
**ADDIS ABABA UNIVERSITY**

**ADDIS ABABA INSTITUTE OF TECHNOLOGY - AAiT**

**SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING – SiTE**

# A Multimodal Security Information and Event Management Solution Empowered by Deep Learning and Alert Fusion

---

*Author*:
Behailu Adugna Wondimneh

*Advisor*:
Sileshi Demisie (Phd)

*A thesis submitted in fulfillment of the requirements for the degree of MSc in*

**Cyber Security**

November 2024

# Approval Sheet

This certification affirms that the thesis authored by Behaylu Adugna Wondimneh, titled "Enhanced Multimodal Intrusion Detection for SIEM System through Deep Learning and Alert Fusion" has been submitted to satisfy the partial requirements for the Master of Sciences degree in Cyber Security. The document meets to university regulations and attains accepted standards in terms of originality and quality.

**Approved by Board of Examiners**

| Name | Signature | Date |
|---|---|---|
| _____ (School Dean) | _____ | _____ |
| Sileshi Demisie (Phd) (Advisor) | _____ | _____ |
| _____ (External Advisor) | _____ | _____ |
| _____ (Internal Advisor) | _____ | _____ |

## Acknowledgements

First and foremost, I extend my deepest gratitude to God the Almighty, whose boundless grace, wisdom, and strength have guided me through every stage of this research. His unwavering presence has been my source of inspiration and resilience in overcoming challenges.

I am profoundly thankful to my family, whose sacrifices have made this journey possible. Their belief in me has been a driving force, and their love has provided me with the strength to persevere.

A special word of appreciation goes to my dear friend Tirufat whose support, motivation, and insightful discussions have greatly contributed to the progress of this research. Her encouragement and belief in my abilities have been invaluable throughout this journey.

To all who have supported me directly or indirectly, I extend my heartfelt appreciation.

## Declaration of Authorship

I, Behailu Adugna Wondimneh, declare that this thesis titled, "Enhanced Multimodal Intrusion Detection for SIEM System through Deep Learning and Alert Fusion" and the work presented in it are my own. I confirm that: This research was conducted primarily during my pursuit of a master's degree in Cyber Security at Addis Ababa University (AAU), Addis Ababa Institute of Technology (AAiT)

- No portion of the thesis has been previously submitted for any degree or qualification at Addis Ababa University or another academic institution.
- Proper attribution is consistently provided for all referenced published works.
- All quoted material from external sources is accompanied by appropriate citations. Apart from these quotations, the content of this thesis is original and solely my own work.
- I have duly acknowledged all significant sources of assistance received during the course of this research.

| Name | Signature | Date |
|---|---|---|
| Behaylu Adugna Wondimneh<br>(Name of Student) | _____ | _____ |

This thesis has been submitted for examination with approval as a university advisor.

| | | |
|---|---|---|
| Sileshi Demisie (Phd)<br>(Name of Advisor) | _____ | _____ |

## Abstract

The cybersecurity threat landscape is marked by a growing number of increasingly complex and sophisticated attacks affecting organizations across various sectors. In response, solutions like SIEM systems are essential for providing centralized threat detection, real-time analysis, and compliance support, making them integral to modern cybersecurity strategies. One of the reasons for this is that SIEM solutions collect and aggregate log data from across an organization's IT infrastructure, providing a single pane of glass for monitoring security events. And this centralized approach is essential for identifying threats that span multiple systems and environments, identifying indicative patterns of attacks such as privilege escalation and polymorphic malware, helping proactively identify signs of unusual data accesses or exfiltration before significant damage occurs. Furthermore, SIEM solutions support compliance by maintaining detailed audit logs and providing preconfigured reporting tools.

However, SIEM systems usually encounter significant challenges in effectively identifying and responding to sophisticated cyberattacks. Since they rely heavily on predefined rules, even if complex correlations, and signatures, they struggle to adapt to novel attack techniques that do not match the predefined patterns. They often lack sophisticated analytics capabilities such as deep learning and behavioral analysis, which deprives them of the effectiveness at detecting advanced threats. Furthermore, they frequently produce an overwhelming volume of alerts, many of which are irrelevant or false positives. This leads to alert fatigue, causing cybersecurity analysts to become desensitized to alerts and increase the risk of overlooking critical incidents.

This research proposes a multimodal architecture of SIEM designed to overcome current limitations in threat detection by integrating diverse data sources, including network traffic and event logs. The solution utilizes advanced neural networks to analyze intricate relationships within network connection features and their temporal dependencies. By further employing alert fusion, it creates a melting-pot for alerts from different sources that can provide a more comprehensive and complementary understanding of potential threats that can address the issue of false positives.



***Keywords*:- Security Information and Event Management, Intrusion Detection Systems, Machine Learning, Deep Learning, Alert Fusion**

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

1. DDoS – Distributed Denial of Service
2. DoS – Denial of Service
3. IDS – Intrusion Detection System
4. ML – Machine Learning
5. MLP – Multi-layer Perceptron
6. NIDS – Network Intrusion Detection System
7. SIEM – Security Information and Event Management

# Chapter 1

## 1. Introduction

### 1.1. Overview

In today's interconnected and data-driven environment, network security is essential for protecting digital infrastructures from a range of threats. Organizations are increasingly aware of these risks, strengthening their security measures and adopting proactive defense strategies in response. However, cybercriminals continue to advance their tactics, constantly developing new methods to exploit vulnerabilities in digital systems and finding ways to circumvent traditional prevention and detection measures. Notably, Distributed Denial of Service (DDoS) attacks have become a significant, recurring challenge for network administrators and security professionals. One of the main reasons for this shift can be attributed to geopolitical upheavals, particularly following the Russian invasion of Ukraine in February 2022 and the impact of the 2016 U.S. election [1].

With this regard, cybersecurity threats to systems such as industrial control systems (ICS) have surged significantly in recent years, driven primarily by heightened activity from nation-states and cybercriminals. Attackers have grown increasingly sophisticated and dangerous, making timely and effective detection a serious challenge. Current examples of cybersecurity incidents impacting IT and ICS include: ransomware attacks; malware disrupting business and operational functions; phishing campaigns targeting executives, executive assistants, SCADA engineers, IT administrators, and other privileged users; business email compromises, such as account takeovers or executive impersonation; data leaks and theft; and social engineering tactics to extract sensitive information from personnel [2].

To address these complex and dangerous cyber-attacks, there is a high demand for more advanced and cutting-edge cybersecurity solutions. Cybersecurity is crucial in safeguarding critical ICT infrastructures against threats of this scale. Increased awareness around security has led many organizations to invest heavily in protecting their networks with sophisticated cybersecurity measures, emphasizing proactive defense mechanisms rather than relying on the previous, passive, and reactive approaches [1].

SIEM is a security monitoring system designed to enhance situational awareness within a network, addressing the aforementioned challenges [2]. SIEM helps in consolidating the analysis effort by a variety of security solutions in such a way that creates a unified view of what really is happening. Furthermore, SIEM aids analysts in developing security policies and managing syslog events across various security devices and solutions.

However, SIEM systems, which mostly use rule-based or signature-based correlation analysis, and modern IDS solutions, with their signature and anomaly-based detection capabilities, are increasingly challenged by the rapidly evolving landscape of cyber-attacks, making it difficult, if not impossible, to keep pace [3].

While efforts exist to implement more advanced and predictive machine learning techniques for intrusion detection in SIEM, these models require substantial computational resources for training and calibration. Additionally, SIEM solutions must allocate resources to other demanding tasks, such as event collection and normalization, to support advanced analytics modules. This makes integrating machine learning into such systems particularly challenging, especially in terms of computational complexity [4]. This operational focus on other high-priority tasks causes SIEM implementers to emphasize syntax rather than semantics, often resulting in the use of correlation languages with limited features. As a result, current SIEM correlation rules are relatively simplistic, primarily relying on basic Boolean logic to link events that follow a specific attack path. Only a few SIEM solutions incorporate advanced correlation engines capable of analyzing deviations and historical correlations, which are crucial for identifying instances that warrant investigation after detecting a zero-day attack [2].

On the other hand, IDSs, event logs of which are ingested by SIEM for further correlation, generate a large volume of events and numerous false positives, making it challenging to identify the real attacks among these alerts. As a result, handling the alerts produced by these systems often overwhelms security teams in many organizations. To address this, a more modern approach focuses on the integration and cooperation of multiple IT security tools, especially IDSs, to reduce false positives and provide a more comprehensive understanding of network activity [5].

Even those SIEM and IDS solutions, being researched in the communities, that incorporate dynamic analysis techniques, such as machine learning, are experiencing a high rate of false positives. This issue arises because their training and test sets are not specifically tailored for the unique context of SIEM and IDS applications [6].

Furthermore, [7] discusses that, in today's fast-evolving digital landscape, cybersecurity threats are becoming more complex, often using multiple attack vectors that exploit both text and visual elements. Conventional cybersecurity monitoring, detection and prevention methods, which generally analyze these data types separately, face challenges in effectively detecting and mitigating such sophisticated, multi-modal attacks. This limitation has contributed to a rise in successful breaches that costs the global community a data breach reaching millions of dollars in 2022.

Most current studies analyze traffic data with limited consideration of its full complexity. In recent studies, researchers recognize the complex relationships among features describing network connections and apply unsupervised learning methods, such as auto-encoder neural networks, to extract intermediate representations. However, with advancements in network infrastructure and architecture, feature heterogeneity is increasing, making it challenging for traditional neural networks to capture meaningful information from traffic data's domain knowledge due to their simple structure. These methods either overlook or simplify the complex features within a network connection and the temporal information between connections, leading to models that inevitably miss critical traffic data insights and rely on incomplete feature information for classification [8].

Our study proposes a SIEM architecture that incorporates a multi-modal threat analysis approach. The architecture leverages additional data sources such as network traffic and correlation analysis with the employment of deep learning techniques to enhance detection capabilities. Then finally, by implementing alert fusion technique called majority voting, the architecture aims to minimize false positives and improve overall detection accuracy.

## 1.2. Background

In this section, we will discuss foundational concepts about SIEM system. Furthermore, the architecture of the existing and typical kind of SIEM solution is discussed with its components and sub-systems elaborated.

### 1.2.1. Security Information and Event Management (SIEM)

SIEM plays a crucial role in network defense as comprehensive security solutions. SIEM solutions usually collect event logs from different sources across an organization's IT infrastructure using either agents (agent-based collection) or by simply configuring the sources to send the events to a predefined storage system (agentless collection). Acting as the central hub for security monitoring, SIEM processes logs, correlates events, and applies predefined correlation rules to detect potential threats. Through advanced analysis, SIEM systems identify anomalies and potential risks, generating alerts or notifications. These alerts are then directed to administrators or security teams, enabling timely responses to emerging security issues [9].

The architecture of a SIEM system, as shown in *Figure 1,* below comprises the following major components: data sources, collector, parser, storage, correlation engine and dashboard (visualization) [10].

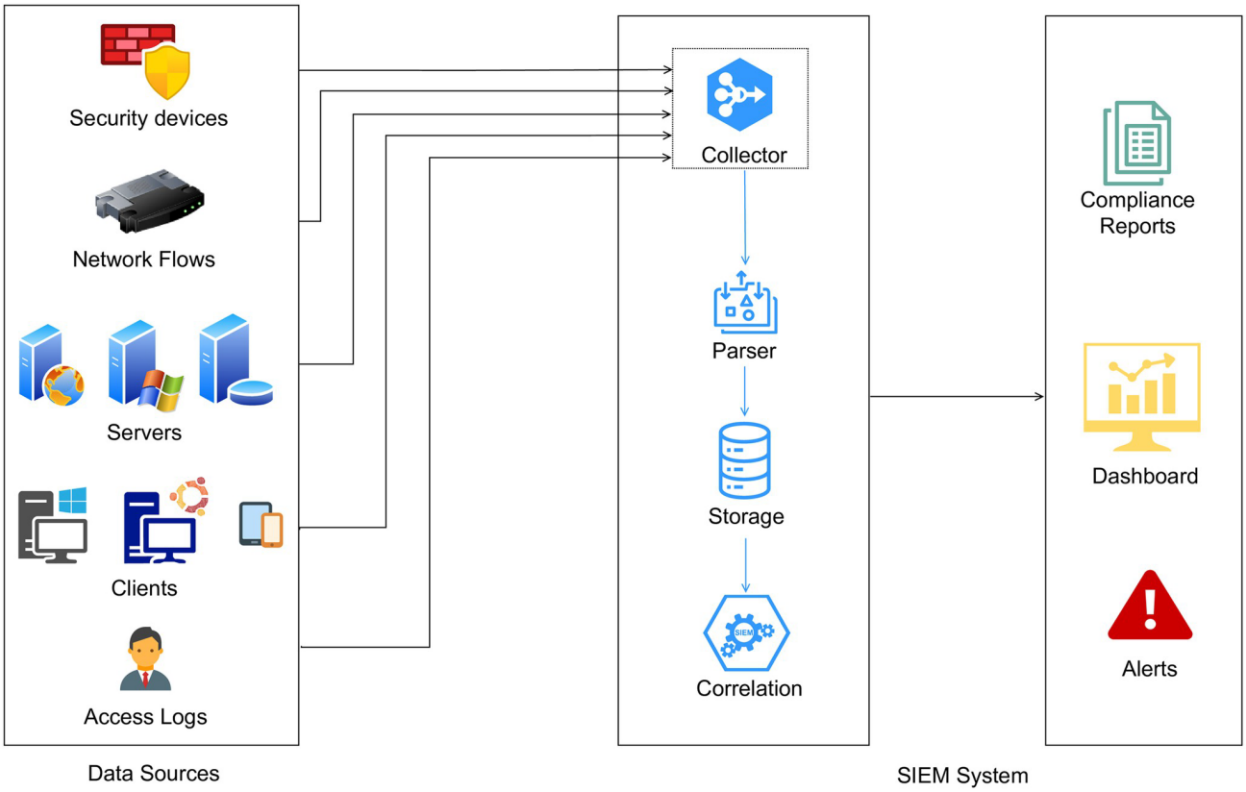



***Figure 1***: SIEM Architecture

### 1.2.2. Data Sources

SIEM solutions have a wider visibility of event logs and different types of data collected from solutions and devices across an organization's network infrastructure. The commonly used entities that serve as data sources for SIEM are network security devices and/or solutions (such as IDSs and firewalls), end-point solutions such as antiviruses, event logs (from sources such as operating systems, storage systems, web servers, different types of applications) and individual devices (such as IoTs, mobile devices, and workstations) [10].

