Collection, analysis and harnessing of communication flows for cyber-attack detection.

Almamy Toure

Université Polytechnique Hauts-de-France

INSA Hauts-de-France

Doctoral School Polytechnique Hauts-de-France (ED PHF n°635)

Laboratory of Industrial and Human Automation control Mechanical engineering and Computer science (LAMIH – UMR CNRS 8201)

IBM Services Center France

Thesis submitted and presented for PhD graduation in Computer Science and Application

By Almamy TOURE

On July 05, 2024 in Valenciennes

Collection, analysis and harnessing of communication flows for cyber-attacks detection.

Composition of the Jury

David Espes	*Professor, Université de Bretagne Occidentale*	**Reviewer**
Florence Sedes	*Professor, Université Toulouse3 Paul Sabatier*	**Reviewer**
Nathalie Mitton	*Research Director, Inria Lille-Nord Europe*	**President**
Frédérique Laforest	*Professor, INSA Lyon*	**Examiner**
Antoine Gallais	*Professor, UPHF - INSA Hauts-de-France*	**Co-Director**
Thierry Delot	*Professor, UPHF*	**Co-Director**
Youcef Imine	*Associate Professor, UPHF*	**Co-supervisor**

Université Polytechnique Hauts-de-France

INSA Hauts-de-France

École Doctorale Polytechnique Hauts-de-France (ED PHF n°635)

Laboratoire d'Automatique, de Mécanique et d'Informatique Industrielles et Humaines (LAMIH – UMR CNRS 8201)

IBM Services Center France

Thèse soumise et présentée pour obtenir le grade de Docteur en Informatique et Applications

Par Almamy TOURE

Le 05 Juillet 2024 à Valenciennes

Collecte, analyse et exploitation des flux de communication pour la détection de cyberattaques.

Composition du jury

David Espes	*Profeseur, Université de Bretagne Occidentale*	**Rapporteur**
Florence Sedes	*Professeure, Université Toulouse3 Paul Sabatier*	**Rapporteur**
Nathalie Mitton	*Directrice de Recherche, Inria Lille-Nord Europe*	**Présidente**
Frédérique Laforest	*Professeure, INSA Lyon*	**Examinateur**
Antoine Gallais	*Professeur, UPHF - INSA Hauts-de-France*	**Co-Directeur**
Thierry Delot	*Professeur, UPHF*	**Co-Directeur**
Youcef Imine	*Maître de conférences, UPHF*	**Co-Encadrant**

Abstract

The increasing complexity of cyberattacks, characterized by a diversification of techniques and tactics, an expansion of attack surfaces, and a growing interconnection of applications with the Internet, makes network traffic management in professional environments imperative. This management has become a crucial element in ensuring the security of exchanged data and preventing the compromise of information systems. In this regard, the collection and analysis of communication flows and other security events play a fundamental role for businesses across all sectors.

Within the scope of this PhD thesis, we have focused on enterprise network traffic management with the aim of detecting cyberattacks using dynamic approaches. This management involves comprehensive collection of communication flows, meticulous end-to-end analysis, and the utilization of these data to differentiate between legitimate and malicious traffic. The objective is to ensure the detection of attack scenarios, whether they are simple or complex, known or unknown, while addressing challenges related to the encrypted and obfuscated nature of data flows.

Faced with the diversity of existing solutions, ranging from static rule-based correlation approaches to obfuscation techniques, this thesis proposes an innovative approach based on the classification of network flows using one-dimensional convolutional neural networks (1D-CNN). This approach builds upon the fundamental principles of network flows and CNN, aiming to analyze the different communication phases of a network flow to extract generalist attributes and combine them with feature detection mechanisms unique to CNN. This dual extraction enables a better classification of network flows.

The effectiveness of our approach has been demonstrated through its evaluation on datasets from both industrial and community contexts. Our results have shown a significant reduction in the number of attributes used for flow classification, as well as a decrease in model execution time. Furthermore, our approach has maintained a high level of precision while reducing the rate of false detections.

Despite the effectiveness of known network traffic classification, it does not allow for the detection of new types of attacks. Indeed, cyber threats are constantly evolving, and zero-day attacks, which exploit previously unknown vulnerabilities, are becoming increasingly common. To address this challenge, this thesis proposes an innovative detection framework that combines supervised and unsupervised algorithms for effective zero-day attack detection. The framework encompasses the entire detection process, from data collection to anomaly identification. Traffic classification is performed using machine learning techniques, while anomaly identification is based on silhouette score, a measure of similarity between data points. Online learning is also integrated into the framework, enabling continuous model updates based on new collected data. This approach ensures effective detection of zero-day attacks even in a constantly evolving environment.

Finally, this thesis highlights a significant issue in current research regarding the datasets used to evaluate machine learning and deep learning models in the field of intrusion detection. These datasets are often outdated, do not reflect the real context of information systems, and are not tailored to the specific needs of industrial and scientific communities.

To address this issue, this thesis proposes a synthetic data generator that creates recent datasets adapted to the original context while ensuring the confidentiality of the original information system through anonymization. Moreover, it is not limited in terms of attack scenarios, allowing for testing intrusion detection models in varied and realistic conditions. The synthetic data generator relies on three major techniques to validate the generated data. Thus, this synthetic data generator enhances the relevance and reliability of intrusion detection model evaluations, contributing to the advancement of research in this crucial domain for information system security.

Abstract

La complexité croissante des cyberattaques, caractérisée par une diversification des techniques et tactiques, une expansion des surfaces d'attaque et une interconnexion croissante d'applications avec Internet, rendent impérative la gestion du trafic réseau en milieu professionnel. Cette gestion est devenue un élément crucial pour assurer la sécurité des données échangées et prévenir la compromission des systèmes d'information. Dans cette optique, la collecte et l'analyse des flux de communication ainsi que d'autres événements de sécurité jouent un rôle fondamental pour les entreprises de tous secteurs.

Dans le cadre de ce projet de recherche, nous nous sommes penchés sur la gestion du trafic réseau en entreprise dans le but de détecter les cyberattaques en utilisant des approches dynamiques. Cette gestion implique la collecte exhaustive des flux de communication, une analyse minutieuse de bout en bout, et l'exploitation de ces données pour différencier le trafic légitime du trafic malveillant. Ce processus vise à garantir la détection des scénarios d'attaques, qu'ils soient simples ou complexes, connus ou inconnus, tout en tenant compte des défis liés à la nature chiffrée et obfusquée des flux de données.

Face à la diversité des solutions existantes, allant des approches statiques de corrélation manuelle des règles aux techniques d'obfuscation, cette thèse propose une approche novatrice basée sur la classification des flux réseau à l'aide de réseaux de neurones à convolution (1D-CNN). Cette approche repose sur les principes fondamentaux des flux réseau et des CNN, visant à analyser les différentes phases de communication d'un flux réseau afin d'extraire des attributs généralistes et à les combiner avec les mécanismes de détection de caractéristiques propres aux CNN. Cette double extraction permet une meilleure classification des flux réseau. L'efficacité de notre approche a été démontrée à travers son évaluation sur des jeux de données provenant du contexte industriel et de la communauté. Nos résultats ont montré une réduction significative du nombre d'attributs utilisés pour la classification des flux, ainsi qu'une diminution du temps d'exécution des modèles. De plus, notre approche a permis de maintenir un niveau élevé de précision tout en réduisant le taux de fausses détections.

Malgré l'efficacité de la classification du trafic réseau connu, elle ne permet pas de détecter les nouveaux types d'attaques. En effet, les cybermenaces évoluent constamment et les attaques zero-day, qui exploitent des vulnérabilités encore inconnues, sont de plus en plus fréquentes. Pour faire face à ce défi, cette thèse propose un framework de détection innovant. Ce framework combine des algorithmes supervisés et non supervisés pour une détection efficace des attaques zero-day. Il traite l'ensemble du processus de détection, depuis la collecte des données jusqu'à l'identification des anomalies. La classification du trafic est effectuée en utilisant des techniques d'apprentissage automatique, tandis que l'identification des anomalies est basée sur le score de silhouette, une mesure de la similarité entre les données. L'apprentissage en ligne est également intégré dans le framework, permettant ainsi une mise à jour continue du modèle de détection en fonction des nouvelles données collectées. Cette approche permet de garantir une détection efficace des attaques zero-day, même dans un environnement en constante évolution

Enfin, cette thèse soulève un problème important dans l'état actuel de la recherche concernant les jeux de données utilisés pour évaluer les modèles de machine learning et de deep learning dans le domaine de la détection d'intrusion. En effet, ces jeux de données sont souvent obsolètes, ne reflètent pas le contexte réel des systèmes d'information et ne sont pas adaptés aux besoins spécifiques des communautés industrielles et scientifiques.

Pour répondre à cette problématique, cette thèse propose un générateur de données de synthèse qui permet de créer des jeux de données récents et adaptés au contexte d'origine, tout en garantissant la confidentialité du système d'information d'origine grâce à un processus d'anonymisation. De plus, il n'est pas limité en termes de scénarios d'attaques, ce qui permet de tester les modèles de détection d'intrusion dans des conditions variées et réalistes et s'appuie sur trois techniques majeures pour valider les données générées. Ce générateur de données de synthèse permet ainsi de renforcer la pertinence et la fiabilité des évaluations des modèles de détection d'intrusion, tout en contribuant à l'avancement de la recherche dans ce domaine crucial pour la sécurité des systèmes d'information.

Acknowledgements

Je tiens à exprimer ma profonde gratitude envers le Tout-Puissant qui m'a donné la force, le courage, l'abnégation, l'endurance et toute l'énergie nécessaires pour mener à bien ce projet et ainsi réaliser un rêve d'enfance.

Après plus de trois années précieuses dans mon parcours personnel et professionnel, je tiens tout d'abord à remercier les deux pères fondateurs de ce projet de recherche. A Antoine Gallais, je lui témoigne toute ma reconnaissance pour tout depuis nos premiers échanges. Robin Giraudo, qui a été un acteur majeur dans mon recrutement et a fortement contribué à mon évolution au sein du SOC d'IBM CIC France, a donné son feu vert dès les premiers échanges pour initier ce projet, je lui témoigne toute ma reconnaissance. Merci à ces deux personnes sans lesquelles ce projet n'aurait jamais existé.

Un projet de recherche implique un encadrement, un suivi et une direction. Je tiens donc à remercier du fond du cœur mes deux directeurs de thèse, Antoine Gallais et Thierry Delot, pour le temps, la disponibilité, les précieux conseils, le suivi et la bienveillance qu'ils m'ont apportés. À Youcef Imine, mon encadrant, je tiens à lui exprimer mes remerciements infinis pour les échanges techniques, l'attention portée aux moindres détails, les conseils quotidiens et la proximité permanente. À ces trois personnes, un simple merci ne suffit pas.

