation Survey: Framework and Techniques, 2006.

[49] T. Pietraszek, Using adaptive alert classification to reduce false positives in intrusion detection, 2004.

[50] Ikram ST, Cherukuri AK. Improving accuracy of intrusion detection model using PCA and optimized SVM. J Comput Inf Technol. 2016;24(2):133–48.

[51] Khloud Al Jallad1 , Mohamad Aljnidi ,Mohammad Said Desouki, Anomaly detection optimization using big data and deep learning to reduce false-positive, Journal of big data, 2020.

[52] Gholamreza Farahani, Feature Selection Based on Cross-Correlation for the Intrusion Detection System, 2020.