### 1.2.3. Data Parsing and Normalization

As mentioned above, SIEM solutions collect event logs from different sources using either agent-based or agentless collection mechanisms. As a given network is composed of solutions and devices from a variety of vendors, once collected, the data passes through different parsing, aggregation and normalization processes.

Typically, SIEM solutions include built-in parsers for commonly used data sources, and some also support the creation of custom parsers for new sources. The aggregation and normalization

process creates a global event data format which makes the application of correlation and other types of analysis consistent and uniform.

### 1.2.4. Data Storage

Once the data is normalized and changed to a global format, it will be sent to the central database of the SIEM solution. For the normalized data, SIEM solutions usually use structured databases such as MySQL, Oracle DB, Microsoft SQL server or MariaDB. On the other hand for storing the untouched unstructured and semi-structured data for the purpose of archiving and further analysis, SIEM usually use unstructured databases such as MongoDB and Elastic search. Keeping the formatted and unformatted data in databases and storages mentioned above enables effective analysis, report generation and performing complex searches and queries. Especially, big data versions of these data are kept in data warehouse solutions for business intelligence, big data analytics and perform extensive analysis [10].

### 1.2.5. Correlation Engine

The correlation engine is at the heart of SIEM solutions which is responsible for applying the correlation rules on to singular events or aggregates of events from multiple sources so as to identify malicious activities. Correlation analysis can be performed using several methods, including statistical analysis, rule-based reasoning, case-based reasoning, model-based reasoning, Bayesian reasoning, and graph-based techniques.

### 1.2.6. Visualization, Reporting and Alerting

SIEM systems offer a dashboard that enables real-time monitoring of an organization's security status. This dashboard typically presents data through visualizations like charts, graphs, and widgets. Reports can be generated either on a scheduled basis or on demand, supporting compliance, incident analysis, and executive overviews. Additionally, the dashboard displays alerts from the correlation engine when potential security incidents or suspicious activities are detected. Alerts may prompt notifications via email, SMS, or integration with incident response systems and are usually prioritized by severity to streamline response efforts.

## 1.3. Motivation

In today's increasingly interconnected digital landscape, organizations face an ever-growing number of sophisticated cyber threats. Traditional SIEM systems, while instrumental in monitoring and managing security events, often struggle with challenges such as high false positive rates, data silos, and the inability to process a variety of data sources effectively. This leaves critical gaps in security operations, which adversaries can exploit.

The advent of deep learning and advanced data fusion techniques offer unprecedented opportunities to address these limitations. Unlike conventional methods, deep learning can uncover intricate patterns and correlations across diverse data types, enhancing threat detection accuracy. Similarly, multimodal analytics and alert fusion enable the integration of heterogeneous security event data into a unified and actionable framework, thereby streamlining threat analysis and decision-making processes. Despite these advancements, existing SIEM implementations rarely capitalize on these cutting-edge technologies, highlighting a significant gap in both research and practice.

This research aims to bridge that gap by proposing a multimodal SIEM solution empowered by deep learning and alert fusion. The motivation lies in addressing critical challenges such as reducing alert fatigue, improving detection precision, and enabling adaptive responses to sophisticated threats. By incorporating multimodal data processing and leveraging deep learning models for enhanced threat detection and correlation, this study seeks to improve both the efficiency and effectiveness of modern cybersecurity operations.

Beyond its technical implications, this research is motivated by its potential to impact broader academic and professional domains. It will provide a foundational framework for future studies expanding on multimodal analytics as well as alert fusion with the expansion of the log and traffic sources as well as the fusion techniques in cybersecurity and contribute to the development of more robust and scalable SIEM solutions. Furthermore, the insights gained could inform industry best practices, ultimately enhancing organizational resilience against cyber threats.

## 1.4. Statement of the Problem

The majority of cyber security solutions these days, such as SIEM and IDSs, mostly utilize the analytics methods namely rule-based, signature-based or anomaly-based analytics. It is well established that the aforementioned methodologies base their analysis on templates that were crafted on a basis of already known threat characteristics in the effort of identifying potential attacks. As the dynamics of the threat landscape continuously change and as these templates do not incorporate the evolving nature of threats, these techniques are not capable of identifying zero day attacks or for that matter any attack that is not known beforehand. For instance, these systems rely on rule-based techniques which mainly base analytics on pre-defined rules and thresholds for threat detection [3], which in turn are characterized by their inability to encode the real nature of attacks and pure dependency on human expertise in creating and interpreting the rules.

In the comparative analysis made between open source and proprietary SIEM systems, it is described in [11] that proprietary SIEM solutions are costly but are easy to deploy and are in need of less time and expertise. When considering their analytics capabilities, it is implicated that cutting age detection mechanisms are employed to detect intrusions and various threats in a given network.

The proprietary SIEMs considered in [12], which are QRadar, Splunk, Securonix, Exabeam and LogRhythm, have the capabilities of threat intelligence, behavior profiling and user monitoring, where most of the open sources lack these capabilities.

However, whether they are open source SIEM or that of the proprietary ones, detailed system descriptions are not provided. Hence, we cannot determine what kind of technical architecture or the type of algorithms they used.

On the other hand, in the case of researches that are trying to tackle such issues, mostly they implement traditional machine learning algorithms such as SVM, decision tree, linear and logistic regression and random forest.

There are also researches which implement rule-based, anomaly detection, signature based, heuristic-based and behavior based techniques for detecting threats such as malicious, fraudulent, privileged vs non privileged and persistent attacks [12]. But as mentioned above, other than their inability to cope with the changing threat landscape, these techniques suffer

higher false positive rates, inducing additional analysis or verification and creating alert fatigue on analysts.

Hence many researches propose that for next generation SIEMs to become effective and efficient in their detection and response capabilities, they need to incorporate artificial intelligence and ML technologies as their core analysis engines [2].

In another perspective, the rapid changes that are happening in the threat landscape requires for complex and layered threat detection techniques to shift their considerations which is usually limited to single or isolated data streams. Multimodal threat detection has become a powerful trend fusing information from a diversity of sources such as event logs, network traffic and content analysis with the aim of creating situational awareness and uncover hidden malicious patterns. Multimodal threat detection approach offers several advantages, namely: first, each data source provides a unique insight into what is going on in the network. For instance, network traffic reveals anomalous data transfers. While email communications show suspicious contents and links, documents might expose unauthorized access attempts. Consolidating these insights from a multitude of data sources together provides a situational awareness on the whole network and user behaviors thereby enhancing detection accuracy and minimizing false positives [12].

Similarly, as researches suggest, another highly promising direction for improvement lies in the use of multimodal fusion methods that combine various data streams, e.g., network traffic, system logs, and user behaviors, to better understand what possible threats might exist. One can utilize neural networks in order to fuse and exhaustively investigate data from these various streams of data, contributing to higher precision as well as the dependability of threat detection capabilities [3].

Building on the preceding discussions, this research proposes to apply deep learning-based multimodal intrusion detection combined with alert fusion techniques to improve detection accuracy in SIEM. Multimodal fusion technique mitigates the issue of false positives, by which single-modal analysis are characterized. Anomalies in one data stream might be legitimate in another, acting as cross-validation and reducing the number of false alarms while performing investigations [12]. Furthermore, complementing the analysis made by the correlation rule with

that of the deep learning, and vice versa, greatly minimizes the number of false positives as combining the two types of analytics enhance detection accuracy.

## 1.5. Research Questions

The following are the research questions that this research responds to

**RQ1:-** What is the impact of deep learning-driven multi-modal threat detection with additional data sources on a better intrusion detection accuracy in SIEM?

**RQ2:-** What is the impact of alert fusion, in SIEM, with complementing alert outputs in order to minimize false positives?

## 1.6. Research Objective

### 1.6.1. General Objective

The general objective of this research is to propose a deep learning-driven multi-modal SIEM solution that utilizes alert fusion for a better intrusion detection accuracy and minimized false positives.

### 1.6.2. Specific Objective

The specific objectives of this research are:

- To design a multi-modal SIEM solution architecture empowered by deep learning and alert fusion
- To pick a multimodal data environment for experimentation
- To generate intermediary data for further analysis (event logs)
- To implement the deep learning, correlation rule and alert fusion analytics
- To evaluate the results of the experiment with the pre-set evaluation parameters

## 1.7. Contribution of the Study

This research presents a multimodal threat detection that integrates the classical correlation rule-based SIEM with deep learning-based intrusion detection to detect network intrusions. The proposed solution utilizes multimodal threat detection with data sources of security event logs from different security solutions and network traffic in the form of network flows in a given enterprise network.

Hence this research presents the following main findings

1. A multimodal data source threat detection involving event logs and network traffic in SIEM,
2. A deep learning-based intrusion detection for traffic analysis with MLP,
3. A complementary analytics of, event logs from correlation engine and network traffic from deep learning analytics, via alert fusion techniques.

## 1.8. Scope of the Research

### 1.8.1. In-Scope of the Research

The scope of this research is to design and implement a prototype for a SIEM solution for intrusion detection with multimodal data sources and analytics capabilities driven by deep learning and alert fusion techniques.

### 1.8.2. Delimitation of the Research

The real-time implementation of a multimodal SIEM solution empowered by deep learning and alert fusion is not in the scope of this version of the research.

## 1.9. Organization of the Document

This research is organized in the following structure:

**Chapter 1:-** This chapter is organized into the following sections: an introduction that provides background on the general discussion of SIEM, the research motivation, the problem statement, the research questions, and the research objectives, including both general and specific aims. It also presents the study's contributions and defines its scope in the stated order.

**Chapter 2:-** This chapter is primarily structured into the Literature Review and Related Work sections. The Literature Review is further divided into subsections that examine the global cyber threat landscape, the application of deep learning techniques for intrusion detection with a particular focus on multilayer perceptron (MLP), and the various SIEM and Intrusion Detection System (IDS) solutions.

**Chapter 3:-** This chapter outlines the study's methodology, including sections on the research approach, study design—with subsections on the design overview, data overview, and implementation details—the tools employed, the research variables, and the proposed system, which is further divided into the proposed architecture and its functional processes.

**Chapter 4:-** This chapter presents the experiment and analyzes the results, organized into sections on implementation—which includes data generation and experimental setup—evaluation metrics, research findings, and a discussion of the results.

**Chapter 5:-** This chapter is all about summarizing the whole process of the study and its result with the indication of future works.

# Chapter 2

## 2. Literature Review

The literature review part of this research is organized with the following perspectives: the global cyber threat landscape, deep learning for threat detection, Multilayer Perceptron, SIEM and related work.

### 2.1. The Global Cyber Threat Landscape

In many cases, Internet has become the default way of life in this modern era, facilitating the daily functions of businesses and government institutions alike. The aggressive use of digital technologies for a range of critical services underscores the Internet's role as a primary medium for communication, commerce, and administration [4]. Moreover, most organizations use technologies that are networked and connected to external environments through technologies such as virtual private networks (VPNs) and the internet itself, facilitating both internal and external business processes [13].

However, besides these critical benefits of the Internet, being connected to networks interfacing with external others present significant security challenges. The constant presence of cyber attackers, who consistently seek to exploit vulnerabilities within these systems made the concerns over cyber security and safety of users' data even worse than ever [4]. And this is because of the fact that the frequency of network intrusion attempts and successful attacks, including distributed denial-of-service (DDoS) attacks, cross-site scripting, and probing activities are skyrocketing. Information and communication technology (ICT) systems, which frequently handle sensitive user data, are particularly vulnerable to these types of attacks, leading to significant risks of data breaches [8]. Consequently, safeguarding users' information, and the network infrastructure where it resides, has become critically important in addressing the risks associated with the increasing digital dependency that has been and will be created [4].

Even more to it, [14] as the introduction of artificial intelligence (AI) has significantly transformed the field of cybersecurity by offering powerful tools for defensive measures, the same innovations have become to pose new risks, as they too enable cyber attackers to craft extremely complex cyber threats, including AI-enabled attacks. In particular, malicious actors now exploit

AI to design and execute cyber threats, making it difficult for traditional security solutions to detect and respond to these consistently evolving attacks. AI enables rapid adaptation to evolving cyber defense technologies and empowers cybercriminals to target system vulnerabilities with high precision. Furthermore, AI-empowered tools can go to the extent of simulating human behaviors, which creates a conducive platform enhancing the process of crafting phishing-attack attempts and even increasing the probability of successful breaches. Some of these AI enabled tools for cyber-attacks are DeepExploit, Metasploit, Scikit-learn and NMAP [9].

This duality in AI's role within cybersecurity—both as empowering safeguards and potential threats – characterizes the complex landscape that modern cybersecurity professionals must deal with [14].

## 2.2. Deep Learning for Intrusion Detection

Deep learning, a specialized area within machine learning, leverages artificial neural networks to model complex relationships between input and output variables. In the realm of Network Intrusion Detection Systems (NIDS), deep learning approaches provide significant advantages compared to traditional machine learning methods. First, deep learning models possess the ability to learn complex features directly from raw data by themselves, thereby avoiding the necessity for manually selecting those features with high predictive power, a capability that is particularly advantageous in the high-dimensional environments common to NIDS applications. Furthermore, deep learning models demonstrate increased resilience to noise and outliers, a critical asset for NIDS, as they often encounter noisy or incomplete data. Lastly, deep learning facilitates the creation of innovative NIDS methodologies, such as anomaly detection and hybrid NIDS frameworks, expanding the potential for improved network security [15].

The field of Artificial Intelligence (AI) has recently experienced substantial advancements, particularly in the learning capability of deep learning techniques compared to traditional machine learning approaches. These advancements have shown notable impacts within the area of cybersecurity, especially with respect to enhancing the detection accuracy and efficiency of intrusion detection systems. Application of Deep Neural Networks (DNNs) to intrusion detection domain has proven to bring about a great deal of enhancement in the detection accuracy of IDSs with respect to resistance to noisy data and efficiency of their outcomes.

As deep learning models have shown considerable promise in network intrusion detection, they too face several key challenges that need to be addressed for them to produce the desired result. Some of these challenges include the limited availability of labeled data essential for training deep learning algorithms and the interpretability of these models. Collecting labeled data that can be used for training and testing network intrusion detection systems requires specialized expertise and substantial resources, making it a complex and resource-intensive task. Moreover, interpretability is crucial to understanding the decision-making processes of deep learning models, providing insight into the rationale behind detection outcomes and enhancing trust in their applicability [15].