Je tiens à saluer les différentes équipes et personnes avec lesquelles j'ai collaboré au sein d'IBM, tant les équipes et collègues du SOC, qui m'ont accompagné et ont toujours répondu présents, que les autres équipes et personnes pour leur disponibilité constante. À mes collègues de l'équipe Threat OPS qui sont partis ou qui sont toujours là, merci pour ces moments d'échanges, de partage et ces afterworks tous les jeudis: vous êtes les meilleurs. A Alexis Semnont, l'avoir en alternance a été d'une aide précieuse pour l'aboutissement de ces travaux. Je le remercie et le souhaite le meilleur pour l'avenir. Pourquoi pas une continuité de certains travaux initiés?

Je salue toute l'administration et tous mes collègues doctorants et docteurs au sein du LAMIH pour les différents échanges, conseils et activités partagées ensemble. Vous avez été d'un soutien constant au cours de ces trois dernières années: vous êtes au top, l'équipe du LAMIH.

Je remercie tous mes professeurs et encadrants, de l'école primaire à ces études de doctorat. Je ne souhaite pas citer de noms de peur d'en oublier, mais je tiens à leur témoigner toute mon affection, ma reconnaissance et mes remerciements les plus sincères. Si j'aime le monde académique et l'enseignement, c'est en grande partie grâce à eux.

Enfin, je termine là où tout a commencé, mon entourage : mes parents, mes frères et sœurs, ma fiancée. L'attention, l'affection et l'amour, les conseils et la présence de chacun d'entre eux m'ont donné la force et le courage de poursuivre mes rêves et de surmonter les moments difficiles. Dédicace speciale à mon père et ma mère qui m'ont toujours accompagné et soutenu quels que soient mes choix de vie!

"Tu sais mon fils, les voies de Dieu ne sont pas les nôtres. Retiens bien ceci : Les gagnants ne sont pas ceux qui n'ont jamais chuté, ce sont ceux qui se relèvent à chaque fois." Ces belles paroles de M. Claude Mabudu ont été un déclic dans ma vie!

"Dans la vie, rien n'est à craindre, tout est à comprendre.", dixit Marie Curie!

Contents

List of Tables

List of Figures

Publications

Journal articles:

- **A framework for detecting zero-day exploits in network flows**
  Almamy Touré, Youcef Imine, Alexis Semnont, Thierry Delot and Antoine Gallais
  Computer Networks
  https://doi.org/10.1016/j.comnet.2024.110476

International Conference paper:

- **Automated and Improved Detection of Cyber Attacks via an Industrial IDS Probe**
  Almamy Touré, Youcef Imine, Thierry Delot, Antoine Gallais, Robin Giraudo and Alexis Semnont
  38th International Conference On ICT Systems Security and Privacy Protection IFIP SEC 2023, June 14-16, 2023.
  https://doi.org/10.1007/978-3-031-56326-3_14

National Conference paper:

- **Automation and Improvement Detection of Cyber Attacks via an Industrial IDS Probe**
  Almamy Touré, Youcef Imine, Alexis Semnont, Thierry Delot and Antoine Gallais
  Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information, station touristique du Valjoly à Eppe-Sauvage, 2024.

To be submitted:

- **A Detailed Analysis of a Network Information System Synthetic Data Generator: NIS-SDG**
  Almamy Touré, Alexis Semnont, Youcef Imine, Thierry Delot and Antoine Gallais
  The 49th IEEE Conference on Local Computer Networks (LCN) October 8-10, 2024, Caen, Normandy, France.

Chapter 1

Introduction

Contents

1.1 Context

In the modern world, computing has become an essential element of our society, present in all sectors of activity. Digital data has emerged as the driving force of the economy and innovation, and its automated processing is now essential to ensure the competitiveness and sustainability of businesses. However, this growing dependence on computing and digital data has resulted in an exponential increase in cybersecurity risks. Indeed, organizations in all sectors face major challenges in protecting their digital assets and ensuring data confidentiality, integrity, and availability. On the other hand, cyberattacks are becoming increasingly sophisticated and targeted, with potentially severe consequences for victim organizations. In fact, the IBM 2023 Data Breach Report revealed that the average cost of a breach reached 4.45 *million dollars* in 2023, marking a 15% increase over three years [1].

Cyber threats can take many forms, ranging from denial-of-service (DoS) attacks to phishing and ransomware attacks, zero-day vulnerabilities, and supply chain attacks. Attackers can exploit known or unknown vulnerabilities in defense systems to access sensitive data and disrupt business operations [2, 3, 4]. In addition, the detection of these threats poses a significant challenge and must be addressed in order to maintain a secure information system. Another major challenge lies in the daily connection of diverse applications to the Internet and which complicates the management of network traffic within the information system. Moreover, the obfuscation techniques bring with it a diversification of attack techniques targeting these systems [5, 6]. In addition to applications, the attack surface of organizations is constantly expanding with the proliferation of devices connected to the information system and the adoption of new technologies such as cloud computing and the Internet of Things (IoT). Moreover, the increasing complexity of information systems makes it more difficult to detect and prevent cyber threats.

To address these challenges, organizations implement effective cybersecurity measures. Cybersecurity solutions include continuous monitoring of networks and systems, threat detection and prevention, vulnerability management, incident response, and disaster recovery. Security Operations Centers (SOC) play a key role in the implementation and monitoring of these solutions. Indeed, a SOC offers intelligent solutions to help businesses mitigate the significant threats and risks. The main tool in a SOC is the Security Information and Event Management (SIEM) which allows to detect attacks by providing intelligent analysis for quicker threat identification. It relies on events and logs from various equipment (servers, databases, antivirus software, firewalls, etc.) collected for monitoring. In addition to event logs, network activities (also known as flows, which can be encrypted) can be collected by a dedicated probe and centralized in the SIEM tool. However, the latter generates alerts based on pre-packaged and manually established correlation rules. Once a piece of collected data matches a correlation rule, some security alerts are provided. In addition to the SIEM, the SOC relies on Security Orchestration, Automation, and Response (SOAR) tool to automate and accelerate the detection and response to incidents.

1.2 Motivation and research challenges

In our context, attack detection primarily relies on manually created and updated correlation rules, which are managed by expert correlation engineers. This detection approach common to SOC, including IBM CIC France's, can entail significant human costs. It requires substantial engineering effort to analyze detection scenarios and establish rules correlations among the large volumes of collected data by effectively modeling scenarios within detection tools. Thus, there is a vast variety of rules, constantly implemented and updated according to the evolution of attack scenarios. Monitoring tools must necessarily have an up-to-date database of these rules to minimize vulnerabilities in the observed systems. Additionally, this detection approach requires heightened vigilance to monitor and respond to identified alert escalations.

However, several aspects limit the effectiveness of these mechanisms against current attacks. On the one hand, these correlation rules are highly restrictive and do not allow all desired matches to be made in the context of network flow analysis, for example (pattern matching). It is necessary to establish the limits of these rules to better exploit these low-level collection tools to generate offenses that will be analyzed and processed by security analysts. On the other hand, the reduction of false positives is another major challenge for SOC as it significantly impacts the efficiency and reliability of threat detection and response. Indeed, false positives can result in a loss of time and resources for security analysts who must prioritize and investigate security alerts, as well as a decrease in confidence in the security detection system. Therefore, improving the accuracy of threat detection tools is crucial to ensure a timely and effective response to real threats while optimizing the human and technical resources allocated to attacks detection and response.

In addition to known detection scenarios, zero-day attacks are by definition unknown and cannot be detected by traditional security systems. To address this challenge, one approach is to use machine learning and deep learning techniques to analyze system behavior and detect anomalies that may indicate the presence of a zero-day attack. However, the implementation of these techniques is complex and requires advanced skills in computer security and data science. Therefore, proposing new approaches to detect zero-day attacks is an important challenge to improve current detection and response strategies.

Beyond the nature of the attacks (known or zero-day), most of the collected flows are encrypted, which reduces the amount of exploitable information from these data. Thus, network traffic analysis of encrypted payloads is a challenge for SOC as it limits the visibility of the content exchanged, thereby making the detection of potential threats more difficult. To address this challenge, one approach is to rely on the metadata associated with network flows, such as source and destination IP addresses, used ports, packet size and frequency, etc. While this information does not allow for in-depth analysis of the content exchanged, it can be crucial for detecting suspicious or abnormal behavior, such as connections to IP addresses with a malicious reputation or exchanges of unusually sized packets. The use of machine learning and artificial intelligence techniques can also be considered to improve the accuracy and reliability of the analysis of metadata from encrypted network flows.

Furthermore, we note that using these machine learning techniques also requires significant volumes of data for training and evaluating models. Datasets available in the literature are often static, limited in scenarios and scalability, and do not meet the needs for recent synthetic data. Synthetic data generation techniques often rely on these datasets or simulated environments, limiting their realism and ability to represent real threats.

In light of these challenges, improving security incident detection and response strategies requires a proactive and dynamic approach. This entails building more precise detection scenarios, reducing human effort, automating processes through artificial intelligence and machine learning algorithms, reducing false positive rates through appropriate mechanisms, reducing execution time, and detecting approaches. Finally, it is necessary to propose a synthetic data generator to effectively evaluate the proposed attack detection techniques.

Back in 2020, as a security analyst at IBM Security, I had the opportunity to initiate a research collaboration with LAMIH UMR 8201, that started on February 2021. IBM is one of the leading companies in the field of cybersecurity. It offers intelligent solutions to assist businesses in preparing for and mitigating these significant threats and risks. IBM has a SOC in Lille (France), which plays a crucial role in preventing, detecting and responding to cyber threats. This thesis project aims to address some of the challenges related to the enhanced detection and response to security incidents.

1.3 Our contributions

In this section, we introduce the various scientific contributions proposed within the scope of this thesis in response to the challenges raised.

1.3.1 Automated attack detection for known scenarios

Given the diversity of approaches proposed for intrusion detection via IDS probes and the tendency of existing solutions to cater to specific contexts, cross-scenario model evaluation becomes challenging. To address this, we present an approach for detecting and classifying malicious network flows based on the main properties of network flows which includes:

- A standardized and scalable engineering process for extracting relevant feature classes essential for any network flow classifier.
- A deep learning model that integrates our feature extractor with a CNN-based feature detector to achieve high-performance classification.

- A comprehensive evaluation of our model on different datasets from various contexts, validated with the MITRE ATT&CK framework.
- A comparative analysis between our model, traditional machine learning methods, and deep learning solutions outlined in the current literature.

Our solution outperforms existing solutions like [7, 8, 9, 10] in terms of classification metrics and execution time.

1.3.2 Zero-day attack detection

Given the unknown nature of zero-day attacks, characterized by the absence of signatures and patterns for detection, many IDS solutions leverage learning techniques to build novel attack detection systems. However, these solutions often prioritize accuracy enhancement for specific attack types, overlooking the potential for multiple attack scenarios. In response, we propose an approach for detecting zero-day attacks. Therefore, our contributions to this zero-day detection framework, compared to existing works [11, 12, 13, 14, 15, 16] in the field, revolve around the following key points:

- Hybridization of classification techniques: Integrating supervised and unsupervised classification techniques is essential. The utilization of supervised classification models such as CNN aids in categorizing known network flows, while employing unsupervised algorithms like K-Means unveils concealed patterns in the data.
- Precision optimization through combined learning methods: The approach emphasizes algorithms fusion, combining the prowess of supervised CNN-based methods with decision trees (DT), random forests (RF), K-Nearest-Neighbors (KNN) and Naive Bayes (NB). This cross-breeding aims to maximize the detection model's precision, offering a more accurate and dependable insight into potential threats, without solely focusing on hyper-parameter's optimization.
- Correlation between supervised and unsupervised results : Creating a correlation table between supervised classification outcomes and clusters formed by the unsupervised approach aids in associating clusters with attack classes or normal flow categories.
- Outlier-based anomaly identification for zero-day detection: Identifying outliers within clusters formed by the K-Means algorithm and using this data to create new potential attack classes enables proactive zero-day attack detection. This facilitates real-time model updates through online learning, adapting the model regularly to new data, thereby enhancing its capability to detect and respond to zero-day attacks in real-time.
- Validation of results on two datasets: One derived from an industrial context containing real flows and recent attacks, and the other from the constantly used state-of-the-art dataset for intrusion detection and zero- day detection mechanism validation.

1.3.3 Synthetic data generation

Existing datasets proposed in the literature suffer from limited size and nature of attack types, as well as potential biases due to their collection and lack of diversity and evolutionary nature. While synthetic data generation techniques, typically relying on these outdated datasets, may sometimes lack realism or introduce artificial biases into the generated data. We propose a synthetic data generation approach that:

- Ensures realistic synthetic data generation while preserving the confidentiality of the original information system data.
- Provides reliable training data for attack detection systems, using a multi-phase solution including collection, anonymization, activity profiling, and behavior analysis.
- Ensures various evaluation and validation methods of the generated data, such as profiling, statistical analysis, and the use of discriminant models.

1.3.4 Impacts of contributions

Our first significant contribution enhances the intrusion detection and cyberattack strategy by introducing an innovative methodology for classifying network traffic. Our solution, grounded in a comprehensive approach, ensures optimal feature extraction by leveraging the intrinsic properties of network flows and employing convolutional neural networks. This method sets itself apart from traditional intrusion detection solutions that rely on machine learning and deep learning algorithms on standard datasets. One of the major advantages of our solution lies in reducing model execution time, made possible by decreasing the number of attributes to be processed. Additionally, our approach demonstrates remarkable adaptability, capable of implementation regardless of a specific context, which is a considerable asset.
Our methodology also aims to reduce the complexity of network flow classification models and algorithms while preserving their effectiveness. This approach is intended to be easily integrable into any environment or information system, thereby offering appreciable flexibility in implementation.

Our second contribution, aiming to detect zero-day attacks, stands out for its innovative nature and enhances existing approaches to attack detection. By employing cross-validation of both supervised and unsupervised models, our method significantly reduces the false detection rate. The introduction of an anomaly detection technique based on silhouette score in clustering, validated by online learning, represents a notable advancement in scientific literature. The proposed solution is not specific to a particular class of attacks but rather aims to detect any anomaly in network traffic. This approach enables the identification of new attack scenarios, thus broadening the scope of our detection method. Lastly, the regular updating of the knowledge base renders our model scalable and adaptable, ensuring its applicability in any information system. This essential characteristic ensures the sustainability and continuous effectiveness of our solution in the face of the constantly evolving cybersecurity threats.

Our third and final contribution addresses a major issue in the current scientific community. We present an innovative synthetic data generator, distinguished by its techniques of extraction, synthesis, anonymization, and collaborative dimension in the field of intrusion detection. This generator can be used locally to generate data and simulate models, or collaboratively to benefit from a wide variety of data. Our solution enables the extraction of up-to-date, non-simulated real datasets that are scalable and applicable in any information system. Furthermore, its collaborative dimension encourages the continuous enrichment of the database with diverse attack scenarios that are representative of real-world situations.
In summary, this contribution offers an innovative and adaptable approach to address current challenges in synthetic data generation while fostering collaboration among various stakeholders in the cybersecurity domain.

1.4 Outline of the manuscript

The structure of this thesis is outlined as follows. Chapter 2 provides an overview of operational security within an industrial setting, delving into IBM's array of tools designed to bolster cyberattack detection and response. Chapter 3 addresses the formidable challenges encountered in threat detection and response. Here, we conduct a comprehensive examination of both traditional and contemporary approaches, identifying key issues and obstacles associated with each. Additionally, we introduce pertinent datasets and discuss data generation techniques prevalent in the literature.Our novel approach to automating and enhancing attack detection in industrial environments is detailed in Chapter 4. In Chapter 5, we unveil our framework tailored for the detection of zero-day attacks. The culmination of our research journey is encapsulated in Chapter 6, where we present our groundbreaking contribution to synthetic data generation. Chapter 7 serves as the conclusive chapter of this thesis, wherein we offer reflections on our findings and outline prospective avenues for future research endeavors.

Chapter 2

Background of security operational in an industrial context

Contents

2.1 Introduction

In this chapter, we provide a comprehensive overview of operational security, encompassing the technological tools and means employed in the industrial context for cyberattack detection, specifically focusing on the solutions offered by IBM.
Initially, we delve into the fundamental principles of our research, encompassing the understanding of network flows and flow collectors, events and log sources. Furthermore, we scrutinize the MITRE ATT&CK framework, an indispensable instrument for comprehending and categorizing attack tactics and techniques. Subsequently, we investigate the cybersecurity tools employed for attack detection and response within an industrial context. Lastly, we present the Artificial Intelligence tools proposed by IBM, with a focus on their application in use case proposals, model training, and data generation.

2.2 Fundamentals of network security

In this section, we address the essential prerequisites for data collection for intrusion detection. We particularly focus on understanding network flows and asset activity logs, which are fundamental data types in this context.

2.2.1 Network flows - Flow collector

Network flows, representing the movement of data packets between network endpoints, play a crucial role in understanding and analyzing network behavior. Network

flows encapsulate the communication patterns within a network, providing valuable insights into the interactions between devices, users, and services [17]. These flows are characterized by various attributes, including source and destination IP addresses, ports, protocols, and timestamps. Understanding the dynamics of network flows is fundamental for detecting anomalous activities and potential security threats.
As presented by Li et al. in [18], network flows manifest in diverse forms, each furnishing unique insights into network behavior and security posture. Common types of network flows entail:

- **NetFlow:** Developed by Cisco, NetFlow furnishes information regarding IP network traffic, encompassing source and destination IP addresses, ports, and protocols, according to the RFC 3954. In [19], the use of NetFlow data is widely discussed for network traffic analysis and security monitoring.
- **IPFIX:** IP Flow Information Export (IPFIX) constitutes an IETF standard founded on Cisco's NetFlow version 9 [20]. It improves NetFlow by proffering extensibility and support for additional data types, streamlining interoperability across diverse network equipment and vendors.
- **sFlow:** In the RFC 3179 [21], sFlow is a sampling-based monitoring technology that captures network traffic data at wire speed. It offers scalable and efficient monitoring capabilities, rendering it suitable for large-scale network environments.
- **JFlow:** is a flow monitoring technology developed by Juniper Networks [22]. Similar to NetFlow and sFlow, JFlow captures data on network traffic flows. It enables Juniper users to gain comprehensive visibility into their network traffic patterns and security posture. Although a proprietary protocol, it is widely supported by network monitoring tools and security solutions.

In [23], Zhou et al. discussed the importance of collecting network data via flow collectors. These collectors serve as specialized systems designed to capture, aggregate, and analyze network flow data. These collectors intercept network traffic and extract flow information, enabling comprehensive visibility into network activities. By efficiently collecting and processing flow data, organizations can gain real-time awareness of network behavior and proactively identify suspicious or malicious activities. In [24], So-In et al. highlighted the continuous expansion of networks and the growing need for monitoring and analysis tools. Administrators face the critical task of not only detecting and promptly addressing network failures, but also preemptively averting potential disruptions due to network overload or external threats. For instance, analyzing network utilization and traffic patterns can effectively uncover security vulnerabilities.

Flow collectors employ various analysis techniques to distill actionable insights from network flow data [18]. These techniques encompass: Behavioral Analysis, Signature-based Detection, Anomaly Detection, Machine Learning - Deep Learning Detection Techniques, described in Section 3.3. Flow collectors are frequently integrated with existing security infrastructure, encompassing IDS, IPS, and SIEM platforms.

2.2.2 Events - Log source

An event log is an electronic record of a significant activity that has occurred on a computer system or network. These events are recorded chronologically in log files, also known as event logs, to enable monitoring, analysis, and reverse engineering of actions that have taken place [25]. Event logs are crucial for cybersecurity incident detection

and response. By monitoring event logs, it's possible to spot suspicious activities, detect security breaches, and respond swiftly to threats. Additionally, event logs are used for auditing, regulatory compliance and reverse engineering of past security incidents, and essential for forensic analysis.

The event logs can originate from various sources, such as operating systems, applications, network devices (firewalls, routers), proxies, antivirus software, security sensors, etc. as well as any type of asset within an information system. Each log source provides specific information about system or network and application activities, and their integration enables a holistic view of security posture and either utilizes syslog as their default logging format or offer features enabling conversion to syslog format [26]. Syslog, widely adopted across log sources, provides a basic yet versatile framework for log entry generation, storage, and transfer. It facilitates normalization of logs from diverse sources, enabling centralized storage and analysis by a unified system. As a standard protocol, syslog efficiently delivers standardized system messages across networks, detailed in RFCs 3164 and 3195 [27, 28].
Table 2.1 presents examples of data sources, corresponding event types and concrete examples of recorded activities.

Source	**Types of Events**	**Examples of Events**
Operating System	System startup	User "admin" login session start
	User login	Modification of system configuration file
	File system changes	
Applications	Application errors	Failed database connection
	Access to sensitive data	Unauthorized access to sensitive data
Network	Intrusion attempts	Port scanning detection
	Network traffic flows	Suspicious data transfer between machines
Firewalls	Connection blocking	Blocked connection attempt from suspicious IP
	Security rule violations	Violation of defined firewall rule
Routers	Configuration changes	Routing table modification
	Routing events	Network interface configuration change notification
Proxies	Activated content filters	Blocking access to malicious website
	Website access blocks	Activating a filter to block suspicious outbound traffic
Antivirus Software	Virus/malware detections	Quarantine of infected file
	Virus definition updates	Successful virus definition updates
Security Sensors	Detection of abnormal activities	Detection of suspicious movements within monitored area
	Security alerts	Alert of unauthorized access attempt

Table 2.1: Various types of event logs.

Standards and regulations such as the GDPR, ISO/IEC 27001, and PCI DSS compliance directives mandate the collection and analysis of event logs to ensure data protection and computer system security [29]. The GDPR requires close monitoring of activities to comply with data breach notification requirements. Similarly, ISO/IEC 27001 necessitates continuous surveillance to detect and address security vulnerabilities. Furthermore, PCI DSS directives mandate log analysis to detect and prevent credit card transaction-related fraud. By adhering to these standards, organizations bolster their security posture and effectively safeguard sensitive data against threats.