### 2.3. Multi-Layer Perceptron (MLP)

It was Rosay et al. who proposed a multilayer perceptron model for network intrusion detection with the objective of enhancing cybersecurity in Vehicular IoT networks. The approach mainly focuses on training a multilayer perceptron algorithm to demonstrate the capacity and advantages of deep learning techniques for intrusion detection systems. In order to shape the network intrusion detection system, the CIC-IDS2017 dataset was utilized, whereas the CSE-CIC-IDS2018 dataset was utilized for the purpose of validation and testing. At the time of pre-processing, the training dataset was cleaned to address imbalances before being partitioned into three sub-datasets. 50% was allocated for training, 25% was allocated for validation, and 25% was allocated for testing. For the purpose of avoiding those attributes with no prediction capabilities, feature selection was performed on the dataset. The model was developed and trained using Python and the TensorFlow framework. The experimental results indicated that the proposed model outperformed traditional machine learning models in accuracy while maintaining low false positive rates. Furthermore, the model was implemented on a system-on-chip vehicle to enable testing in an embedded environment. However, certain limitations were identified, such as underrepresentation of specific attacks and inadequacies in feature calculations within the datasets. Despite these challenges, the model's adaptability for real-world applications suggests its potential for re-engineering in automotive industry implementations.

The multilayer perceptron (MLP) algorithm was also employed in [6] to develop a misuse detection model. An MLP is a feedforward artificial neural network designed to map known

inputs to accurate outputs through multiple interconnected layers of nodes in a directed graph, where each layer is fully connected to the next. MLP utilizes a supervised learning approach called backpropagation to train the network.

## 2.4. SIEM and Intrusion Detection Systems (IDS)

Recent trends of cyber-attacks have evolved to be increasingly sophisticated and complex, with the aim of exploiting critical ICT resources with advanced methods to access and steal valuable data from these targeted entities [6]. In tackling such attacks, IDSs usually are one category of the solutions that has to deal with problems such as large network traffic volumes, highly uneven data distribution, the difficulty to realize decision boundaries between normal and abnormal behavior, and a need for continuous adaptation to a constantly evolving threat landscape. In general, the challenge is to efficiently capture, understand and classify various attackers' behaviors in a given network [16].

In their research, [4] mentioned that the integration of cybersecurity and threat detection tools has become increasingly important to prevent downtime of critical services and systems. The main cyber security systems which can be seen in such categories SIEM, and IDS. And they have become core parts that happened to be at the center of the effort of monitoring and defending networks and hosts against intrusions.

Unfortunately, the use of these systems is facing challenges with respect to monitoring and detecting attacks, the signature of which not known yet, or attacks that are zero day, and attacks that target specific systems and vulnerabilities. Such attacks can be used to go unnoticed by most organization's defense systems and infiltrate the target network [4]. Most of the times the use of these monitoring and detection systems tend to give false negatives as there is no way of identifying the behaviors of newly emerged attacks due to the limited knowledge base and capability used in detecting the known type of attacks/patterns. Furthermore, as there exists a majorly known vulnerability of a system for each attack, the real effort of these intrusions is to exploit those vulnerabilities of systems and perform that malicious act intends to do. However, most of the current IDSs do not address the vulnerability management area [13].

The 2018 Data Breach Investigations Report reveals that a significant majority—68%—of security breaches went undetected for several months or even longer. This finding underscores a substantial delay in breach detection, despite many breaches occurring within minutes or even seconds [4].

In addition to the point made by [4], the rising complexity and dynamic nature of modern systems and applications, which are closely tied to the expansion of the Internet of Things (IoT), virtualization, and service-oriented architectures, contribute to the increased sophistication of cyberattacks and the challenges associated with their investigation. In light of these advancements, traditional defense mechanisms, such as firewall-based architectures and isolated Intrusion Detection Systems (IDSs), have become inadequate for safeguarding IT infrastructures. Instead, a contemporary security approach necessitates the integration and cooperation of multiple security devices, placing particular emphasis on the role of IDSs to enhance overall protection [5].

Traditional SIEM solutions are widely employed in the network intrusion detection domain and have proved themselves in real-time threat detection in the cases where the attacks'templates or patterns were identified earlier and made into correlation rules. Despite these strengths, SIEM systems struggle to reliably distinguish between benign and malicious network traffic, largely due to the dynamic nature of Distributed Denial-of-Service (DDoS) attacks. These evolving threats make it difficult to establish fixed rules that are capable of accurately capturing their signatures. To address this issue, Artificial Intelligence (AI), especially AI-driven intrusion detection methods, can offer a viable solution by addressing some limitations inherent in rule-based SIEM systems. AI approaches, particularly those employing Machine Learning (ML) algorithms, are designed to detect anomalies by identifying deviations from typical network behaviors, thus potentially enhancing the accuracy of DDoS detection. However, AI techniques alone may produce both false positives and false negatives, given the limited contextual information and the complexity of network traffic patterns [1].

The other limitation with the current SIEM solutions is their tendency to produce an overwhelming volume of alerts, often accompanied by a high false positive rate. The sheer

quantity of daily alerts, which can reach hundreds of thousands, exceeds the capacity of Security Operations Center (SOC) analysts to thoroughly investigate each one. As a result, SOC analysts may prioritize only those alerts deemed to be of high severity or opt to suppress recurring alerts of a similar nature. This approach, while pragmatic, carries the risk of overlooking potentially severe cyberattacks [17].

Consequently, researchers and cybersecurity professionals strive to enhance SIEM and Intrusion Detection Systems (IDS) by integrating intelligence, adaptive mechanisms, and pattern recognition capabilities. Specifically, they employ machine learning (ML) models to boost the efficiency and accuracy of these systems, leveraging historical data to refine model performance. However, ML models present significant challenges due to their intensive computational requirements, both for training and calibration, which can complicate their practical deployment in resource-constrained environments [4]. Furthermore, behavior-based intrusion detection systems (IDS) aim to identify potentially harmful network traffic and operations; however, they often trigger alerts for activities that are, in fact, legitimate. This limitation frequently results in elevated false positive rates, undermining the reliability of these systems for security analysts. High false positive rates compel analysts to perform extensive manual investigations, including detailed log analysis, to verify IDS alerts, effectively duplicating the efforts initially intended to be automated by the IDS. This manual review process not only burdens analysts but also reduces the efficiency of security operations [13].

## 2.5. Related Work

Since the Massachusetts Institute of Technology's Lincoln Laboratory initiated the DARPA Intrusion Detection Evaluation in 1998 and 1999, extensive global efforts have been dedicated to advancing intrusion detection system (IDS) technology. This initiative catalyzed numerous research projects aimed at improving the detection of previously unknown attacks and addressing the challenge of reducing false-positives. Consequently, a diverse range of research studies has been conducted to develop more accurate and effective IDS solutions [5].

Model ensembling is used in [4] which is a very powerful technique to increase accuracy of models on a variety of machine learning tasks. Model ensembling consists of several base models

combined together in order to produce one optimal model that will best predict the desired outcome. Some ensembling methods require the base models to be pre-trained to only combine their predictions on test set; other methods require multiple rounds of training on different chunks of the training set [4]. For the researchers to consider machine learning techniques in the intrusion detection problem makes the detection more dynamic. But here the idea of creating a single optimal model that predicts every outcome is impractical as the working contexts and scenarios differ from one case to another.

On the other hand, [13] argue that IDS systems tend to operate independently as either Host Intrusion Detection Systems or Network Intrusion Detection Systems. Due to the sophisticated and complex nature of cyber-attacks, this separation ultimately reduces their ability to detect threats effectively. Hence, they proposed that merging the two capabilities into one system makes the output more effective. In other words, the proposed system attempts to correlate the alerts received from both HIDS and NIDS. Here the idea of the researchers is to correlate events from more than one security solution makes the output of the analysis stronger. But, as described in the different sections of this paper, just depending on one or two security solutions to detect intrusions does not cope with the ever increasing complexity of cyber-attacks, which needs the consolidation of every relevant security solution in the network.

A machine learning solution was proposed in [5] based upon two major concepts: (1) event fusion into meta-events: collecting, normalizing and fusing together events that are likely to be part of the same attack; (2) classification of meta-events: based upon the attributes of a meta-event, decide whether it represents an attack or a false alarm, using different machine learning techniques. Even though the ideas of event fusion and correlation represents a dynamic and a better analytics capabilities here, the data source that the correlation is considering comes only from alerts generated by IDSs. This limits the comprehensiveness of the analytics by not considering other attack indicators and threat intelligence sources such as events from other sources and the perspective from raw traffic data.

Using wireshark and directly selected machine learning algorithms to determine the presence of attacks was also proposed in [16]. Their research tries to analyze the network traffic and monitor

the flow via wireshark to get the log analysis done. Once the traffic packets' protocols are determined and the log analysis is done, the result is sent for feature selection. Then they have listed a set of machine learning algorithms (SVM, Random Forest, Logistic Regression, Naïve Bayes, Decision Tree, KNN) to be applied on the selected features. Then the accuracy is measured based on the detections. The effort to combine both network traffic and event logs makes the analysis stronger and the output more comprehensive.

A multilayer perceptron (MLP) algorithm was employed in [6] to develop a misuse detection model. To address the high rate of false alarms in anomaly detection, reinforcement learning is applied. In this approach, the network is trained to make decisions and predict potential threats based on reinforcement signals received from the environment. These signals are sent to the module where the weights for the decision-making ability of each agent and the trust levels in individual agent are determined. But the MLP is implemented for just anomaly detection and in combination with KNN and reinforcement learning.

It is also suggested that any machine learning algorithms can be effectively used to train systems to detect potential cyberattacks in [18]. Upon detecting an attack, an email notification can automatically be sent to security engineers or relevant users. Any classification algorithm may serve to identify whether the attack is a DoS/DDoS incident. One example is the Support Vector Machine (SVM), a supervised learning method capable of analyzing data and recognizing patterns. Since it is impossible to fully control the timing, location, or form of an attack, absolute prevention remains elusive. Early detection, however, offers the best current defense, helping to reduce the risk of severe damage from such incidents. Organizations can either employ existing detection solutions or develop their own systems to identify cyberattacks at early stages, minimizing their impact. Ideally, these systems require minimal human intervention.

Most of the studies we reviewed here primarily concentrate on analyzing security event logs, though a few have ventured into examining real-time network traffic as well. To improve the accuracy of intrusion detection methods and reduce the occurrence of false positives, it is crucial to implement multimodal sources of data. This includes event logs generated by security devices,

real-time traffic data, and insights produced by other security systems. By fusing these sources, a more comprehensive view of network activity can be achieved.

Furthermore, when analyzing network traffic, it is critical to structure it in such a way that organizes the packets into groups of data with network flows based on specific time intervals. This time frame-based approach of creating network flows facilitates better pattern recognition and aids in identifying potential security threats as they develop. Additionally, current researches largely emphasize on individual and siloed analytics processes, each following its own unique path. Rather than maintaining these separate channels, there is a need for a complementary and consolidating analytics that extends beyond merely visualizing alerts on a SIEM dashboard. Such approach of multimodal data sources and alert fusion would enhance the capability of SIEM to make alert consolidation, duplicate removal and interpret information based on the situational awareness made based on sources across different security solutions, providing a more robust defense against cyber threats.

# Chapter 3

# 3. Methodology

## 3.1. Research Methodology

In this study, a combination of experimental and machine learning research methodologies are utilized in order to contribute to the field of intrusion detection. Through the integration of these approaches, the objective is to improve the detection accuracy of the implementation with minimized false positive rates.

## 3.2. Study Design

**General Overview**

This research is designed in such a way to detect intrusions at three different layers of analysis configured together in a single solution that involves traffic analysis with an intrusion detection system and a deep learning implementation, correlation with in SIEM's correlation engine and finally a late-alert fusion at the end. This configuration is aimed at bringing about a solution which has a better detection accuracy with minimized false positive rate. Hence a study environment, where a multimodal threat detection mechanism is mimicked, was made ready involving an already captured pcap file, a transformed and labeled network flow-version of the pcap file, an open source-based intrusion detection system, an open source-based SIEM with a deep learning implementation for IDS and finally an alert fusion component within the SIEM that ingest alerts from a signature-based open source IDS, correlated alerts from the SIEM's correlation engine and finally analytics output from the deep learning implementation of IDS, as mentioned in the proposed system.

## 3.3. Data Overview

The data collection or generation process is aimed at producing a representative data for the scenario designed in the architecture, which is deep learning-driven intrusion detection with SIEM in a given generic network. This data generation should be in a way where a raw traffic or pcap file is the source for all the intermediate analysis done by the deep learning implementation, the IDS, SIEM as well as the alert fusion component. Hence, it should be in such a way that the IDS consumes the raw traffic and produces alerts where the same raw traffic would be

preprocessed in a network flow packets that will be input to the deep learning implementation of the SIEM component. On the other hand, the event logs generated by the traditional IDS will be consumed by the SIEM's correlation engine to produce correlated alerts. And finally, the alerts produced by the SIEM's deep learning implementation, the correlation engine and the traditional IDS will be fused to produce a refined alert where false positives and alerts in general are minimized.

Hence, to meet the above scenario, an intrusion detection evaluation dataset (CIC-IDS2017) is used from Canadian Institute for Cyber Security (CIC) and University of Brunswick (UNB). From this repository, a network traffic data (pcap) of 5 days, Monday to Friday, was taken as a base for producing the other derivative data. From the same repository, a transformed network-flow version of the raw pcap file with attack labels was used. One of the data preprocessing tasks, which was producing the network flow, was already done since it was acquired from the same repository.

As mentioned above, the intrusion detection evaluation dataset (CIC-IDS2017) which was used as a basis for this research has the following distribution

According to https://www.unb.ca/cic/datasets/ids-2017.html, "the CICIDS2017 dataset contains benign and the most up-to-date common attacks, which resembles the true real-world data (PCAPs). It also includes the results of the network traffic analysis using CICFlowMeter with labeled flows based on the time stamp, source, and destination IPs, source and destination ports, protocols and attack".