2.2.3 Framework MITRE ATT&CK

In the previous sections, we introduced activity logs and network flows, which can be utilized for attack detection. In this section, we will present the MITRE ATT&CK framework, which plays an indispensable role in the strategy for detecting these attacks.

The MITRE ATT&CK (Adversarial Tactics, Techniques & Common Knowledge) framework is a universally accessible and continuously updated knowledge base used to model, detect, anticipate, and mitigate cybersecurity threats based on known adversarial behaviors of cybercriminals [30]. Developed by the MITRE Corporation, a nonprofit organization, the MITRE ATT&CK framework is an indispensable reference for the prevention and detection of attacks.

As presented in [31], this framework comprehensively catalogs the tactics, techniques, and procedures (TTP) employed by cybercriminals at each phase of the cyber attack life cycle, from initial information gathering and planning behaviors to the final execution of the attack. The information contained within MITRE ATT&CK can guide security teams in various aspects: accurately simulating cyber attacks to test cyber defenses; developing more effective security policies, security controls, and incident response plans; selecting and configuring security technologies for better detection, evasion, and mitigation of cyber threats.

The `Enterprise` framework currently consists of 14 tactics and over 230 techniques, including nested sub-techniques. In Table 2.2, we present the 14 tactics, along with their associated TTPs, the Advanced Persistent Threats (APT) groups that most frequently exploit these tactics, and the required data sources.

By leveraging the collected activity logs and network flows, the MITRE ATT&CK framework provides a structured approach for analyzing, identifying, and categorizing potential cyberattacks.

Tactic	TTP Examples	Associated APT Groups	Data Sources
Reconnaissance (TA0001)	Scan IP blocks, Conduct social engineering attacks, Search public websites	APT29, APT32, Fancy Bear	Network event, application logs, public websites
Resource Development (TA0002)	Purchase compromised infrastructure, Create malicious websites, Steal credentials	Carbanak, FIN7, Silence	Threat intelligence, open-source data, malware analysis
Initial Access (TA0003)	Exploit software vulnerabilities, Send malicious attachments	Sandworm, Lazarus, Emotet	Security patches, known vulnerabilities
Execution (TA0004)	Inject malicious code, Execute malicious commands, Drop and execute malware	REvil, BlackCat, Locky	Security tools, EDR, memory analysis
Persistence (TA0005)	Create scheduled tasks, Modify the registry, Add malicious code to startup scripts	Turla, APT34, Conficker	Windows registry, scheduled tasks, persistence tools
Privilege Escalation (TA0006)	Exploit software vulnerabilities, Use social engineering, Abuse legitimate credentials	Cobalt Strike, Mimikatz, Hydra	Privilege escalation tools, known vulnerabilities, malware analysis
Defense Evasion (TA0007)	Disable security software, Hide malicious files and processes, Use anti-debugging techniques	WannaCry, NotPetya, Ryuk	Security tools, EDR, malware analysis
Credential Access (TA0008)	Brute-force passwords, Steal credentials from memory, Use social engineering	Raccoon, IcedID, TrickBot	Password managers, phishing tools, malware analysis
Discovery (TA0009)	Enumerate network shares, Query Active Directory, Search for sensitive files	APT41, APT28, Winnti	Network analysis tools, Active Directory auditing tools, file search tools
Lateral Movement (TA0010)	Pivot through compromised systems, Use remote access tools, Exploit network vulnerabilities	Cyclops Blink, PowerSploit, PsExec	Network security tools, EDR, malware analysis
Collection (TA0011)	Steal sensitive files, Exfiltrate data over the network, Capture screenshots	Cloakedhopper, Barium, APT33	Network security tools, EDR, network traffic analysis
Command and Control (TA0012)	Communicate with a remote server, Receiving commands from a remote server, Sending data to a remote server	BEACON, Sality, Gh0st	Network security tools, EDR, network traffic analysis
Exfiltration (TA0013)	Exfiltrate data over the network, Send data to a remote server, Upload data to a cloud storage service	Cobalt Strike, Dridex, Zeus	Network security tools, EDR, network traffic analysis
Impact (TA0014)	Deny service to a system, Disrupt business operations, Steal financial data	Mirai, WannaCry, CryptoLocker	Security tools, application logs, incident reports

Table 2.2: MITRE ATT&CK Tactics, Techniques, Associated APT Groups, and Data Sources.

2.3 Attack detection and response products : IBM ecosystem

Cybersecurity in industrial context is a critical frontier in today's security landscape, tasked with protecting vital infrastructure and operational technology systems from evolving cyber threats [32].
Several solution providers offer recognized tools in the cybersecurity market, including Checkpoint, Microsoft, Palo Alto, C, etc. In this section, we provide an in-depth examination of the key detection and response tools within our working context (IBM Security Operation Center), where the application of our research findings will be explored.

- **QRadar SIEM:** IBM's QRadar SIEM plays a pivotal role in empowering security teams to proactively address today's threats in real-time. Going beyond mere threat detection and response, QRadar SIEM equips analysts with artificial intelligence tools for user behavioral analysis, a robust threat correlation engine, threat intelligence feeds, and a plethora of customizable applications for automation and seamless integration with solutions from other vendors. QRadar operates on both cloud-native and on-premise architectures and has consistently been positioned as a leader in the Gartner Magic Quadrant for SIEM reports for over a decade. This sustained leadership underscores its advanced capabilities, adaptability, and effectiveness in meeting the evolving security needs of enterprises [33].

- **QRadar Network Insight - NDR:** IBM QRadar NDR, or Network Detection and Response, is a solution designed to address the evolving challenges posed by sophisticated cyber threats in today's network infrastructures. By integrating several technologies (Flow Collector, QRadar Network Insights, Network Threat Analytics, Network Packet Capture, Incident Forensics), QRadar NDR enables customization of detection and response capabilities at the network level of information systems [34].

  QRadar NDR is capable of detecting a wide range of threats, including lateral movement, data leakage, advanced threats, and compromised assets. It consolidates various telemetry sources such as network flow data, full packet analysis, advanced machine learning-based analytics, threat intelligence, and AI-powered investigations into a single, cohesive platform.

  This solution integrates with IBM Security QRadar SIEM and IBM Security QRadar Security Orchestration, Automation, and Response (SOAR), ensuring comprehensive detection and incident response across on-premises, cloud, and hybrid environments. Thus, QRadar NDR ensures organizations rapid identification and mitigation of threats.

- **QRadar EDR:** IBM Security QRadar EDR is an active defense intelligence platform that automatically detects and responds to threats on endpoints. Using a behavioral detection approach, QRadar EDR identifies both known and unknown threats and monitors application abuse to detect potential security risks. It triggers alerts based on abnormal behavior in running processes and switches to deep monitoring mode to collect additional events, such as file and registry operations, user activities, while conserving resources. QRadar EDR operates seamlessly in disconnected

or offline environments and comprises endpoint agent, server, and dashboard components. The AI-powered agent operates in both online and offline modes, safeguarding endpoints and telemetry data even without network connectivity. QRadar EDR seamlessly integrates with QRadar SIEM and other SIEM products, enabling ingestion of EDR events and alerts for comprehensive threat management [35].

- **QRadar SOAR:** IBM Security QRadar SOAR (Security Orchestration Automation and Response) is an advanced solution designed to expedite security incident response. Through intelligent automation and orchestration, it enhances SOC efficiency by reducing response times and addressing skill gaps [36].

  Tailored to meet the critical needs of organizations facing escalating threats, QRadar SOAR streamlines incident management by automating repetitive tasks and guiding analysts through personalized workflows. Offering an intuitive interface and dynamic protocols, QRadar SOAR enables swift adaptation to evolving incident conditions. Furthermore, QRadar SOAR facilitates compliance with over 180 international regulations pertaining to privacy and data breaches, ensuring incident response adheres to prevailing standards.

  Key features include automated alert correlation and enrichment, orchestration of incident responses via customizable playbooks, and seamless integration with existing tools and technologies in the organization's security environment.

  With IBM Security QRadar SOAR, organizations can bolster their security posture, mitigate cyberthreat risks, optimize human resource utilization, and ensure a swift and effective response to security incidents.

Figures 2.1, 2.2 present the IBM SOC design architecture and its reference model.

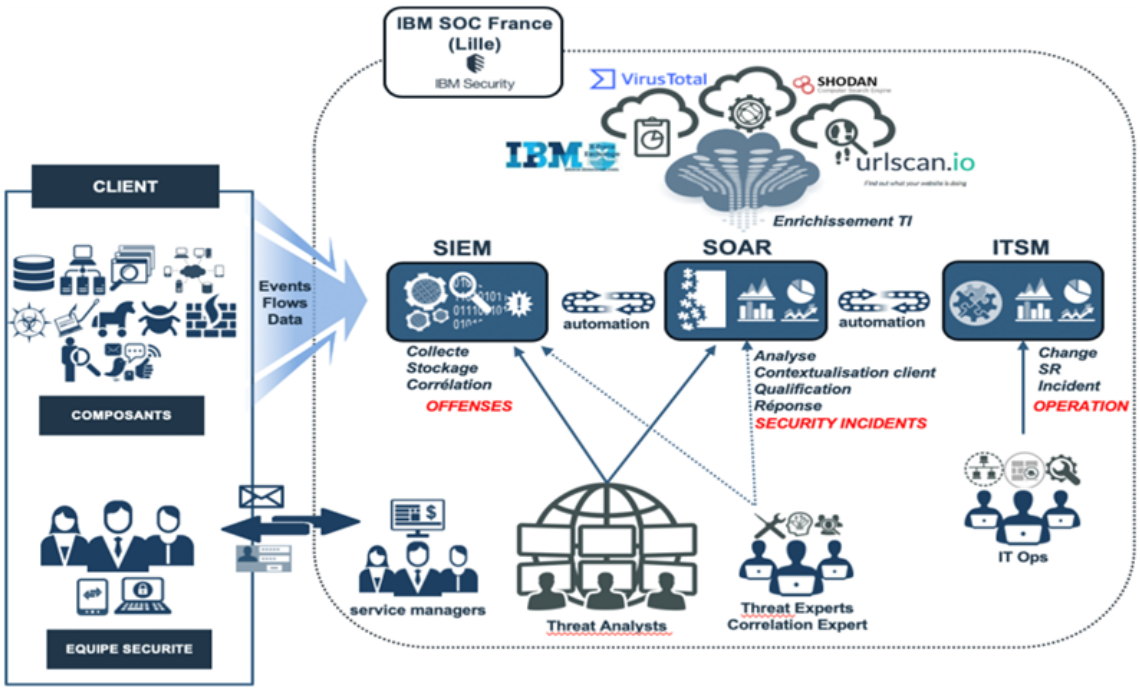

Figure 2.1: IBM SOC - Design.