This environment was setup based on the following IPs and their specific roles as victim and attacker machines:

***Table 1:-*** Victim and attacker networks information

| No. | Server/Machine/Network Device | IP | Role |
|---|---|---|---|
| 1 | Firewall | 205.174.165.80, 172.16.0.1 | - |
| 2 | DNS + DC Server | 192.168.10.3 | - |
| 3 | Kali | 205.174.165.73 | *Attacker Network* |
| 4 | Windows | 205.174.165.69, 70, 71 | *Attacker Network* |
| 5 | Web Server | 192.168.10.50, 205.174.165.68 | *Victim Network* |
| 6 | Ubuntu Server 12 Public | 192.168.10.51, 205.174.165.66 | *Victim Network* |
| 7 | Ubuntu 14.4 (32B & 64B) | 192.168.10.19, 17 | *Victim Network* |
| 8 | Ubuntu 16.4 (32B & 64B) | 192.168.10.16, 12 | *Victim Network* |
| 9 | Win 7 Pro, 64B | 192.168.10.9 | *Victim Network* |
| 10 | Win 8.1, 64B | 192.168.10.5 | *Victim Network* |
| 11 | Win Vista, 64B | 192.168.10.8 | *Victim Network* |
| 12 | Win 10 Pro, 32B | 192.168.10.14 | *Victim Network* |
| 13 | Win 10, 64B | 192.168.10.15 | *Victim Network* |
| 14 | MAC | 192.168.10.25 | *Victim Network* |

Hence the analysis and interpretation of the alerts produced later will be evaluated based on the perspectives set in this table.

An example of the attack simulation can be represented as

- ***Attack***: 205.174.165.73 -> 205.174.165.80 (Valid IP of the Firewall) -> 172.16.0.1 -> 192.168.10.50
- ***Reply***: Reply: 192.168.10.50 -> 172.16.0.1 -> 205.174.165.80 -> 205.174.165.73

### 3.4. Tools Utilized

Google Collab is utilized for implementing the data preprocessing and deep learning implementation of the intrusion detection part of the SIEM. On the other hand, Anaconda Navigator (Spyder) is utilized for writing the python codes for implementing the alert fusion component of SIEM.

## 3.5. Research Variables

Based on the research questions set, the different variables (independent, dependent, control, intervening and extraneous) set to be observed in the experiment phase are described as follows.

### Independent Variables

The first independent variables identified for this experiment are the types of deep learning models in the IDS implementation as well as the correlation methods used in the SIEM system. As the newly implemented component of the SIEM utilizes a deep learning algorithm, the selection and implementation of the algorithm governs the detection accuracy. Furthermore, as the correlation method implemented in the open source SIEM is rule-based, it also determines how accurate and efficient the event logs are analyzed. The second independent variable is the training data size and its diversity. The more and diverse training data used the more accurate the model turns out to be. For that, a 5 days long training data with about 14 types of attacks and with more than 3 million records is used for training the model. The third independent variable is the types of additional data sources used. As the raw pcap file is the source of any derivatives used in the proposed system, it has a great deal of impact on the analysis made. The other independent variable identified is data preprocessing techniques utilized. As the data is already cooked for research in such a way that it passed so many preprocessing phases, in this research, managing missing values, if any, and intermediary transformations for the model building will be implemented. The final and the fifth independent variable is the alert fusion modality which might either early or late fusion. Since there are no sensors and each analytic path has its own way of producing alerts, the late-fusion method is implemented, as described in the study design above.

### Dependent Variables

Three dependent variables are identified here, which are accuracy of intrusion detection, rate of false positives and efficiency metrics with regard to processing time and resource utilization. As described above, the intrusion detection accuracy and the rate of false positives of this implementation are dependent on the types of algorithms implemented for detection. Furthermore, the processing time and resource utilization too depend on the number of layers of the deep learning algorithms used and how big the training data is.

### Control Variables

As control variables, time frame of data collection, monitored environment and system configuration are identified. The original data is collected across the five days of the week, Monday through Friday, which was used for all the components, namely IDS (Suricata), deep learning implementation of intrusion detection on SIEM and correlation rule of the SIEM system. Hence all types of analysis done are on the same data with different derivatives. The same is true for the monitored environment, which is revealed through the collected data. On the other hand, all the experiments are done, from start to end, under the same system configuration.

### Intervening Variables

As intervening variables, quality of the training data, configuration of the selected SIEM, quality and relevance of additional data source and the different configuration settings can be considered. Even if well performing algorithms are implemented, the quality of the training data greatly impacts the intrusion detection accuracy. For that, a well transformed and prepared data set is used. The configuration of the selected SIEM and the associated correlation rules determine the creation and accuracy of the alerts by the SIEM system.

And finally, quality and relevance of the additional data stream used, which is the raw pcap data, also determines the accuracy and efficiency of the detection algorithms and techniques. In this case, a well cooked and prepared IDS evaluation dataset is used from CIC. Hence the expectation is that instead of intervening in the experiment in a negative way, it will enhance the performance of the algorithm. With regard to relevance, the raw pcap data serves as the source for all derivative data namely event logs, IDS alerts and network flow.

### Extraneous Variables

As extraneous variables, network environments, variability in the types of data being fused and human factors are identified. Network environment, especially the type of the network traffic used for the experiment, greatly impacts the analysis with respect to the attack-benign ratio and the representativeness of the captured data. On the other hand, since the alerts to be fused were generated via different analysis processes, they are represented by different data types, which in turn might create challenge to the alert fusion algorithm. This issue will be managed with data

conversion and data preprocessing techniques. And finally the human factor in relation with converting the analysis experience into correlation rules for, such as the SIEM, might be an issue if it is not rich enough.

## 3.6. Proposed System

Based on the problem statement, the literature review and the required result, this research is proposing a multimodal threat detection solution with layers of analysis containing deep learning-based intrusion detection, correlation analysis and alert fusion.

### 3.6.1. Proposed System Architecture

The newly proposed system architecture incorporates the classical SIEM architecture as one of the paths of the multimodal threat detection. In addition to the classical SIEM components, the proposed system has the following capabilities and features:

**Data Source**

There are two kinds of data sources for all types of analytics involved in the proposed architecture: the original network traffic (pcap) and intermediary data.

Network Traffic:- This is the source that directly captured by the intrusion detection system to produce alerts based-on the traditional signature based detection.

Intermediary Sources**:** these data sources are mainly those intermediary derivatives of the raw traffic, such as network flow, event logs and alerts from security solutions such as intrusion detection systems.

**Pre-processing**:- This is the component responsible for pre-processing the network traffic in such a way that makes the traffic suitable for analysis by the deep learning algorithm. Here pre-processing tasks such as data cleaning will be done.

**Deep Learning Component**:- This is one of the core component of the analytics in SIEM where it implements multilayer perceptron, a neural network algorithm for threat detection.

**Storage**:- this is responsible for storing the raw traffic, the pre-processed traffic data and finally the produced alerts from the machine learning analytics.

**Alert Fusion**:- This component is responsible for fusing the alerts produced from both the correlation engine and the deep learning analytics using a majority voting algorithm.

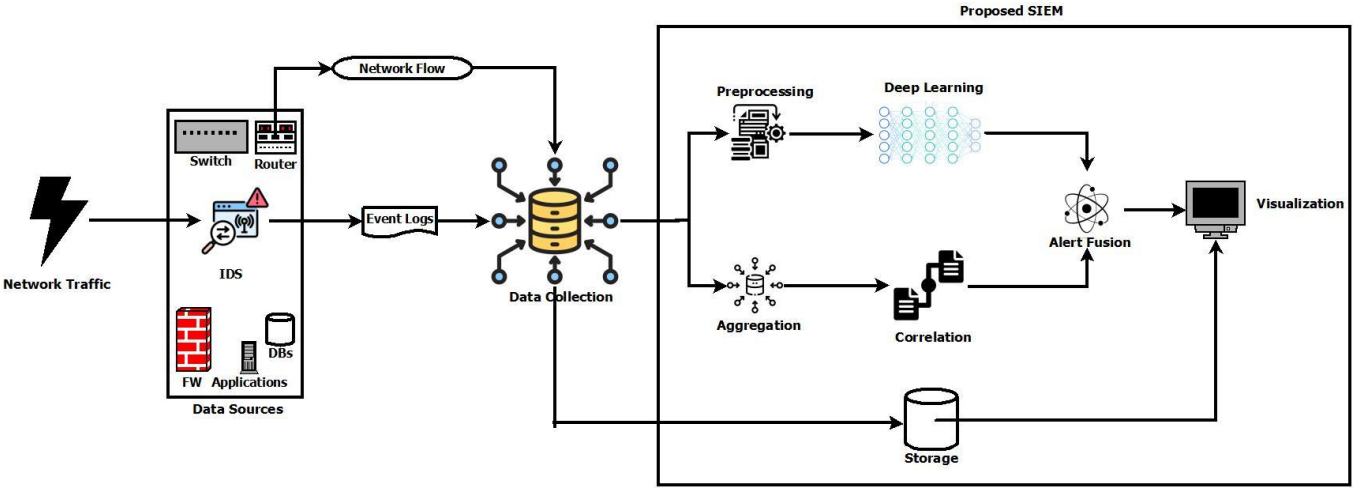



***Figure 2***: Proposed Solution Architecture

### 3.6.2. How does the Proposed Architecture Functions?

The proposed architecture has three streams of analysis:

- the correlation rule engine analysing the event logs and
- the network traffic path collecting, pre-processing and analysing network traffic using deep learning techniques, and
- the alert fusion component which serves as a melting pot for all the other alerts (traditional IDS, deep learning and event correlation)

For the correlation engine analysis an open source SIEM was selected and implemented. This open source SIEM is configured on an environment where it collects, normalizes and analyses event logs generated by an IDS that captured and analysed the network traffic in a simulated environment.

The architecture is going to function as follows:

**Data Generation**

The data is generated in the following scenario:

There are already classified traffic data based on the days of the week, the types of attacks they represent and the security solution with which they were captured. And this data is categorised as benign and malicious.

The plan is to set up an IDS and to let the IDS analyse the-already classified raw traffic network and produce security event logs.

## Collection, Pre-processing and Aggregation

### Collection

Once the traffic is sent to the IDS and the logs are generated, a collection agent for the IDS that was written for collecting IDS logs will read and send the logs to the SIEM.

The network flow is generated from the raw traffic at the router and is directly configured to be sent to the deep learning component of the SIEM, where it will be ready for pre-processing.

### Pre-processing

The network flow, after being generated from the raw traffic, is sent to the Deep Learning components where it passes through a variety of pre-processing steps including merging the data from each week days into a single data frame, missing values management and describing the data with different statistical parameters.

When checking the data for missing values was performed, no data element was found to be missing, as the data was prepared in a simulated network.

### Aggregation

At the SIEM, event aggregation is done if there are other formats of event logs from different event log sources. This happens due to the fact that devices and solutions from different vendors produce event logs with their own formats. For the correlation engine to make analysis on these collected event logs, there need to be a common or global format. This global format is created here at the aggregation stage.

On the other hand the network flow data will also be collected and sent to the pre-processing component, where features are extracted with higher predictive power.

The events will be collected and aggregated to create a global format of the correlation engine and the network traffic will be pre-processed to extract the features for analysis and prediction.

## Analysis

This component has three major modules/functions:

- **Correlation**

  Here at the correlation engine, predefined rules will be written for capturing different types of attacks in the data to further analyse the event logs.

- **Deep Learning**

  Here the network flow with already labelled attack classes will be fed to the deep learning algorithm called multilayer perceptron (MLP) for building a model.

  Based on the accuracy expectation of the algorithm, it will be used for further classification of other flows.

- **Alert Fusion**

  The third function is that the alerts produced from both analytics will be sent to this fusion component separately.

  According to [12], recent advancements in fusion techniques demonstrate potential in optimizing the integration of multimodal data for complex analysis. Broadly, fusion methodologies can be categorized into early, mid-level, and late fusion strategies, each with distinct approaches and applications:

  **Early Fusion:** In early fusion, raw data from multiple modalities is combined at the initial stage, prior to any feature extraction. This approach relies on sensor fusion algorithms or probabilistic models to synthesize disparate data sources, aiming to leverage the inherent relationships among different data streams.

**Mid-Level Fusion:** Alternatively, mid-level fusion occurs after features have been extracted from each individual modality. This method focuses on consolidating these features through techniques such as feature selection and dimensionality reduction, which are employed to enhance analytical efficiency and minimize computational complexity.

**Late Fusion:** Finally, late fusion involves performing independent analyses on each modality before integrating the resulting decisions. Approaches commonly used in late fusion include weighted voting and rule-based systems, which combine the distinct outputs to form a comprehensive decision framework.

Here we use late fusion approach, since we implement the alert fusion component in the architecture where all the analysis are made and isolated decisions have been reached by the siloed analytics stream.

For the alert fusion, we implement majority voting algorithm, which is one of the weighted voting techniques. Majority voting is discussed in detail in the next parts of this study.

- **Visualization**

  Here the fused alerts from both modalities via late fusion techniques will be displayed to the analysts for taking actions.

# Chapter 4

## 4. Experiment and Result Analysis

This section presents the implementation of the multimodal threat detection with alert fusion in SIEM with its experimental evaluation.

### 4.1. Implementation

The experiment is set up in 5 parallel and sequential blocks of the proposed architecture. These five blocks are the data preprocessing block, the classic IDS block, the deep learning IDS block, the correlation block and the alert fusion block.

**The Experiment Setup**

The experiment, the data and analysis flow is setup in the manner of ***Figure 3*** where the source of data for all intermediary processes is that of the raw traffic (pcap).

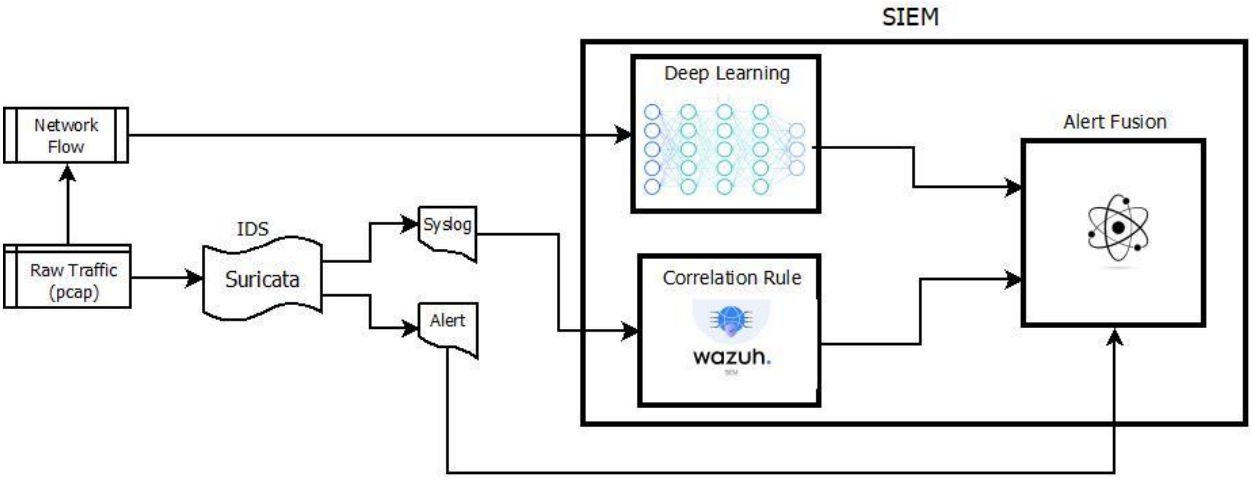



***Figure 3:-*** The Experiment Setup

As described above in the Methodology chapter, two open source tools are used in the experiment. The first one is an open source IDS called Suricata, which is a network intrusion detection tool. The second one is an open source SIEM solution called Wazuh, which has the basic features of classical SIEM solution.

Hence the study was designed to input the raw traffic pcap file to the IDS (Suricata) with the aim of producing alerts and event logs that will be used as inputs for the correlation engine at the open source SIEM, Wazuh and the alert fusion.

Then Wazuh will collect the event logs, aggregate/normalize them and feed the output to the correlation engine where they will be correlated to produce alerts based on the contexts set in the default correlation rules.

On the other hand, the transformed network flow data from the original pcap file will directly be input to the deep learning implementation to build a model for intrusion detection on the original network traffic. Then this model is used to predict network flow which is without labels. Here also alerts will be generated with a possibility of detecting new and unknown attacks.

Then finally, the alert fusion component will ingest alerts from the correlation engine, the deep learning implementation of IDS and from that of the traditional IDS (Suricata) to perform alert fusion, and produces the final alert sets to be visualized to the analyst.

**Correlation**

For the correlation engine rules were written based on the attack types depicted in the original dataset. One example of correlation rule that was used in this experiment is shown in ***Appendix A 1.1***.

This rule is written to fire DoS attack alerts if the conditions set in the different parameters of the correlation rule are met.

**Deep Learning**

For the deep learning, multilayer perceptron (MLP) was implemented. A Multilayer Perceptron (MLP) is a type of artificial neural network that consists of multiple layers of interconnected nodes, or neurons. It's a fundamental building block in deep learning and is used for various tasks like classification, regression, and pattern recognition.

An MLP typically consists of three main types of layers:

**Input Layer:**

Receives input data, where each input node corresponds to a feature of the input data.

**Hidden Layers:**

Process the input data through a series of transformations, where each hidden layer consists of multiple neurons, each with its own set of weights and biases.

These layers extract features from the input data and learn complex patterns.

**Output Layer:**

Produces the final output of the network, where the number of nodes in the output layer depends on the specific task.

For classification tasks, as in our case, the output layer might have nodes representing different classes.

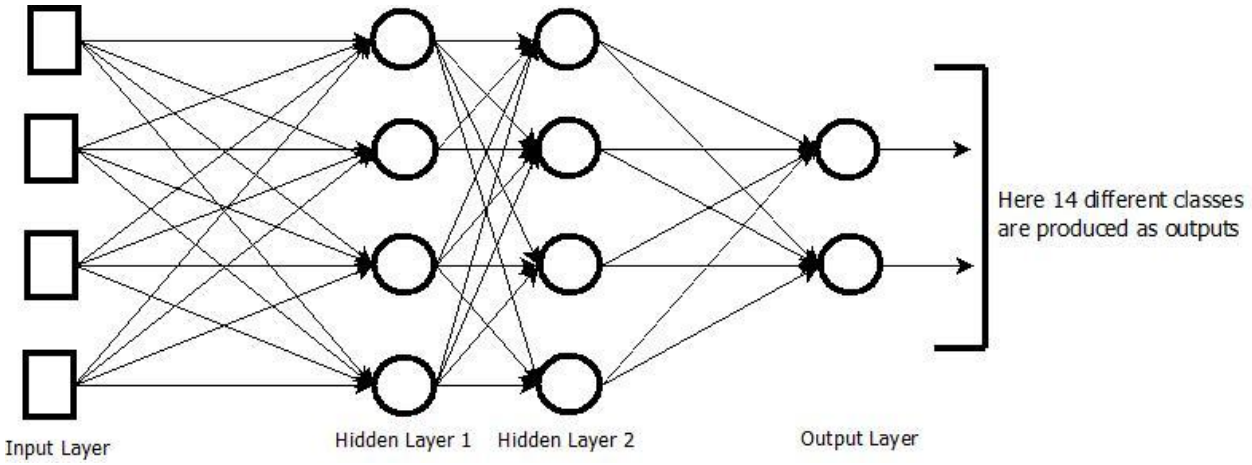



***Figure 4***: Multilayer Perceptron Implementation

Multilayer perceptron algorithm works as follows:

At first, the input data is fed into the input layer. Then through Forward Propagation, it is passed through the hidden layers, one layer at a time, in our case 2 hidden layers. Each neuron in a hidden layer performs a weighted sum of its inputs and applies an activation function to introduce non-linearity. The output of the final hidden layer is fed to the output layer.

At the output layer, the final output of the network is produced. If the output is incorrect, the error is calculated. The error is propagated backward through the network, adjusting the weights and biases of each neuron to minimize the error.

***The Activation Functions***: introduce non-linearity, enabling the network to learn complex patterns. Common activation functions include ReLU, sigmoid, and tanh.

***Weight and Bias***: Each connection between neurons has a weight, and each neuron has a bias. These parameters are adjusted during training to minimize the error.

***Training***: The network is trained on a large dataset of input-output pairs. The weights and biases are adjusted iteratively using optimization algorithms like gradient descent.

**Alert Fusion**

As mentioned earlier, for the alert fusion, a voting technique called majority voting algorithm is employed.

***Majority Voting***

Majority voting alert fusion technique basically assigns voting power to each alert source which will be weighted based either on confidence levels, historical accuracy, the reliability of the source or could easily be weighted equally.

This technique is effective in alert fusion because it ensures that only those alerts corroborated by multiple sources are acted upon, enhancing the reliability and accuracy of threat detection. Furthermore, by requiring agreement among alert sources, this approach minimizes false positives.

But in this case of alert fusion, the alerts should pass through processes such as alert normalization and label formatting for the purpose of creating a common format for alerts from different sources. Hence, all the alerts are normalized to the format

*Source IP, Source Port, Destination IP, Destination Port, Protocol, Label*

where the label is the attack class generated.

## 4.2. Evaluation Metrics

For evaluating the performance of the analysis components of the implementation, we will use the usual evaluation metrics described below:

- **Accuracy**: is the proportion of correctly classified samples among all samples, which implies those correctly classified as normal or attack traffics based on the dataset. And it is calculated as:

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

- **Precision**: Measures the ratio of true positive predictions to the total predicted positives, implying the model's effectiveness in minimizing false alarms by identifying genuine attacks. And is calculated as:

$$Precision = \frac{TP}{TP + FP}$$

- **Recall (Sensitivity)**: Measures the ratio of true positive predictions to all actual positives, providing insight into the model's ability to capture all relevant instances, and it is calculated as

$$Recall = \frac{TP}{TP + FN}$$

- **F1 Score**: The harmonic mean of precision and recall, providing a balanced measure that is particularly useful when classes are imbalanced, and is calculated as:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Where TP is true positive instances where the model correctly predicts the positive class, whereas TN is true negative instances where the model correctly predicts the negative class. FP is false positive instances where the model incorrectly predicts the positive class as attacks, whereas FN is false negative instances where the model incorrectly predicts the negative class or attack traffics misclassified as BENIGN.

## 4.3. Result

The first of the 5 blocks of the experiment setup is where the network flow data is uploaded, examined and preprocessed in preparation for the deep learning implementation of the intrusion detection. This network flow data is of the 5 weekdays of the week where the data is collected in terms of day of the week (Monday through Friday), part of the day (Morning/Afternoon) and the type of attack it represents as shown below.

***Table 2:-*** Description of the Dataset

| No. | Description | Weekday | Status | Count |
|---|---|---|---|---|
| 1 | Monday-WorkingHours.pcap_ISCX | Monday | BENIGN | 529,918 |
| 2 | Tuesday-WorkingHours.pcap_ISCX | Tuesday | BENIGN+Attack | 445,909 |
| 3 | Wednesday-workingHours.pcap_ISCX | Wednsday | BENIGN+Attack | 692,703 |
| 4 | Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX | Thursday/ Morning | BENIGN+Attack | 458,968 |
| 5 | Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX | Thursday/ Afternoon | BENIGN+Attack | 288,602 |
| 6 | Friday-WorkingHours-Morning.pcap_ISCX | Friday/ Morning | BENIGN+Attack | 191,033 |
| 7 | Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX | Friday/ Afternoon | BENIGN+Attack | 286,467 |
| 8 | Friday-WorkingHours-Afternoon-DDos.pcap_ISCX | Friday/ Afternoon | BENIGN+Attack | 225,745 |

The data was also described with respect to different statistical metrics such as count, mean, standard deviation and others as shown in ***Appendix A 1.2*** and ***Appendix A 1.3***.

Hence for training the model, a network flow of 3,119,345 rows was used with 85 columns as shown in ***Appendix A 1.4***.

Of these 3,119,345 labeled network flows, there are flows that are labeled as one of the 14 attacks identified below and there are flows that are classified as benign too. The distribution of the 14 attacks and the benign flow is as shown below in *Figure 8*.

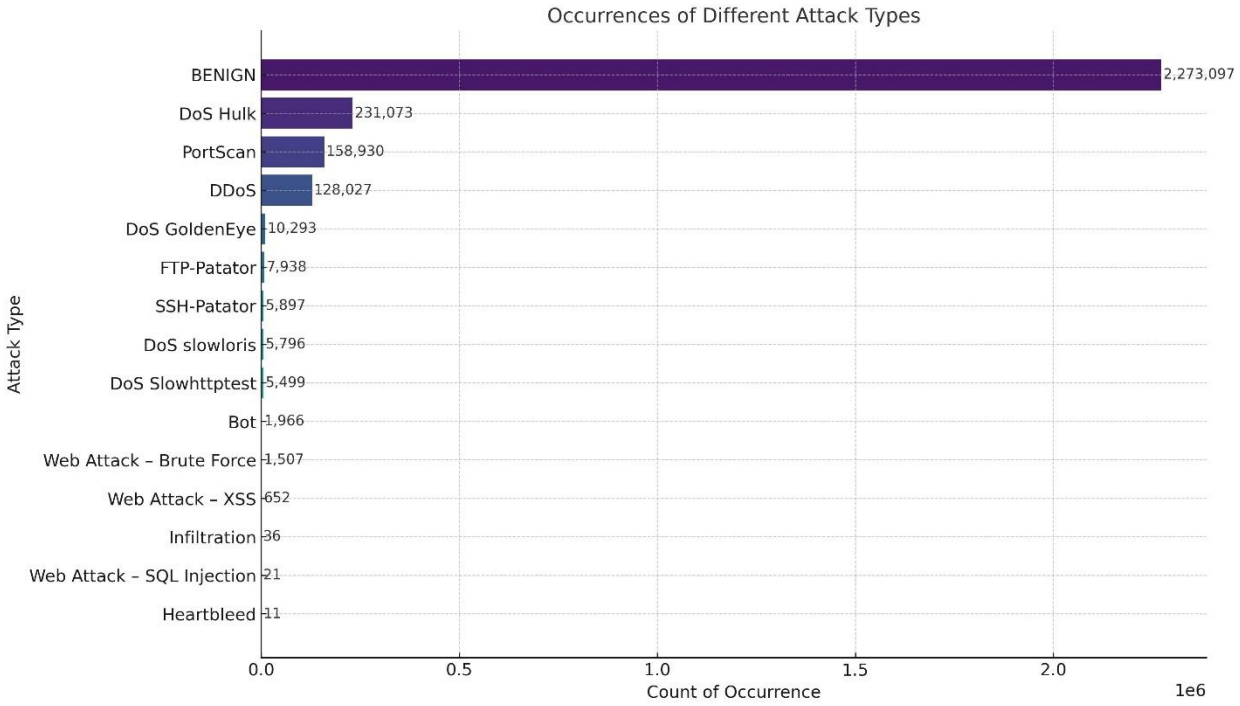



***Figure 8***:- Distribution of Benign and Attack Data

The next task to do in this block is to do preprocessing with respect to the training set if there are any missing values. If there are any missing values, the response will be based on the type and amount of the missing values.

But as the sample is shown in ***Appendix A 1.5*** above, there are no missing values in all columns, as the data was well prepared for IDS evaluation.

The second block of the proposed architecture is building the deep learning model using the preprocessed data. Here the selected deep learning algorithm is multilayer perceptron.

For this experiment, on the deep learning intrusion detection component of the SIEM, a multilayer perceptron (MLP) neural network is used for classifying the CIC-IDS2017 dataset. The model was trained using 80% of the dataset, with a final test accuracy of 94.2% (0.942).

The confusion matrix of the deep learning MLP model is

***Table 3***: Confusion Matrix of MLP implementation

| | **Predicted: Intrusion** | **Predicted: Normal** |
|---|---|---|
| **Actual: Intrusion** | 1,409,705 | 149,728 |
| **Actual: Normal** | 31,194 | 1,528,479 |

Hence,

Total Correct Predictions = 1,409,705 + 1,528,479 = 2,932,184

Total Incorrect Predictions = 31,194 + 149,728 = 187,161

The Total Samples = 2,932,184 + 187,161 = 3,119,345

80% of the total dataset = 2495476 (training set)

20% of the total dataset = 623869 (test set)

***Table 4***: Evaluation Metrics for MLP implementation

| Model | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| MLP | 0.9419 | 0.9108 | 0.97999 | 0.9444 |

On the other hand, the third block with the classic IDS (Suricata), which implements signature-based and anomaly-based detection methodology, was fed more than 48GB of a raw pcap traffic that has produced more than 31GB of event logs that will be directly fed to the SIEM (Wazuh) for event correlation.