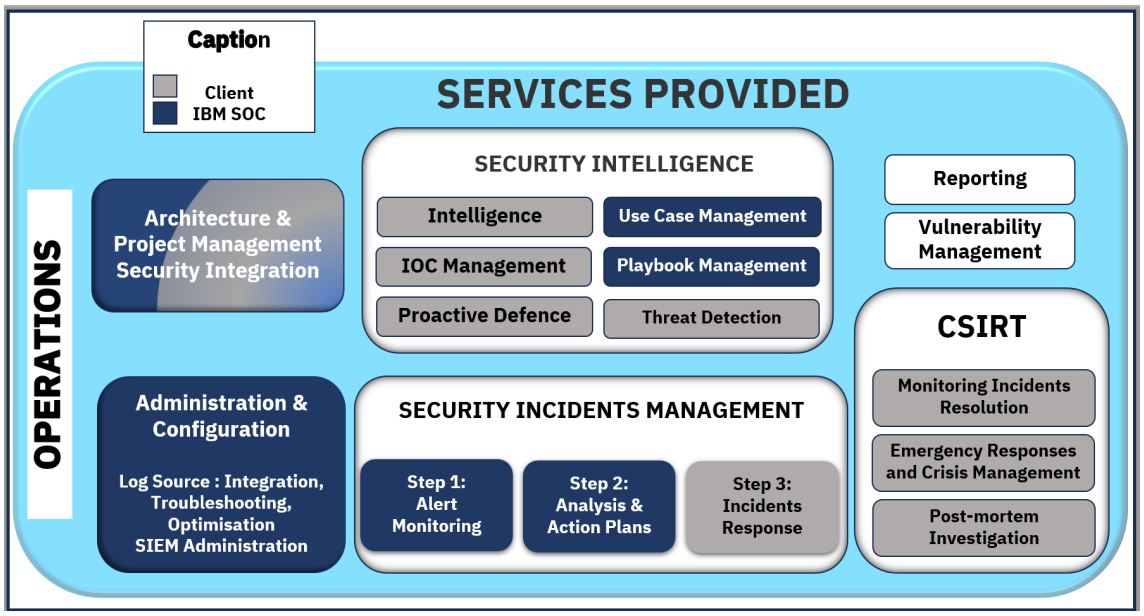

Figure 2.2: IBM SOC - Reference Model.

2.4 IBM generative AI products

In this section, we present two generative AI solutions from IBM, which can be used for use cases in detection, incident response, and data generation:

- **Watsonx.ai:** is an artificial intelligence (AI) platform developed by IBM that provides a comprehensive solution for building and deploying AI models. The platform combines traditional machine learning (ML) techniques with advanced AI model generation capabilities powered by foundation models [37]. Watsonx.ai also offers a variety of pre-trained models developed by IBM, open-source, and third-party, as well as the ability to build custom models [38]. The platform stands out for its end-to-end AI governance approach, which enables enterprises to scale and accelerate the impact of AI with reliable data across the business, using data wherever it resides. This approach includes privacy and data security tools, as well as model management features to ensure transparency, fairness, and regulatory compliance.

  In terms of research, Watsonx.ai offers interesting possibilities for exploring new AI approaches, particularly in the areas of AI model generation, transfer learning, and federated learning. The platform also provides tools for building and customizing performant prompts, which can be useful for researchers working on conversational AI or natural language processing applications.

- **Watsonx.data:** Watsonx.data is a data storage platform developed by IBM, providing access to all data through open formats and integration with existing databases and modern tools.
  This comprehensive data storage platform enables researchers and data scientists to easily access all their data, prepare data for advanced machine learning and generative AI use cases, and unlock new data insights through a generative AI-powered conversational interface. Hybrid deployment options also offer flexibility to meet organizations' data governance and security needs [37, 39].

2.5 Conclusion

In this chapter, we have established the essential prerequisites for attack detection and emphasized the importance of the MITRE ATT&CK framework for data collection and exploitation within an information system.
We then presented security incident detection and response tools, focusing on the solutions provided by IBM, which form the basis of our research. We also discussed generative artificial intelligence tools and their functioning.
In the next chapter, we will examine the major challenges businesses face in terms of cybersecurity. We will present a state of the art on cyberattack detection and the use of synthetic data to improve threat detection and response capabilities.

Chapter 3

Network security challenges & state-of-the-art

Contents

3.1 Introduction

In this chapter, we delve into the varied challenges that companies encounter in terms of cybersecurity. Among these essential challenges, we will emphasize detection and response to security incidents. After identifying these issues, we will review the research and approaches proposed for detecting attacks, whether they are known or zero-day. We will examine both traditional methods and newer approaches.

Furthermore, we will also address the datasets used to evaluate attack detection models along with data generation techniques

3.2 Major challenges

Businesses today face critical challenges in terms of cybersecurity, due to the significant increase in their assets. It is imperative for them to carefully examine each of these

assets in order to protect themselves against cyber threats aimed at paralyzing their information systems. It is with this in mind that we will present, in this section, the major challenges they face. We will start with the preservation of confidentiality and integrity of data, moving on to identity and access management, obsolescence and vulnerability management, as well as compliance with standards and laws in force. We will pay particular attention to the detection and response to security incidents.

3.2.1 Data privacy and integrity

In an environment where cyber threats evolve rapidly and attackers exploit often subtle vulnerabilities to compromise systems and networks, data privacy [40] and integrity [41] are two interrelated aspects posing significant challenges in cybersecurity [42].
With the proliferation of information exchange across networks, preserving data privacy has become a complex challenge. Cyber attackers deploy a plethora of techniques to intercept data in transit, thereby compromising the confidentiality of communications. Attacks such as data interception, packet sniffing, and network eavesdropping represent serious threats to the confidentiality of information exchanged between legitimate users. Additionally, data leaks resulting from poor system configurations or human errors can also compromise data privacy, exposing organizations to considerable risks in terms of reputation and regulatory compliance [43].
Maintaining data integrity is as crucial as preserving data privacy. Cyber attackers often target data integrity by modifying or tampering with legitimate information to sow confusion, inflict damage, or bypass security mechanisms. Attacks such as code injection, sabotage, packet manipulation, and distributed denial-of-service (DDoS) can compromise data integrity, undermining trust in systems and processes reliant upon it. Furthermore, data processing errors, software bugs, and hardware failures can also lead to unauthorized data alterations, jeopardizing the integrity of critical information [44].

In the context of our work, data privacy and integrity hold paramount importance. Indeed, Nova et al. emphasize that the ability to identify and respond to threats largely depends on the quality and reliability of collected and analyzed data [45]. Thus, it is important to develop methods and techniques that not only detect attacks but do so while preserving the confidentiality of sensitive information and ensuring the integrity of manipulated data.

3.2.2 Identity and access management

In the IBM X-Force Threat Intelligence 2024 report [46, 47], the weakest links for European organizations were identified as identities and emails, with the abuse of valid accounts and phishing being the most common causes of attacks in Europe. Furthermore, they note that one of the three most significant challenges for organizations based in Europe is the collection of identification information (28%).
Therefore, Identity and Access Management (IAM) represents a major challenge for businesses due to the increasing complexity of IT environments. As the number of devices, applications, and users continues to grow, managing access to sensitive data and systems becomes more challenging.
Cyber-attackers use a multitude of techniques to impersonate identities in order to access sensitive data, thereby compromising the security of communications. Attacks such as identity spoofing, based on the initial access and credential access tactics of the MITRE matrix, pose serious threats to the security of information exchanged between legitimate users. Moreover, information leaks resulting from inadequate system configurations or human errors can also compromise identity and access management, exposing organiza-

tions to significant risks in terms of reputation and regulatory compliance.

As presented by Ghaffari et al. in [48], mechanisms such as multi-factor authentication, PKI and biometric identifiers, as well as Role-Based Access Controls (RBAC), least privilege principles and identity federation are increasingly widespread and used by organizations. However, these organizations face challenges such as malicious insiders, Cool Boot attacks, weak password reset vulnerabilities, phishing and malicious redirects, replay attacks, identity theft, cross-site request forgery, cross-site scripting, and Trojan horse susceptibility.

With the rise of cloud computing, enterprises are increasingly adopting this technology. In [49], Indu et al. presented identity and access management in a cloud environment as also a major challenge for organizations due to the complexity and vulnerability of cloud computing. Challenges related to features such as multi-tenancy and third-party managed infrastructure in a cloud environment require special attention for effective identity and access management to ensure the security of cloud data and resources.

To address these challenges, companies must adopt a comprehensive approach to IAM that includes identity verification, access control, continuous monitoring, regulatory compliance, and cloud security. This approach should also consider the use of emerging technologies such as artificial intelligence and machine learning to improve identity and access management capabilities. In addition, organizations should establish security policies and procedures, provide regular security awareness training to employees, and conduct periodic security audits to ensure the ongoing effectiveness of their IAM systems.

3.2.3 Obsolescence and vulnerability management

Industrial legacy systems and software are often vulnerable to cyberattacks due to the absence of security patches. In fact, the rapid evolution of industrial environment where technologies and IT infrastructures are constantly changing, the efficient management of obsolescence and vulnerabilities has become a crucial challenge. First, this challenge involves identifying, assessing, and addressing obsolete or vulnerable elements within a company's technological infrastructure. This will allow to prevent potential security breaches and maintain optimal levels of performance and reliability [50].
Several factors contribute to the complexity of managing [51]:

- Difficulty in tracking and mapping IT assets: The rapid evolution of IT environments can make it difficult to accurately identify and track all software and hardware assets, as well as their end-of-life status.
- Rapid proliferation of technologies and software within a company creates a major challenge in terms of tracking and managing obsolete versions. This can lead to security gaps, incompatibilities, and increasing complexity in maintaining the entire system.
- Software vulnerabilities represent potential entry points for cyberattacks, requiring constant monitoring and proactive action to identify and correct them before they are exploited.
- Migration and upgrade costs: Migrating to new systems or upgrading existing software versions can represent significant financial and operational costs for organizations.
- Dependence on suppliers: The availability and cost of spare parts for obsolete hardware systems can be problematic, creating a dependence on suppliers and increasing the risks of service interruption.

Managing obsolescence and vulnerabilities is therefore a major challenge for the cybersecurity of businesses. A proactive and structured strategy is essential to mitigate the risks associated with end-of-life IT assets and ensure the resilience of information systems against cyberattacks. This strategy should include regular assessment of IT assets, constant monitoring of vulnerabilities, planning for migration and upgrading of systems, as well as effective management of relationships with suppliers [52]. By meeting these challenges, companies can maintain their competitiveness and security in a constantly evolving digital environment.

3.2.4 Compliance and regulation

Within a digital landscape marked by a proliferation of regulations and standards in data protection, compliance has become a significant challenge for businesses. This challenge entails the implementation and maintenance of data management practices that comply with progressively stringent legal and regulatory requirements [53].

One of the biggest cost amplifiers of data breaches in 2023 was the non-compliance with regulations [1]. Several factors contribute to the complexity of compliance management, including the proliferation of regulations. This has been caused by several factors [54, 55], such as:

- Complexity of regulations: Regulations can be complex and varied depending on the geographic region, industry, and business sector of the company. It is crucial to understand and interpret these regulations correctly to ensure that all requirements are met.
- Data management and privacy: Data protection regulations impose strict requirements for the collection, storage, and processing of personal data of customers and employees. Compliance with these regulations requires appropriate policies and security measures to protect individuals' privacy.
- Risks of non-compliance: Companies face financial, legal, and reputational risks in case of non-compliance with current regulations. Fines, lawsuits, and loss of customer trust can have severe consequences for the viability and reputation of the business.

To address these challenges, companies must establish comprehensive and integrated compliance programs, involving close collaboration between compliance teams, legal departments, IT services, and key stakeholders [56]. These programs should include regular compliance assessments, training to raise employee awareness of legal requirements, as well as monitoring and reporting mechanisms to ensure ongoing compliance. By adopting such a posture, companies can mitigate the risks associated with non-compliance, strengthen their competitive position, ensure personal data protection, maintain customer trust, and avoid financial penalties and reputational damage.

3.2.5 Threat detection and response

The detection of cyber attacks requires a systematic and rigorous collection, centralization, and correlation of data from various systems and networks to identify anomalies that may indicate malicious activity.

In an interconnected digital environment where numerous access points, networks, and data sources coexist, the ability to effectively correlate data streams becomes crucial. Data correlation involves the fusion and the analysis of several datasets to identify patterns, anomalies, and potential indicators of malicious activity. However, achieving this

integration and analysis seamlessly is complex and fraught with challenges [57].
In [58], various points to threats detection and response cyber are highlighted by Khan et al.:

- The large volume and diversity of data generated by security systems pose a challenge in terms of correlation. Also, diverse data formats can create compatibility and interoperability issues between correlation tools. The lack of context and information on threats and vulnerabilities can limit the ability to identify potential security incidents.
- Modern cyberattacks are often complex and multidimensional, requiring advanced correlation techniques to detect, while traditional correlation methods may not be sufficient to identify sophisticated attacks. Correlation tools can also generate a large number of false alerts, overwhelming security analysts and delaying the detection of real incidents. Furthermore, analysts tiredness can reduce their effectiveness and ability to identify critical threats.
- Temporal correlation is necessary to identify events that may appear insignificant when taken individually but reveal attack patterns or suspicious behavior when considered in a broader temporal context. Additionally, data correlation may raise privacy concerns, particularly when personal data is involved, and thus it requires ethical and legal data collection and processing. Finally, correlation systems must be scalable to handle variable workloads and constantly evolving environments while maintaining high performance.

Security Information and Event Management (SIEM) systems can play a crucial role in enhancing the correlation capabilities of security systems [59, 60]. In Figure 3.1, we present the architecture proposed by Podzins et al. for collection from various data sources, correlation, and detection using a SIEM tool [61].

Faced with these challenges, our research focuses on developing methodologies to strengthen the correlation of collected data to enhance cyberattack detection and response capabilities. By adapting data correlation techniques to the nuances of cyber threat landscapes, our aim is to build the resilience of digital infrastructures against malicious intrusions. However, attack detection faces some major issues:

1. **Data processing pipeline for ML-based intrusion detection**

   While machine learning (ML) and deep learning (DL) techniques are widely used in attack detection, their effectiveness is often limited by the quality and preparation of the data used for training and evaluating the models [62]. In Figure 3.2, Maharana et al. present a overview of the data pre-processing steps.

   Firstly, effective preprocessing methods are crucial to prepare data for model training. This may include handling missing values, data normalization, noise reduction, and variable transformation. Secondly, managing class imbalances is a common challenge in imbalanced datasets, where one class is over-represented compared to another. This situation can lead to misleading and biased model accuracy, requiring techniques such as oversampling, undersampling, or synthetic data generation to balance the classes [63]. Additionally, selecting relevant features is a crucial step to improve model performance. Data may contain redundant, irregular, or irrelevant features that can affect model accuracy and efficiency. Proper feature selection can help reduce data dimensionality, improve model interpretability, and reduce

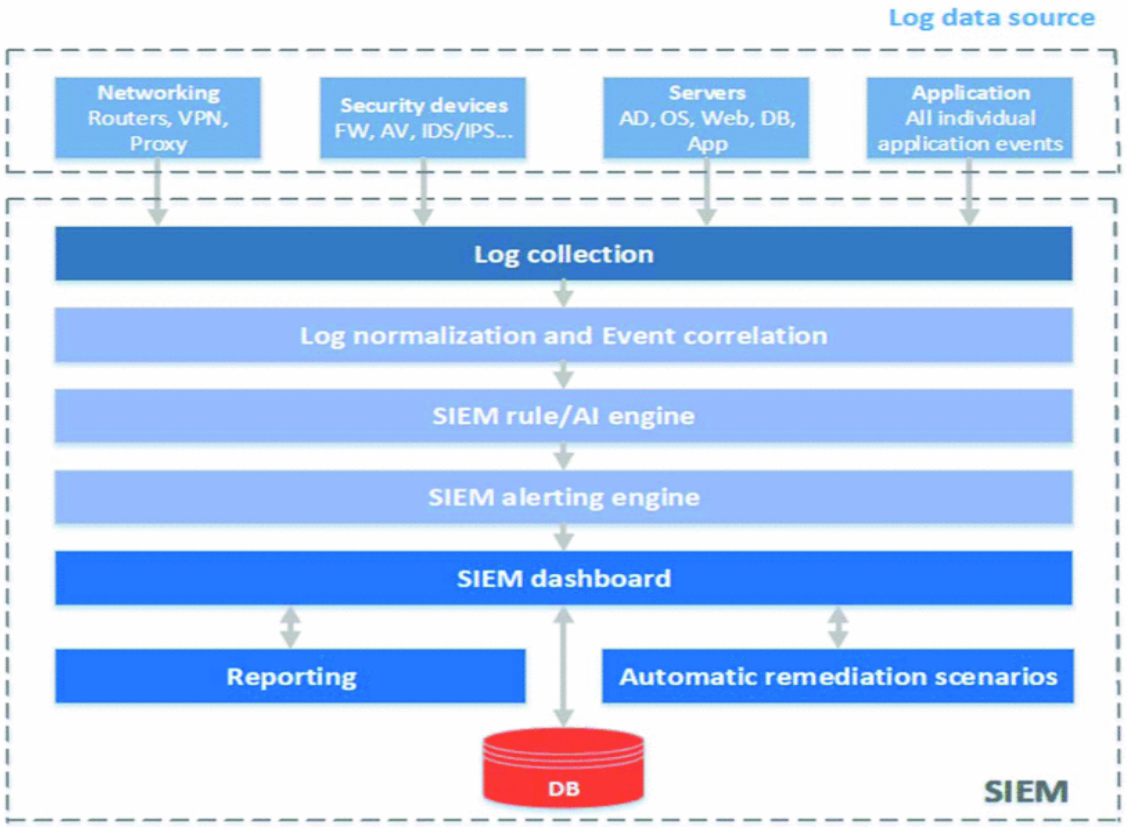

Figure 3.1: SIEM Architecture proposed by Podzins et al.

Problems with data collected from real-world		
Too much data	**Too little data**	**Fractured data**
- Feature extraction - Irrelevant data - Substantial data sizes - Categorical/Numerical mixed data - Noisy data	- Missing attributes - Missing values - Less amount of data	- Incompatible data - Data collected from multiple sources - Data from various levels of granularity

Data Pre-processing Techniques		
Data Transformation	**Information Gathering**	**Generating new Information**
- Data Ordering - Data Modelling - Filtering, Aggregation and Summarization - Indexing and Ordering - Enrichment and Imputation - Modelling, Typecasting, Formatting, and Renaming	- Data Sampling - Data elimination - Data selection - Data visualization	- Data engineering - Time series analysis - Data fusion - Data augmentation - Dimensional analysis - Constructive induction

Data Analysis
- Knowledge extraction - Prediction - Modelling systems - Building knowledge-based system

Figure 3.2: Challenges with data collection and processing.

computation time. Finally, optimizing model hyperparameters is an important step to enhance model performance. Additionally, significant attention must be devoted to the security and confidentiality of data used in the model training process [64].

Optimizing data processing for intrusion detection presents a major challenge. This involves exploring best practices in data preprocessing, cleaning, and labeling, as well as selecting the most relevant features to ensure precise outcomes during model training. Additionally, it is essential to reduce model execution time to enhance organizational responsiveness to cyber threats. This issue represents a critical concern for strengthening the effectiveness and reliability of intrusion detection systems.

2. **Automation of attacks detection: known and zero-day**

   Given the fast evolution of cyber threats and their potential impact on information systems, the effective detection of suspicious activities on computer networks is of paramount importance. This task involves more than just recognizing previously documented attack patterns; it also requires the ability to track and identify new zero-day threats that exploit unknown vulnerabilities. Additionally, there are several major challenges associated with the automation of known and zero-day attack detection, including reducing the volume of alerts, addressing performance and scalability issues, and accounting for data obfuscation techniques or the use of non-standard protocols [65, 66].
   In our examination of current solutions, we scrutinize both traditional methods and recent advancements proposed in specialized literature on cyberattack detection. Special emphasis is placed on automating attack detection, particularly for new attacks that have never been detected (also known as zero-day attacks), with the aim of developing advanced methodologies and tools to detect and respond to threats proactively and swiftly, thereby minimizing potential harm to computer systems [67].

   In this regard, various issues must be considered:

   - Effectively leveraging recent advancements in the fields of artificial intelligence (AI), machine learning, deep learning, and behavioral modeling.
   - Creating systems capable of learning normal activity patterns and accurately identifying anomalies that may indicate an ongoing attack.
   - Developing robust detection techniques for zero-day attacks, which are inherently challenging to identify due to their novel and unpredictable nature.

   Addressing these challenges is essential for improving the reliability and security of data management systems

3. **Generating synthesis data for ML model assessment**

   The effectiveness and reliability of Machine Learning (ML) and Deep Learning (DL) models largely depend on the quality and diversity of available data for their training and evaluation [68]. However, the current datasets used pose significant limitations: they are static, outdated, and often constrained in terms of size, diversity, and representativeness of attack scenarios [69].
   Moreover, real-world data available may also be subject to limitations, such as incompleteness, bias, or limited availability. These factors compromise the models' ability to generalize effectively and reliably detect new threats.

To address these challenges, synthetic data generation emerges as a promising solution. This approach enables the creation of artificial datasets that faithfully replicate the characteristics of real data while providing full control over simulated scenarios and attack types. Consequently, researchers and practitioners have access to diversified and balanced datasets to train, test, and evaluate the performance of ML models in cyberattack detection.

Developing realistic datasets that depict authentic scenarios and evolve over time in response to cyberattacks is imperative today [69]. The primary challenge lies in enhancing the quality and diversity of available data for cyberattack detection, as well as for the validation of ML and DL models.