In the third block, where the event logs generated from the IDS (Suricata) which is more than 31GB in size, is directly fed to the Wazuh (SIEM) correlation engine for further analysis. Before the analysis, a set of sample correlation rules were written for the correlation engine targeting 10 of the 14 attacks in the dataset, in alignment with the rule format of the open source SIEM (Wazuh). Based on the rules, the following are the alerts generated:

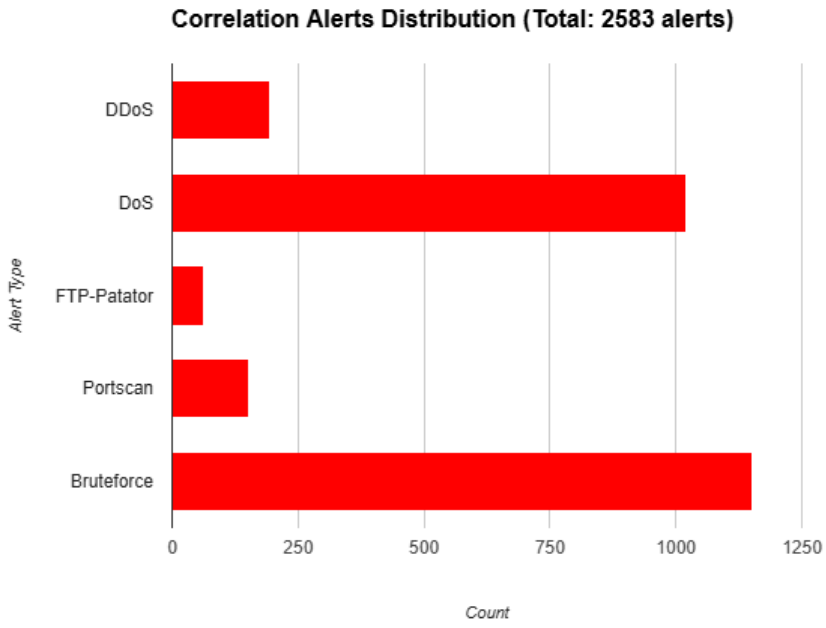



***Figure 10:*** Correlation alert distribution

The last block of the proposed architecture is the one with alert fusion, where it implements the majority voting technique for consolidating the alerts from the deep learning, IDS and the correlation engine.

According to the majority voting applied on the alerts coming from the deep learning implementation, the IDS and the correlation engine, ***Figure 11*** shows the fused alerts with their occurrence:

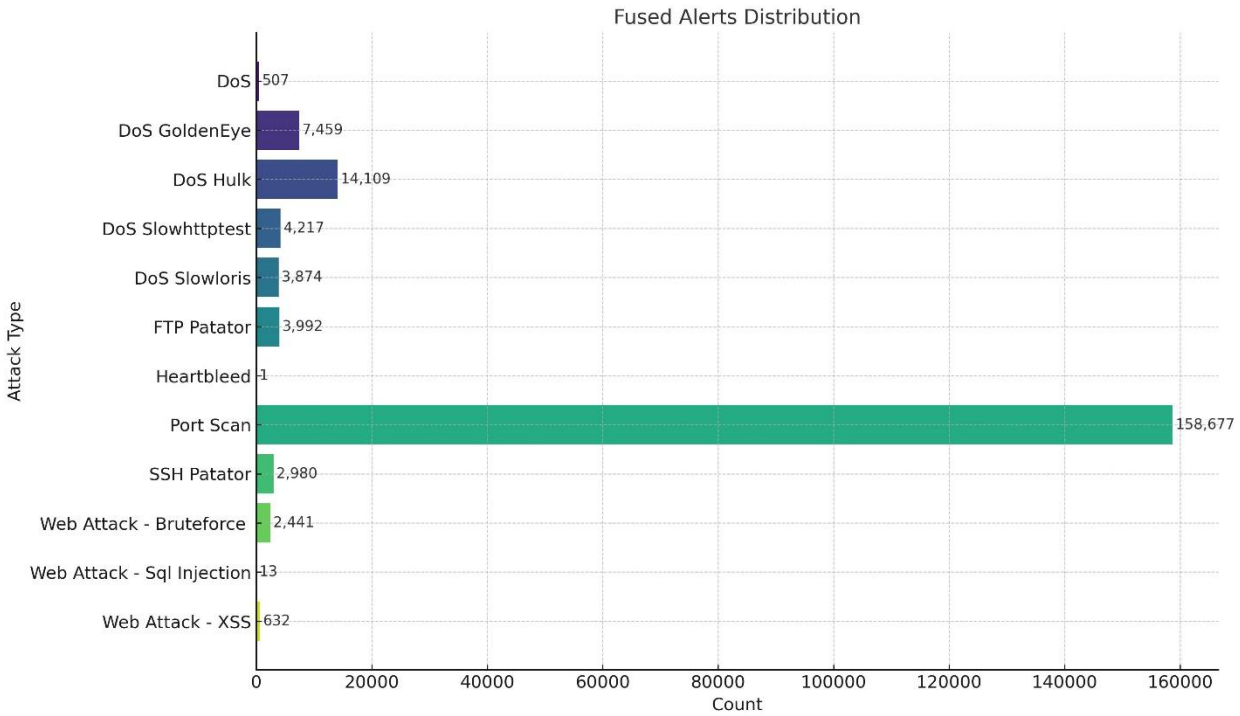



***Figure 11:-*** Alert Distribution after Fusion

## 4.4. Analysis/Discussion

As discussed above, in this proposed architecture, there are three streams of threat detections involving the IDS (Suricata) ingesting live traffic, the deep learning component taking in network flows and the correlation engine collecting and analyzing logs and alerts from the IDS.

As it can be seen from the results section of this paper, the correlation engine of the classic SIEM (Wazuh) solution was able to produce alerts based on the pre-configured correlation rules; where it cannot produce any further alerts unless the analyst feed the attack contexts via the rules.

What we can see from the output graph in ***Figure 10*** is that there are alerts the correlation engine could not detect from the event logs, even if the attacks are there. Furthermore, as it is displayed in ***Figure 12*** below, there are alerts that the correlation engine misclassified as false positives, as the scenarios or the indicated IP addresses were not classified as potential attacks or victims in ***Table 1*** above.

Hence, this shows two critical points: a) the threat detection capability of the correlation engine depends on the knowledge which should be pre-encoded by the analyst in order for the engine to capture it, b) there is a high probability that those correlation rules might misclassify normal event logs as if they are attacks.

As displayed in the pie chart below, 37.3 % or about 964 alerts are considered to be either misclassifications or normal scenarios being reported as attacks. That is, correlation rules, or well-known patterns by the analyst, might misclassify events.

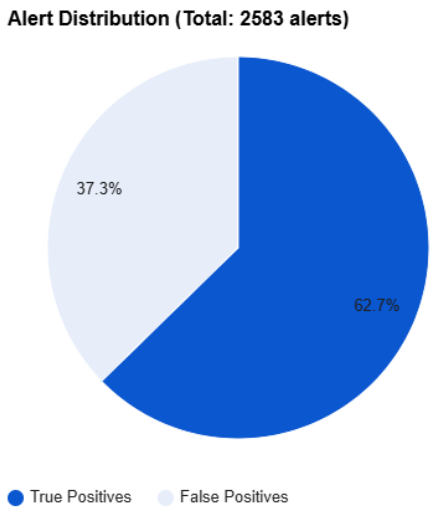



***Figure 12:-*** Correlation Alerts Distribution

***Table 5***: Evaluation Metrics for Correlation Rule Implementation

| Model | True Positives | False Positives | Precision | |
|---|---|---|---|---|
| Correlation Rule | 37.3% | 62.7% | 0.62679 | |

When we consider the event flow analysis by the deep learning implementation of multi-layer perceptron (MLP), as the data was made ready for intrusion detection evaluation, and as the algorithm was trained on this same dataset, it was able to build a model with accuracy of (94.2%). This accuracy is able to positively influence the final output of the alert fusion block.

When we examine the final block, which is the alert fusion block, there are interesting facts that we can see based on the metrics of false positives, detection accuracy and manageability of the alerts.

One of the techniques that majority voting works with is that it assigns voting power to each alert sources and those alerts that are corroborated by the many will be considered as legitimate threats.

Here are the findings that we can witness from the result of the alert fusion:

1. The total number of alerts, which could have been 560,197, has been minimized to 244,291 with the following rationale:
    a) Total number of alerts of the correlation engine has been minimized from 2584 to 2306 by removing duplicated and false positives
    b) Total number of alerts of the deep learning engine has been minimized from 555,645 to 242,703
    c) Total number of alerts of the traditional intrusion detection system (Suricata) has been minimized from 1968 to 1903.
2. When we examine the dynamics of the alerts produced from all the sources, the distribution of the duplicates is as follows:
    a) There were 277 duplicate alerts from the correlation engine
    b) There were 312, 941 duplicate alerts from the deep learning implementation
    c) There were 63 duplicate alerts from the traditional intrusion detection system (Suricata)
3. Here is the interesting fact, even though the total number of alerts from all the sources was 560,197, the total number of alerts after fusion has become 244,291. This shows a difference of 315,906. But when we see the total number of duplicates from all the sources, it shows that it is 313,281.
4. From #3 above, we can see that there is a difference of 2625 alerts between the total number of duplicates and that of the one after alert fusion.

The above four points, with regard to the number of alerts in different scenarios, such as total number of alerts, number of duplicate alerts and those alerts other than duplicates but are dropped after alert fusion is depicted in the graph below.

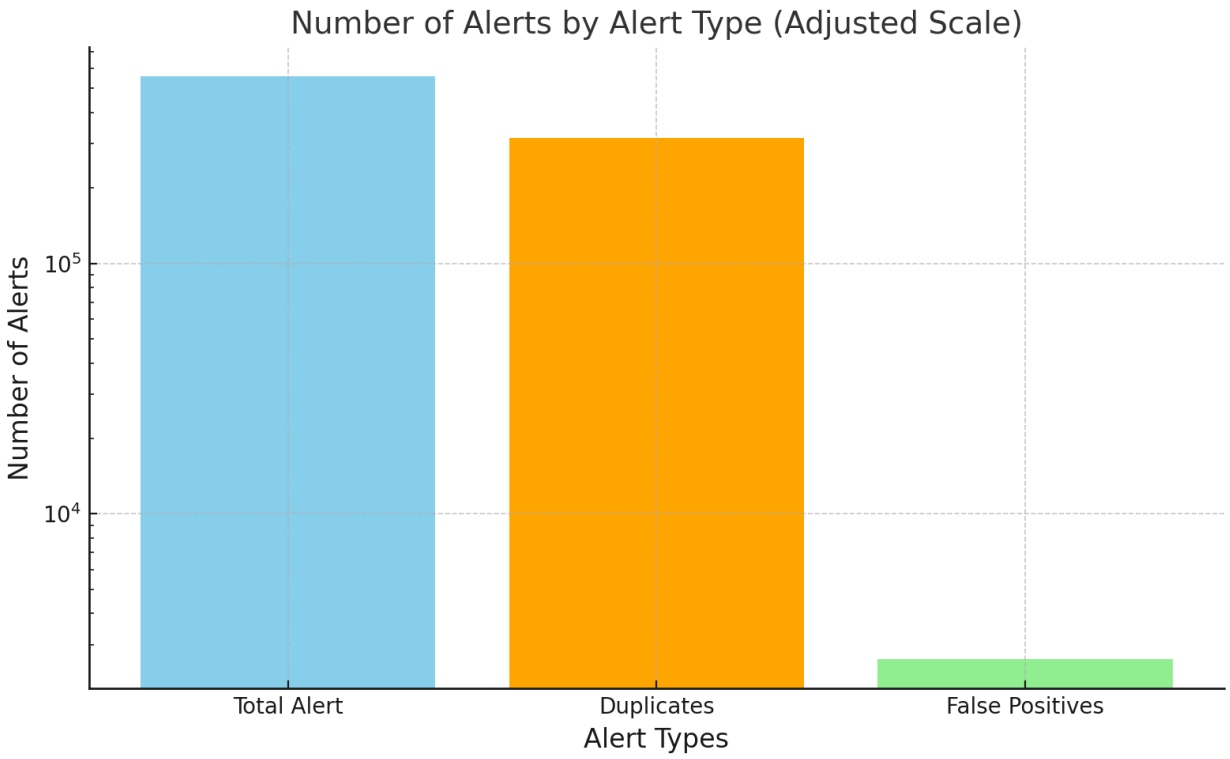



***Figure 13:-*** Alerts Distribution with Duplicates and False Positives

## Observation on the Alert Fusion

Based on the result above, the following are the interesting observations from the alert dynamics created in different scenario:

The first observation is that the total number of alerts is much minimized to less than 44%, of the total number of alerts before fusion, during the alert fusion stage due to the fact that the fusion stage removed the duplicate alerts coming from all sources. To be exact the number of alerts came down to 43.61% of the total amount of alerts before fusion. This has brought a great deal of manageability to the alerts so that the analysts comprehend what is going on in their network without going to the alert-fatigue condition.

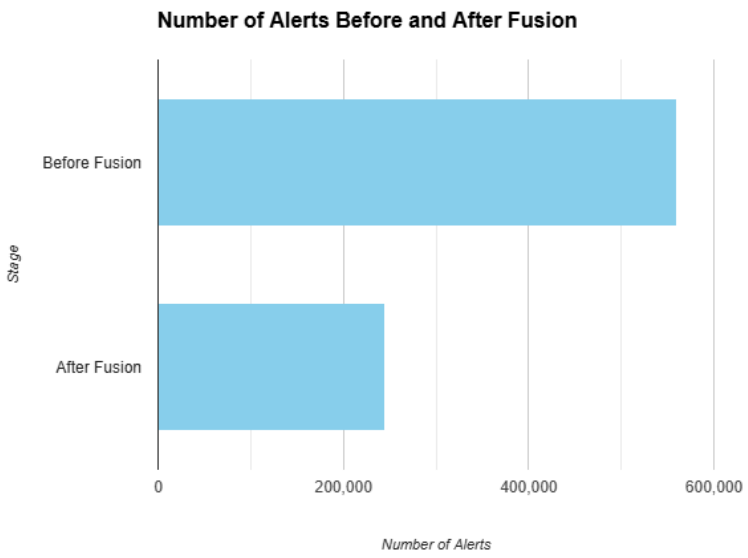



***Figure 14:-*** Number of Alert Comparison - Before and After Fusion

The second observation is that there cannot be additional false positives created during the alert fusion stage due to the fact that majority voting technique is consolidating the best capabilities of each sources or analytics where all alerts with better votes put into a melting pot for decision making.

Thirdly, as this research is aiming to enhance the analytics with deep learning and alert fusion, the better deep learning techniques/algorithms are used, the better the outcome of the alert fusion will be as the alert quality and the voting weight of the deep learning stage influence the outcome of the alert fusion by much.

The fourth observation is, even better, the more complex and advanced the alert fusion techniques are, even including the deep learning implementation of the majority voting, the more efficient and optimized the final outcome of the alert fusion.

Furthermore, we can clearly see that some alerts that were not in one of the alert sources appearing in the fusion result, implying that these alert sources complement each other as long as the sources have the proper vote, which in turn maximizes the threat detection capability and accuracy.

On the other hand, we can clearly see that new false positives will not be created, other than those that have already appeared either in the alert sources or in the final fusion version. Instead, interestingly, there were many false positives, that used to appear in the individual sources, dropped and that did not appear in the final fused alert list. We can see this in the Figure 13 above and in the observation above under #4. To be exact, there were 2625 alerts dropped that were not in the duplicates, but happened to be detected as false positives in the individual sources and dropped at the alert fusion stage. This enhances the detection accuracy of the combination of the deep learning and that of the alert fusion.

As an illustration we can see that those alerts from the correlation engine coming from the source IPs 192.168.10.15 and 192.168.10.19 and that were categorized as false positive in the above graph (37.3%) were totally dropped in the alert fusion stage, which minimizes the amount of false positives as well as enhances the detection accuracy.

In addition, due to the high detection accuracy of the deep learning component implemented and the complementary nature of alerts from the correlation engine and the deep learning, the detection accuracy of the system is improved.

Hence, we can be confident to say that the deep learning in combination with the majority voting, incorporating voting and deduplication of alerts, brought about a decrease in the amount of false positives and a better threat detection accuracy.

# Chapter 5

# 5. Conclusion and Future Work

## 5.1. Conclusion

In addressing the challenges of false positives and detection accuracy in traditional IDS and SIEM solutions, this research proposes a deep learning driven and multimodal data source-based SIEM solution that can integrate a variety of data sources (event logs, alerts, raw traffic and network flow) that can create a better insight into what is going on in a given network at a situational awareness level. Furthermore, using alert fusion technique as a major analytics method driven by deep learning algorithms showed a better detection accuracy than the individual analytics techniques.

## 5.2. Future Work

This research has shown the framework by which multimodal data sources could be directed and analyzed to describe what is really going on in a given network. The other component of this framework is where deep learning algorithm enhances the detection accuracy of SIEM solution. And finally, this framework proves how alert fusion implementations minimizes false positives and enhance detection accuracy of the SIEM solution.

Hence as future work, the following tasks could be considered. Implementation of the proposed architecture in a real-time environment has a potential to reveal better enhancements of the implemented analytics techniques. Furthermore, simulating this experiment's setup in a real-time context would enable to watch some parameters which cannot be observed here.

For instance, the effect of the network latency that could arise among the 5 different blocks of the proposed architecture could easily be seen in a real environment.

On the other hand, this will enable the experiment to capture the real traffic, fed to the IDS, where by the event logs generated by the IDS too will directly be collected by the SIEM agents for the purpose of correlation.

Furthermore, implementation of alert fusion techniques that are more complex and that are also based on deep learning techniques, can be on stream of future work.

And finally incorporating more multimodal data sources such as more event log sources, user profiles and OSINT that can be fed to the correlation engine, the deep learning component and at last to the alert fusion module.

# References


[1] Federica Uccello, Mark Pawlicki, Salvatore D'Antonio, Rafal Kozik, Michal Choras, "Towards Hybrid NIDS: Combining Rule-Based SIEM with AI-Based Intrusion Detectors", University of Naples 'Parthenope' Centro Direzionale, 80133 Napoli, Isola C4, Italy; Bydgoszcz University of Science and Technology, PBS, Bydgoszcz, Poland, 2024. (11)

[2] Gustavo González-Granadillo, Susana González-Zarzosa, Rodrigo Diaz, "Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures", Cybersecurity Unit, Atos Research & Innovation, ATOS Spain, 28037 Madrid, Spain, Jul. 12, 2021. (8)

[3] Mrs. Ch. Ramya Bharathi, L. Mounika, D. Amulya, L. Niteesh, D.Chinna Khasim, "Cyber Threat Detection Using Neural Networks And Machine Learning", Department of CSE-Artificial Intelligence & Machine Learning, SRK Institute of Technology, Vijayawada, 2024. (9)

[4] Nabil Moukafih, Ghizlane Orhanou, Said El Hajji, "Neural Network-Based Voting System with High Capacity and Low Computation for Intrusion Detection in SIEM/IDS Systems", Laboratory of Mathematics, Computing and Applications-Information Security, Faculty of Sciences, Mohammed V University in Rabat, BP1014 RP, Rabat, Morocco, Jul. 16, 2020. (1)

[5] Kleber Stroeh, Edmundo Roberto Mauro Madeira, Siome Klein Goldenstein, "An approach to the correlation of security events based on machine learning techniques", Icaro Technologies, Campinas, SP, Brazil, 2013 (5)

[6] Vivek Darak, Mohak Gadge, Shreyash Dhangare, "A Machine Learning Framework for Cybersecurity Operations", Pune Institute of Computer Technology, Pune, Mar. 03, 2021. (3)

[7] Rathish Mohan, Abhishek Vajpayee, Srikanth Gangarapu, Vishnu Vardhan Reddy Chilukoori, "Mitigating Complex Cyber Threats: An Integrated Multimodal Deep Learning Framework for Enhanced Security", Lore Health LLC, USA, Metropolis Technologies, USA, AT&T Services INC, USA, Amazon.com Services LLC, USA, Sep 2024. (12)

[8] Haitao He, Xiaobing Sun, Hongdou He, Guyu Zhao, Ligang He , And Jiadong Ren, "A Novel Multimodal-Sequential Approach Based on Multi-View Features for Network Intrusion Detection", College of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China, Department of Computer Science, University of Warwick,

Coventry CV4 7AL, U.K., 3Key Computer Virtual Technology and System Integration Laboratory, Yanshan University, Qinhuangdao 066004, China, Dec. 27, 2019. (13)

[9] Azran Abdul Razak , Helmy Hanyff Hairudin Ruzaili , Mohamad Fadli Zolkipli, "Study on Machine Learning Implementation in Cybersecurity for Security Defend and Attack", Jabatan Teknologi Maklumat dan Komunikasi (JTMK), Politeknik Tuanku Syed Sirajuddin (PTSS), Arau, Perlis, Malaysia, School of Computing, College of Arts and Sciences Universiti Utara Malaysia, Sintok, Kedah, Malaysia, Jun. 04, 2024. (15)

[10] Srinivas Reddy Pulyala, "From Detection to Prediction: AI-powered SIEM for Proactive Threat Hunting and Risk Mitigation", SmileDirectClub, USA, 2024. (14)

[11] Muhammad Sheeraz, Muhammad Arsalan Paracha, Mansoor UI Haque, Muhammad Hanif Durad, Seyd Muhammad Mohsin, Shahab S. Band, Amir Mosavi, "Effective-Security-Monitoring-Using-Efficient-SIEM-Architecture", April 30, 2023 (20)

[12] Fatima Rashed Alzaabi, Abid Mehmood, (Member, IEEE), "A Review of Recent Advances, Challenges, and Opportunities in Malicious Insider Threat Detection Using Machine Learning Methods", College of Engineering, Abu Dhabi University, Abu Dhabi, United Arab Emirates, Mar. 01, 2024. (10)

[13] D.W.Y.O.Waidyarathna, W.V.A.C.Nayantha, W.M.T.C.Wijesinghe, Kavinga Yapa Abeywardena, "Intrusion Detection System with Correlation Engine and Vulnerability Assessment", Department of Information System Engineering, Sri Lanka Institute of Information Technology, Kandy Road, Malabe, Sri Lanka, 2018. (2)

[14] Srinivas Reddy Pulyala, "From Detection to Prediction: AI-powered SIEM for Proactive Threat Hunting and Risk Mitigation", SmileDirectClub, USA, 2024. (16)

[15] Salwa Elsayed , Khalil Mohamed, And Mohamed Ashraf Madkour, "A Comparative Study of Using Deep Learning Algorithms in Network Intrusion Detection", Systems and Computers Engineering Department, Faculty of Engineering, Al-Azhar University, Nasr City, Cairo 11765, Egypt, May 01, 2024. (18)

[16] Akhila Anilkumar, Alona Shibu, Meera Anna Varghese, Priya P Sajan, A.L. Sreedeep, "Detecting and Analysing Network Logs Using Machine Learning Techniques", Gestao, Inovacaoe, Tecnologias, Dec. 05, 2021. (6)

[17] Chinthapalli Sudheer, M S Venugopala Rao, “User-Centric Machine Learning Framework for Cyber Security Operation Center”, SKBR PG COLLEGE, AMALAPURAM, E.G.DIST, ANDHRA PRADESH – 533201, Mar. 01, 2020. (4)

[18] B. Ramesh, J. Vineeth Kumar, Vuppala Vyshnavi, Mutha Prem, “Artificial Intelligence framework for Cyber Attack Detection and Notifying using Machine Learning Techniques”, Institute of Technology, Hyderabad, Telangana, Apr. 04,2021 (7)

[19] Diogo Gaspar, Paulo Silva, And Catarina Silva, “Explainable AI for Intrusion Detection Systems: LIME and SHAP Applicability on Multi-Layer Perceptron”, Laboratório de Informática e Sistemas (LIS), Instituto Pedro Nunes, 3030-199 Coimbra, Portugal, February 2024.

[20] Richard Kimanzi, Peter Kimanga, Dedan Cherori, Patrick K. Gikunda, “Deep Learning Algorithms Used in Intrusion Detection Systems - A Review”, Department of Computer Science, Dedan Kimathi University of Technology, P.O Box Private Bag - 10143, Nyeri, Kenya, Feb. 26, 2024

[21] Madjid G. Tehrani, Eldar Sultanow, William J. Buchanan, Malik Amir, Anja Jeschke, Mahkame Houmani, Raymond Chow, Mouad Lemoudden, “Stabilized quantum-enhanced SIEM architecture and speed-up through Hoeffding tree algorithms enable quantum cybersecurity analytics in botnet detection”, 2024

[22] Jose Luis Gutierrez-Garcia, Eddy Sanchez-DelaCruz, and Maria del Pilar Pozos-Parra, “A Review of Intrusion Detection Systems Using Machine Learning: Attacks, Algorithms, and Challenges”, 2023

[23] Tao Ban, Takeshi Takahashi, Samuel Ndichu, Daisuke Inoue, “Breaking Alert Fatigue: AI-Assisted SIEM Framework for Effective Incident Response”, Cybersecurity Research Institute, National Institute of Information and Communications Technology, Tokyo, 29 May, 2023

[24] Muhammad Sheeraz, Muhammad Arsalan Paracha, Mansoor UI Haque, Muhammad Hanif Durad, Seyd Muhammad Mohsin, Shahab S. Band, Amir Mosavi, “Revolutionizing SIEM Security: An Innovative Correlation Engine Design for Multi-Layered Attack Detection”, July 28, 2024

[25] Jose Luis Gutierrez-Garcia, Eddy Sanchez-DelaCruz, Maria del Pilar Pozos-Parra, “A Review of Intrusion Detection Systems Using Machine Learning: Attacks, Algorithms, and Challenges”, 2023

[26] Johan Note and Maaruf Ali, “Comparative Analysis of Intrusion Detection System Using Machine Learning and Deep Learning Algorithms”, Epoka University, Albania. July 1, 2022

[27] Adabi Raihan Muhammad, Parman Sukarno, Aulia Arif Wardana, “Integrated Security Information and Event Management (SIEM) with Intrusion Detection System (IDS) for Live Analysis based on Machine Learning”, Telkom University, Bandung, Indonesia, Wrocław University of Science and Technology, Wrocław, Poland, 2022

[28] Massimo Guarascio ∗, Nunziato Cassavia, Francesco Sergio Pisani, Giuseppe Manco, “Boosting Cyber-Threat Intelligence via Collaborative Intrusion Detection” Institute for High Performance Computing and Networking, Italian National Research Council, ICAR-CNR, Via P. Bucci, 87036, Rende (CS), Italy, April 30, 2022

[29] Tim Laue, Timo Klecker, Carsten Kleiner, Kai-Oliver Detken, “A SIEM Architecture for Advanced Anomaly Detection”, Hochschule Hannover - University of Applied Sciences and Arts, Ricklinger Stadtweg 120, 30459 Hannover, Germany, {tim.laue | carsten.kleiner }@hs-hannover.de BDECOIT GmbH, Fahrenheitstrase 9, 28359 Bremen, Germany {klecker | detken }@decoit.de, 2022

[30] Vasudeva, “Comparative analysis of Machine Learning algorithms for Intrusion Detection”, IOP Conf. Ser.: Mater. Sci. Eng. 1013 012038, 2021

[31] Gyana Ranjana Panigrahi, Prabira Kumar Sethy, Santi Kumari Behera, Manoj Gupta, Farhan A. Alenizi, Pannee Suanpang, Aziz Nanthaamornphong, “Analytical Validation and Integration of CIC-Bell-DNS-EXF-2021 Dataset on Security Information and Event Management”, IEEE Members, 2024

# Appendix

## Appendix A

### A 1 – Screenshots from the Implementation

### A 1.1 - Sample correlation rule used in the correlation engine.

```
<group name="DoS Attack Rules">
    <rule id="100021" level="5">
      <decoded_as>syslog</decoded_as>
      <description>Potential DoS: Excessive failed connections</description>
      <group>dos,syslog,failed_connection</group>
      <signature>connection refused|too many open files</signature>
      <frequency>50</frequency>
      <timeframe>60</timeframe>
      <firedtimes>5</firedtimes>
      <mitre>
        <id>T1499</id>
        <tactic>Impact</tactic>
        <technique>Denial of Service</technique>
      </mitre>
    </rule>

    <rule id="100022" level="10">
      <decoded_as>syslog</decoded_as>
      <description>Potential DoS: Multiple failed connections from the same IP</description>
      <group>dos,syslog,bruteforce</group>
      <if_sid>100021</if_sid>
      <same_source_ip />
      <frequency>100</frequency>
      <timeframe>120</timeframe>
      <firedtimes>5</firedtimes>
      <mitre>
        <id>T1499</id>
        <tactic>Impact</tactic>
        <technique>Denial of Service</technique>
      </mitre>
    </rule>

    <rule id="100023" level="12">
      <decoded_as>syslog</decoded_as>
      <description>DoS: System resource exhaustion</description>
      <group>dos,syslog,resource_exhaustion</group>
      <signature>kernel: out of memory|memory allocation failure|too many processes|CPU load too high</signature>
      <frequency>15</frequency>
      <timeframe>30</timeframe>
      <firedtimes>1</firedtimes>
      <mitre>
        <id>T1499</id>
        <tactic>Impact</tactic>
        <technique>Denial of Service</technique>
      </mitre>
    </rule>

    <rule id="100024" level="15">
      <description>Correlated DoS attack detected</description>
      <group>dos,syslog</group>
      <if_sid>100022,100023</if_sid>
      <frequency>3</frequency>
      <timeframe>600</timeframe>
      <firedtimes>1</firedtimes>
      <mitre>
        <id>T1499</id>
        <tactic>Impact</tactic>
        <technique>Denial of Service</technique>
      </mitre>
    </rule>
</group>
```