3.3 Literature review for attack detection

In this section, we delve into the heart of attack detection techniques. As cyber threats evolve in complexity and sophistication, it becomes imperative to understand the various approaches used to detect and counter these attacks. We will thus explore the primary detection paradigms, ranging from classical methods such as signature-based detection, to more advanced approaches like anomaly detection, machine learning and deep learning. Additionally, we will address the distinction between known attacks and zero-day attacks, emphasizing the crucial importance of developing effective strategies to confront these emerging threats. By examining existing literature, we will highlight recent advancements, persistent gaps, and promising directions in the field of attack detection, thereby laying the necessary groundwork for our own contribution in this ever-evolving domain.

3.3.1 Traditional attack detection techniques

In this section, we present the classical, also known as traditional, techniques used for intrusion detection. Three primary families are highlighted in the state-of-the-art: signature-based, anomaly-based, and payload-based.

Signature-based approach:

Signature-based detection represents one of the earliest documented approaches in the literature for attack detection, emerging in the 1970s in the form of Signature-Based Intrusion Detection Systems (IDS). This method relies on creating specific signatures or patterns corresponding to known attack patterns. These signatures are then used to analyze network traffic or system activities, searching for matches with suspicious behaviors. Signature-based IDSs were among the first effective tools for detecting well-known attacks such as Denial of Service (DoS) attacks or password compromise attempts [70].
Signature-based detection is essentially a string matching problem, meaning it continues to search for a pattern or substring within a collection of patterns or large strings [71]. Various techniques have been proposed in the state of the art:

- **Pattern matching approaches:** A first variant of this approach involves comparing signatures of suspicious packets or files with a database of known signatures. Fast search algorithms such as the Knuth-Morris-Pratt (KMP) [72], Boyer-Moore (BM) [73], Wu-Manber [74] and Aho-Corasick (AC) [75] algorithms are commonly used for this purpose. These methods are effective for detecting attacks.
  To enhance these algorithms for a better detection, Aldwairi et al. introduced a novel algorithm named Exhaust, derived from the Wu-Manber algorithm. Exhaust

cleverly employs Bloom filters as exclusion filters to minimize costly searches in the hash table, a bottleneck in detection systems [76, 77]. Through preprocessing and optimized search processes, Exhaust demonstrates improved performance and efficient resource utilization, meeting the escalating cybersecurity demands without compromising speed or accuracy. Innovative approaches aim to optimize pattern search processes in order to address the increase in malicious behaviors and malware. One such method, proposed in [78], develops a new pattern search algorithm that strategically avoids positions not matching the desired pattern. Test results exhibit superior performance compared to established algorithms like Aho-Corasick and Boyer-Moore, indicating significant potential for enhancing intrusion detection system efficiency.

Another approach, [79], proposes a hybrid solution of two models. Firstly, parallelization of the Direct Matching Algorithm (PDMA) is achieved using OpenMP technology in a multicore architecture for faster pattern searching. Then, PDMA is applied in Network Intrusion Detection System (NIDS) engines to enhance detection speed. This solution has shown a significant improvement in performance compared to traditional sequential algorithms, with reduced processing time and improved NIDS engine performance compared to the current SNORT-NIDS engine.

- **Sequence analysis approaches:** This method aims to detect specific patterns of malicious activities by analyzing sequences of packets or events. Models such as finite automata or Markov chains are used to identify malicious behaviors, enabling the detection of attacks based on event sequences rather than specific signatures. In a recent study [80], Papadogiannaki et al. propose an innovative solution to address the growing challenge of intrusion detection in encrypted networks. This approach introduces a signature language that focuses on packet metadata, enabling the detection of characteristic packet sequences of intrusions. Using pattern mining algorithms, expressive signatures are automatically generated to identify intrusion attempts efficiently, even in encrypted environments.

- **Automatic signature generation approaches:** Some signature-based detection mechanisms have enabled the automatic generation of signatures for new patterns or schemes. For example, in [81], a method is presented for detecting polymorphic worms using a graph-based classification framework for polymorphic worm signatures, accompanied by a new signature method called Conjunction of Combinational Motifs (CCM). Tongaonkar et al. [82] present an automated system for detecting new application signatures for traffic classification purposes. In this work, the authors present a system for automatically identifying keywords of unknown applications. In [83], a mechanism is presented for botnet C&C signature extraction. The mechanism identifies frequent strings in traffic and then ranks the frequent strings based on traffic clustering methods.

In summary, we note that signature-based detection is a rapid and accurate method for identifying known attack patterns by searching for matches with predefined models. While it offers advantages in terms of speed and precision, it presents significant limitations, including the risk of false positives, the frequency of updating the knowledge base, and the difficulty in detecting new zero-day attack patterns [84]. Additionally, attackers' obfuscation techniques can bypass this detection method. Thus, while valuable for combating known threats, signature-based detection requires improvements to address the challenges posed by emerging and sophisticated attacks.

Anomaly-based approach:

Anomaly-based intrusion detection is a method that focuses on identifying non-conforming behaviors or activity patterns within a computer system. In contrast to signature-based approaches, which rely on recognizing patterns of known attacks, anomaly detection seeks to pinpoint aberrant behaviors that may indicate malicious activity. The emergence of this approach in the literature of computer security dates back to the 1980s and 1990s, when researchers began exploring methods to detect intrusions based on abnormal behaviors rather than specific signatures. This approach was motivated by the growing recognition that cyber-attacks were evolving rapidly and that traditional detection methods were limited in their ability to identify new threats. Anomaly-based intrusion detection relies heavily on machine learning and statistical modeling to create a knowledge base of normal behavior profiles for the users, applications, or systems. These profiles are then used to detect significant deviations from expected behaviors, which may indicate suspicious activity [85].
In the literature, various techniques are proposed:

- **Statistical anomaly detection:** This technique involves establishing a baseline of normal behavior using statistical metrics such as mean, median, standard deviation, or frequency distributions. Any deviation from this baseline is flagged as potentially malicious. Statistical anomaly detection methods are effective in identifying deviations from normal behavior.

  In [86], Zhen et al. introduce the STP (STatistical Pattern-based feature extraction method) for host-based anomaly detection. This innovative approach transforms system call sequences into n-gram frequency sequences and applies a predefined set of statistical features to construct an anomaly detection model. Experimental findings reveal that STP outperforms other methods in terms of stability and performance, although it does not consistently achieve the highest AUC score. Moreover, cross-platform evaluation demonstrates STP's effectiveness in combining samples from different platforms, thereby enhancing anomaly detection performance. Notably, STP stands out for its computational efficiency, offering promising insights into optimizing feature selection algorithms and enhancing training samples for improved anomaly detection in diverse environments.

  In [87], the Variable Type Detector (VTD) is proposed for anomaly detection in textual computer log data. It identifies variable types within log lines, such as chronological, static, ascending or descending, continuous, range, unique, and discrete. Utilizing the Kolmogorov-Smirnov goodness of fit test, it selects variables for anomaly detection based on data type stability. Additionally, the VTD introduces an indicator function to reduce false positives. Evaluation using diverse datasets shows its superiority over traditional methods like time series analysis and principal component analysis. It enhances intrusion detection in SIEM and EDR systems and supports UEBA. Future work includes refining assessments of continuous data types, introducing time series data types, and deploying the VTD in real-world scenarios.

  Additonnaly, in [88], Friedberg et al. propose a method for detecting Advanced Persistent Threats (APTs) through log-line analysis. Their solution learns system's normal behavior to detect deviations, automatically generating a system behavior model based on event relationships across multiple network machines. This enables the detection of APT-related anomalies like direct accesses to database servers or logging functionality disablement. Evaluation shows effective anomaly detection,

although individual anomalies may not be detected by a single rule. Approximately 40% of defined rules detect each injected anomaly, ensuring detection, despite a relatively high false positive rate. However, logging deactivation is consistently detected, highlighting the system's ability to identify abnormal activities, even if not linked to APTs. The model exhibits a high false positive rate and requires manual intervention for deep causal analysis. Future work may involve developing a more intelligent approach to reduce redundant hypotheses and alleviate model overload.

- **Protocol anomaly detection:** Protocol anomaly detection focuses on identifying deviations from expected network protocol specifications. It involves parsing network traffic to detect anomalies in protocol headers, sequences, or payload structures. Protocol anomaly detection is effective in detecting novel attacks that exploit protocol vulnerabilities.

  In [89], Bao et al. introduce a packet header anomaly-based anomaly detector, an experimental protocol for modeling network and host intrusion detection systems. This detector analyzes the behavior of packet header field values across layers 2, 3, and 4 of the OSI network model. Named the Packet Header Anomaly Detection (PbPHAD) system, it is designed to detect anomalous behavior in network traffic packets, focusing on specific protocols such as UDP, TCP, and ICMP at the network and transport layers. Its goal is to identify the degree of maliciousness based on a set of anomalous packets detected from statistically modeled abnormal field values. They also highlights the model's performance on a standard evaluation dataset and proposes the incorporation of expert rules.

  In [90], a solution for detecting attacks on Wi-Fi networks is presented through a Wireless Intrusion Detection System (WIDS). This approach relies on behavioral analysis of anomalies, where the normal behavior of the Wi-Fi protocol is modeled using n-grams and machine learning models to classify Wi-Fi traffic as either normal or malicious. The system comprises two main modules: the capture module (Sniffer), which collects Wi-Fi traffic, and the behavioral authentication module (BAM), which analyzes this traffic by converting packets into flows, extracting n-grams, and using machine learning models to classify flows as normal or abnormal. The WIDS has undergone comprehensive testing on various datasets, demonstrating precise detection of all Wi-Fi protocol attacks, with low false positive rates and the capability to detect even minimal footprint attacks such as Minimal Deauthentication.

- **Stateful inspection:** Stateful inspection monitors and analyzes the state and context of network connections to detect anomalies. It tracks the sequence of events within a connection and compares it against expected behavior. Stateful inspection is effective in detecting complex attacks that span multiple network sessions.

  In [91], Parvat et al. propose an innovative approach for anomaly and intrusion detection in networks, with a particular focus on packet-filtering firewalls. Their method combines the use of Deep Packet Inspection (DPI) to identify suspicious behaviors with the analysis of packet record attributes to enhance IDS performance. By leveraging multi-core and multi-threading processing capabilities, this hybrid approach ensures accurate and rapid attack detection while minimizing false alarms

In [92], Togay et al. present an innovative solution for detecting intrafirewall policy anomalies, focusing primarily on packet-filtering firewalls. With the increasing number of connected devices and the growing complexity of firewall rules, policy anomalies are becoming more common, compromising network security. The proposed solution relies on a logic programming-based anomaly detection model, accompanied by a corresponding software tool. This framework transforms existing firewall configurations into actionable knowledge to automatically detect anomalies, offering an efficient and scalable solution for firewall maintenance and security. This approach enables efficient scaling for anomaly detection, providing improved performance even with large rule sets.