## A 1.2 – Data Collection and Count

```
from google.colab import drive
drive.mount('/content/drive')
#data_path = "/content/drive/MyDrive/IDS_BHL/"
data_path = "/content/drive/MyDrive/MyMy/Data/"
os.listdir(data_path)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
['Monday-WorkingHours.pcap_ISCX.csv',
 'Wednesday-workingHours.pcap_ISCX.csv',
 'Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv',
 'Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv',
 'Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv',
 'Tuesday-WorkingHours.pcap_ISCX.csv',
 'Friday-WorkingHours-Morning.pcap_ISCX.csv',
 'Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv']
```

```
files = os.listdir(data_path)
dict_data = {}
for file in files:
  dict_data[file] = pd.read_csv(data_path+file , encoding='latin-1')
for key,value in dict_data.items():
  print(key,value.shape)
```

```
Monday-WorkingHours.pcap_ISCX.csv (529918, 85)
Wednesday-workingHours.pcap_ISCX.csv (692703, 85)
Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv (288602, 85)
Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv (458968, 85)
Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv (286467, 85)
Tuesday-WorkingHours.pcap_ISCX.csv (445909, 85)
Friday-WorkingHours-Morning.pcap_ISCX.csv (191033, 85)
Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv (225745, 85)
```

## A 1.3 – Statistical View of the Data

```
merged_frames.describe()
```

| | Source Port | Destination Port | Protocol | Flow Duration | Total Fwd Packets | Total Backward Packets | Total Length of Fwd Packets | Total Length of Bwd Packets | Fwd Packet Length Max | Fwd Packet Length Min | ... | act_data_pkt_fwd | min_seg_size_forwa |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 2.830743e+06 | 2.830743e+06 | 2.830743e+06 | 2.830743e+06 | 2.830743e+06 | 2.830743e+06 | 2.830743e+06 | 2.830743e+06 | 2.830743e+06 | 2.830743e+06 | ... | 2.830743e+06 | 2.830743e+ |
| mean | 4.112886e+04 | 8.071483e+03 | 9.880341e+00 | 1.478566e+07 | 9.361160e+00 | 1.039377e+01 | 5.493024e+02 | 1.616264e+04 | 2.075999e+02 | 1.871366e+01 | ... | 5.418218e+00 | -2.741688e+ |
| std | 2.229494e+04 | 1.828363e+04 | 5.261922e+00 | 3.365374e+07 | 7.496728e+02 | 9.973883e+02 | 9.993589e+03 | 2.263088e+06 | 7.171848e+02 | 6.033935e+01 | ... | 6.364257e+02 | 1.084989e+ |
| min | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | -1.300000e+01 | 1.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | ... | 0.000000e+00 | -5.368707e+ |
| 25% | 3.277400e+04 | 5.300000e+01 | 6.000000e+00 | 1.550000e+02 | 2.000000e+00 | 1.000000e+00 | 1.200000e+01 | 0.000000e+00 | 6.000000e+00 | 0.000000e+00 | ... | 0.000000e+00 | 2.000000e+ |
| 50% | 5.094400e+04 | 8.000000e+01 | 6.000000e+00 | 3.131600e+04 | 2.000000e+00 | 2.000000e+00 | 6.200000e+01 | 1.230000e+02 | 3.700000e+01 | 2.000000e+00 | ... | 1.000000e+00 | 2.400000e+ |
| 75% | 5.841300e+04 | 4.430000e+02 | 1.700000e+01 | 3.204828e+06 | 5.000000e+00 | 4.000000e+00 | 1.870000e+02 | 4.820000e+02 | 8.100000e+01 | 3.600000e+01 | ... | 2.000000e+00 | 3.200000e+ |
| max | 6.553500e+04 | 6.553500e+04 | 1.700000e+01 | 1.200000e+08 | 2.197590e+05 | 2.919220e+05 | 1.290000e+07 | 6.554530e+08 | 2.482000e+04 | 2.325000e+03 | ... | 2.135570e+05 | 1.380000e+ |

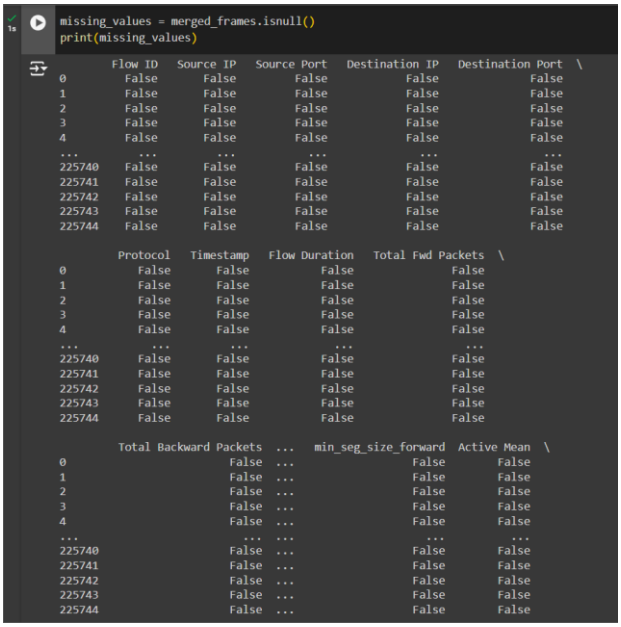
8 rows × 80 columns

## A 1.4 – Training Data Count

```
merged_frames.shape
```

```
(3119345, 85)
```

## A 1.5 – Data Preprocessing for Missing Values



```
missing_values = merged_frames.isnull()
print(missing_values)
```

```
        Flow ID  Source IP  Source Port  Destination IP  Destination Port  \
0         False      False        False           False             False
1         False      False        False           False             False
2         False      False        False           False             False
3         False      False        False           False             False
4         False      False        False           False             False
...         ...        ...          ...             ...               ...
225740    False      False        False           False             False
225741    False      False        False           False             False
225742    False      False        False           False             False
225743    False      False        False           False             False
225744    False      False        False           False             False

        Protocol  Timestamp  Flow Duration  Total Fwd Packets  \
0          False      False          False              False
1          False      False          False              False
2          False      False          False              False
3          False      False          False              False
4          False      False          False              False
...          ...        ...            ...                ...
225740     False      False          False              False
225741     False      False          False              False
225742     False      False          False              False
225743     False      False          False              False
225744     False      False          False              False

        Total Backward Packets  ...  min_seg_size_forward  Active Mean  \
0                        False  ...                 False        False
1                        False  ...                 False        False
2                        False  ...                 False        False
3                        False  ...                 False        False
4                        False  ...                 False        False
...                        ...  ...                   ...          ...
225740                   False  ...                 False        False
225741                   False  ...                 False        False
225742                   False  ...                 False        False
225743                   False  ...                 False        False
225744                   False  ...                 False        False
```

# Appendix B

## B 1 – Source Code

### B 1.1 – Data Loading and Mounting

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import os
from google.colab import drive
drive.mount('/content/drive')
from google.colab import drive
drive.mount('/content/drive')
#data_path = "/content/drive/MyDrive/IDS_BHL/"
data_path = "/content/drive/MyDrive/MyMy/Data/"
os.listdir(data_path)
```

### B 1.2 – Describing the Data

```
files = os.listdir(data_path)
dict_data = {}
for file in files:
  dict_data[file] = pd.read_csv(data_path+file , encoding='latin-1')
for key,value in dict_data.items():
  print(key,value.shape)
```

**B 1.3 – Describing Key-Value Pair with respect to different statistics such as source and destination IP, source and destination port, flow duration, total forwarded packets, minimum segment size forwarded and so on.**

```
for key,value in dict_data.items():
  print(key,value.head())
```

**B 1.4 – Describing Each Column with additional column characteristics and statistical values describing the data.**

```
for i,j in dict_data.items():
  print(i,j.columns)
  print(j.describe())
  print(j.info)
  print(j.isnull().sum())
  print("-----------------------------------------------")
```

**B 1.5 – Describing and comparing the column names.**

```
for i,j in dict_data.items():
  print(i,j.columns)
```

**B 1.6 – Merging and Describing Each Data Frame**

```
frames = []
for key,value in dict_data.items():
  frames.append(value)
merged_frames = pd.concat(frames)
print(merged_frames.shape)
#merged_frames = [df1, df2, df3]
merged_frames.head()
```

```
merged_frames.describe()

merged_frames.info()

merged_frames.shape

merged_frames.columns

merged_frames.dtypes
```

**B 1.7 – Counting the occurrence of each class label**

```
label_count = merged_frames[' Label'].value_counts()

label_count
```

**B 1.8 – Plotting the occurrence of class labels on a bar-graph.**

```
label_count.plot(kind='bar' , xlabel='Type' , ylabel='Count of records'
,title="Count of Labels")

plt.show()
```

**B 1.9 – Managing missing values**

```
missing_values = merged_frames.isnull()

print(missing_values)
```

**B 1.10 – Assigning random numbers to class labels**

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

merged_frames[' Label'] = label_encoder.fit_transform(merged_frames['
Label'])


print(list(label_encoder.classes_))

print(merged_frames[' Label'].unique())
```

**B 1.11 – Describing the data with respect to the destination port**

```
import matplotlib.pyplot as plt
import seaborn as sns


#pd.value_counts(merged_frames['']).plot.bar()
c = pd.value_counts(merged_frames[' Destination Port'])
c.sort_values(ascending=False).head(10)
```

**B 1.12 – Converting the merged frame data to a CSV format.**

```
merged_frames.to_csv('merged_data.csv', index=False)
```

**B 1.13 – Preprocessing for Missing Values (Dropping the NaN values assigned above)**

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
import csv


#data = pd.read_csv('merged_data.csv', low_memory=False)
chunk_size = 100000


# Numerical features to scale
# numerical_features = ['Flow ID', 'Source IP', 'Destination IP', 'Label']


data = pd.read_csv('merged_data.csv', chunksize = chunk_size)


scaler = StandardScaler()


with open('merged_data.csv', 'r') as file:
    reader = csv.reader(file)
```

```
    row_count = sum(1 for row in reader) - 1  # Subtract 1 to ignore the
header row
    print(f'Total rows: {row_count}')


# Process each chunk
for chunk in data:
    # Apply the drop method to each chunk
    chunk.columns = chunk.columns.str.strip()
    chunk = chunk.drop(['Source IP', 'Destination IP'], axis=1)


    numerical_features = chunk.select_dtypes(include=['float64',
'int64']).columns


    chunk.replace([np.inf, -np.inf], np.nan, inplace=True)  # Replace
infinity with NaN
    chunk.dropna(inplace=True)


     # Skip processing if the chunk is empty
    if chunk.shape[0] == 0:
        print("Empty chunk after dropping NaN values, skipping...")
        continue


    # Apply StandardScaler to the numerical columns only
    chunk[numerical_features] =
scaler.fit_transform(chunk[numerical_features])


    X = chunk.iloc[:, :-1].values
    y = chunk.iloc[:, -1].values
```

**B 1.14 – Managing the data for training the algorithm (Splitting the data into Training and Test Set)**

```
import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from sklearn.model_selection import train_test_split

import pandas as pd


import numpy as np

# Assuming y_train contains your target labels

#num_classes = len(np.unique(y_train))  # Calculate the number of unique classes


# Split data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


num_classes = len(np.unique(y_train))  # Calculate the number of unique classes


print(X_train.shape)
```

**B 1.15 – Training, Saving and Applying the Model**

```
import pandas as pd

# Create MLP model

model = Sequential()

model.add(Dense(128, activation='relu', input_dim=X_train.shape[1]))  # Input layer with 84 neurons

model.add(Dense(64, activation='relu'))  # Hidden layer with 64 neurons

model.add(Dense(32, activation='relu'))  # Hidden layer with 32 neurons
```

```
model.add(Dense(num_classes, activation='softmax'))  # Output layer with
num_classes neurons (adjust num_classes)


# Compile the model

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])


# Assuming X_train and X_test are originally NumPy arrays, convert them to
DataFrames

X_train = pd.DataFrame(X_train)

X_test = pd.DataFrame(X_test)

y_train = pd.DataFrame(y_train)

y_test = pd.DataFrame(y_test)


X_train.columns = X_train.columns.astype(str)

X_test.columns = X_test.columns.astype(str)

y_train.columns = y_train.columns.astype(str)

y_test.columns = y_test.columns.astype(str)


'''X_train.columns = X_train.columns.str.strip()

X_test.columns = X_test.columns.str.strip()

y_train.columns = y_train.columns.str.strip()

y_test.columns = y_test.columns.str.strip()'''


#X_train = X_train.drop(['1', '3'], axis=1)
#X_test = X_test.drop(['1', '3'], axis=1)

#y_train = y_train.drop(['1', '3'], axis=1)

#y_test = y_test.drop(['1', '3'], axis=1)


# Example of replacing IP addresses with NaN (or apply specific encoding)
```

```
X_train = X_train.applymap(lambda x: pd.to_numeric(x, errors='coerce'))

X_test = X_test.applymap(lambda x: pd.to_numeric(x, errors='coerce'))

y_train = y_train.applymap(lambda x: pd.to_numeric(x, errors='coerce'))

y_test = y_test.applymap(lambda x: pd.to_numeric(x, errors='coerce'))


X_train = X_train.astype('float32')

X_test = X_test.astype('float32')

y_train = y_train.astype('int')

y_test = y_test.astype('int')


print(X_train.shape)

print(X_test.shape)

print(y_train.shape)

print(y_test.shape)


# Train the model

model.fit(X_train, y_train, epochs=1000, batch_size=32,
validation_data=(X_test, y_test))


# Evaluate the model

loss, accuracy = model.evaluate(X_test, y_test)

print('Test accuracy:', accuracy)


filename = 'finalized_model.sav'

pickle.dump(model, open(filename, 'wb'))
```