The anomaly-based intrusion detection approach provides a valuable alternative to signature-based detection methods by identifying non-conforming behaviors or activity patterns within a computer system. It is widely used in, behavioral analysis, detecting zero-day or sophisticated attacks that evade signature-based detection. However, this approach presents some significant limitations:

- False positives: Statistical anomaly detection methods generate false positives in dynamic environments where normal behaviors can vary significantly.
- Encryption issues: Protocol anomaly detection may be limited by encryption or obfuscation of network traffic, making it difficult to detect attacks in such cases.
- Scalability: Stateful inspection may encounter scalability issues in high-speed networks due to the complexity of monitoring and analyzing numerous simultaneous connections.
- Computational resources: Different approaches may require sophisticated algorithms and significant computational resources to effectively detect malicious activities, posing challenges in terms of performance and costs.

Payload-based detection:

Intrusion detection by payload, also known as 'payload-based detection,' is a method used to identify intrusion attempts by deeply analyzing the content of data packets exchanged over a network. This method delves into the actual content of transmitted data to detect signatures, patterns, or anomalies indicative of malicious activity.
The emergence of this approach in specialized literature stems from the growing need to detect sophisticated attacks that may be concealed within the content of network communications. Researchers have developed techniques to analyze and extract meaningful information from payloads, leading to a variety of payload-based intrusion detection approaches and methods:

- **Content pattern analysis:** This technique involves searching for specific patterns within packet content to detect known attacks. Effective pattern matching methods such as regular expressions or trie trees are used for this task. While precise for detecting specific patterns, it can be evaded by signature variants or obfuscations.

  In [93], an approach is proposed to address a performance issue in intrusion detection, where existing algorithms are complex and slow in handling large amounts of data generated during system or network monitoring. Hence, Kuri et al. introduced a pattern-matching-based intrusion detection filter. This method relies on the concept of insertion distance, treating an attack as a sequence of events and detecting text

portions where all attack events appear in order within a window of k other events. They demonstrated that the filter can rapidly identify suspicious areas in audit logs by scanning millions of events per second, thereby reducing processing overhead for more sophisticated algorithms. Their approach was evaluated using an analytical model and preliminary experiments, showing its effectiveness in efficiently filtering text while maintaining good attack detection.

In [94], an innovative solution for network intrusion detection is presented. This solution, named C_MPM, relies on a multi-pattern matching algorithm based on characteristic values. Unlike traditional approaches, C_MPM simplifies rule tree construction and reduces the number of comparisons by classifying rules based on their length and characteristic value. The algorithm performs parallel analysis of characteristic values of text strings in the matching window, optimizing pattern detection and reducing false positives. Experimental results demonstrate that C_MPM offers better temporal performance, with an exponential decrease in processing time as string length increases.

- **Semantic analysis:** This technique examines packet content at the semantic level to detect malicious behaviors, utilizing techniques such as syntax or semantic analysis to extract meaningful information from data. While capable of detecting sophisticated attacks, it may be limited by the complexity of semantic analysis and language variability in packets.

  In [95], Wu et al. propose an aggregated flow-based inspection method to amplify the features of malicious behavior in network traffic. They introduce a new data analysis approach for efficiently classifying network traffic by utilizing a topic model to construct a doc-word matrix from statistical features, and then analyze latent semantic information to determine whether an aggregated flow is malicious. The proposed technique is evaluated using CIC-IDS2017, UNSW-NB15, and NSL-KDD datasets.

Similar to other traditional detection approaches, data encryption poses a major obstacle to in-depth packet content analysis, thereby reducing the effectiveness of this method in detecting attacks concealed within encrypted communications. Furthermore, the increasing complexity of data transmitted over networks makes packet content analysis challenging and prone to errors. Attackers exploit this complexity by constantly developing new techniques to circumvent detection systems, such as polymorphic attacks or data obfuscation, thereby making payload-based detection less reliable. Additionally, the massive volume of data generated by networks presents challenges in terms of the time and resources required for analysis, potentially leading to delays in threat detection. Lastly, the possibility of generating false positives due to unusual legitimate behaviors underscores the need for payload-based detection solutions to incorporate sophisticated alert validation and correlation mechanisms to ensure accurate detection of malicious activities while minimizing undesirable alerts.

Issues and challenges

In light of the persistent challenges encountered by traditional detection techniques such as signature-based, anomaly-based, and payload-based methods, exploring new detection mechanisms has become imperative. While these conventional approaches have played a pivotal role in combating known threats and non-conforming behaviors, they face significant hurdles in effectively identifying zero-day attacks, sophisticated threats, and encrypted traffic. Challenges such as false positives, scalability constraints, and high

computational resource requirements underscore the need for novel detection techniques. Thus, in order to conduct more comprehensive analyses of network traffic patterns, swiftly identify malicious behaviors with enhanced precision, and proactively adapt to emerging threats without relying on constant updates to signature databases or anomaly rules, various research endeavors have embraced machine learning approaches.

3.3.2 Automatic learning techniques

The constant evolution of cyber threats demands perpetual adaptation of detection methods. Among modern approaches, the utilization of machine learning (ML) and deep learning (DL) has shown promising performances in early and accurate attack detection. ML and DL-based attack detection rely on the analysis of complex mathematical models capable of identifying patterns and anomalies within vast datasets. Essentially, this approach aims to automatically learn from training data to discriminate between normal and malicious behaviors in testing data. ML and DL algorithms leverage techniques such as neural networks, decision trees, and ensemble methods to accomplish this task.
Thus, machine learning algorithms enable learning from historical data and automatically detecting patterns, anomalies, and malicious behaviors in network traffic or system activities:

Supervised learning:

These algorithms are trained on annotated data, where examples are labeled as either normal or malicious. They learn to recognize distinctive features of attacks and can subsequently detect them when encountering new examples in test data. Recent research efforts have explored various enhancements and adaptations of these supervised methods. Numerous studies compare the performance of different supervised machine learning algorithms for intrusion detection. The following paragraphs summarize some of the key findings in this area.

Ensemble models such as Bagging, Boosting, and Stacking appear to outperform classical models such as K Nearest Neighbors (KNN), Support Vector Machines (SVM), or Random Forests (RF). Other works indicate higher performance with decision trees or neural networks. In [96], Jabbar et al. introduced a classification solution using alternating DT, an algorithm that creates prediction and separation nodes. This approach, applied to the NSL-KDD dataset, inspired further research aiming to optimize hyper-parameters and enhance accuracy rates while exploring different algorithms. Additionally, in [97], the authors employed the NB algorithm and its implementation with PCA on the NSL-KDD dataset. Similarly, in [98], Koc et al. applied the Hidden NB (HNB) to the KDD Cup 99 dataset, combining attribute independence with a binary HNB classifier to reduce naivety.

Adversarial learning approaches increasingly challenge ML-based detection systems. Techniques to evade ML-based attack detectors using the RF algorithm have been proposed, highlighting the sensitivity of ML algorithms, particularly RF, to parameter changes and feature order [99]. In [100], the authors conducted an analysis on the NSL-KDD dataset, exploring the performance of ten classification algorithms. Their approach to feature selection relies on attribute evaluators and filtering techniques. The implemented algorithms included Naive Bayes, Bayes-Net, Logistic Regression, Random Tree, Random Forest, J48, Bagging, OneR, PART, and ZERO. Among these, Random Forest emerged as the top-performing classifier, achieving an accuracy of 99.9% with a false alarm rate as low as 0.001. Following closely behind was Bagging, exhibiting an accuracy of 99.8%. Notably, the PART algorithm demonstrated a similar level of performance.

In [101], the authors propose a revised method that consolidates three attack categories into one through supervised learning. This approach aims to simplify the classification procedure and enhance the accuracy of network intrusion detection. By employing ensemble-based feature selection methods, including LightGBM, Random Forest, and ExtraTrees, this study seeks to identify a minimal set of relevant features for network intrusion detection without compromising accuracy. The results of the experiment showed that the Random Forest model achieved an accuracy of 93.3% with an F-score of 24.7. The ExtraTrees model obtained an accuracy of 93% with an F-score of 19.1. However, the LightGBM model achieved the highest accuracy of 95.4%, with an F-score of 61.2.

In [102], an analysis compares the effectiveness of supervised learning models SVM and Random Forest in anticipating future developments in host-based intrusion detection systems. Using datasets collected from various web sources, the analysis reveals that SVM outperforms Random Forest in terms of precision (95.89% versus 94.12%). These findings underscore the advantage of SVM methods in intrusion detection, without observing significant differences between the groups.

Ensemble models, decision trees, and Random Forest have shown promising results, but adversarial learning approaches pose a challenge to ML-based detection systems. Feature selection methods can improve the accuracy of network intrusion detection, and SVM methods may have an advantage in host-based intrusion detection. Other research using DL algorithms focuses on analyzing temporal sequences and extracting distinctive patterns in network traffic. These studies primarily explore the use of Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), or hybrid architectures combining these two approaches.

Deep Learning (DL) is an extension of Machine Learning (ML) principles that processes large datasets using hidden layers for in-depth learning [103]. As with ML algorithms, most attack detection approaches proposed with DL models focus on increasing accuracy and reducing false positives. The following paragraphs discuss various studies that have utilized DL models for attack detection.

One approach to maximize detection success is to combine supervised learning algorithms with feature selection methods. In a study conducted by [104], artificial neural network (ANN) based machine learning with wrapper feature selection outperformed support vector machine (SVM) technique in classifying network traffic. The evaluation was conducted on the NSL-KDD dataset, employing both SVM and ANN techniques. Similarly, in [105], Zhao et al. proposed a sequential classifier using Artificial Neural Network (ANN) to decrease false positives over the KDD99 dataset. Another approach is to implement a multi-layer approach for complex feature extraction to enhance ANN efficiency in Network Intrusion Detection Systems (NIDS). However, varying accuracies across datasets highlight the necessity for more universally applicable solutions. For instance, in [7], Vinayakumar et al. implemented a multi-layer approach for complex feature extraction to enhance ANN efficiency in NIDS. However, the accuracy varied across datasets (KDDCup 99 and UNSW-NB15), highlighting the necessity for more universally applicable solutions.

Efforts in attack detection research also encompass aspects such as reducing execution times, preventing model overlearning, and optimizing hyper-parameters. In [106], Sivamohan et al. combined Random Forest (RF) algorithms with Principal Component Analysis (PCA) for feature extraction from the CICIDS2017 dataset and utilized Bi-directional LSTM (BiLSTM) for attack detection, accelerating computations with GPUs. Similarly, in [107], Wang et al. employed Convolutional Neural Networks (CNN) and LSTM for spatial and temporal feature extraction, although experiencing decreased accuracy for un-

balanced data. Azizjon et al. used a CNN and LSTM combination to serialize TCP/IP packets for supervised learning on the UNSW-NB15 IDS dataset, utilizing CUDA GPU acceleration to reduce computational time [108].

In [109], the authors leverage recurrent neural networks (RNN) with long short-term memory (LSTM) to enhance the accuracy and performance of intrusion detection systems. Their methodology involves several steps: firstly, they test different activation functions for LSTM cells and the number of iterations for backpropagation (BP). Subsequently, the proposed model is evaluated on the UNSW-NB18 dataset, conducting both binary and multiclass classifications. Experimental results show a significant increase in accuracy compared to existing RNN algorithms, with a maximum accuracy reaching 99.96%. Moreover, the model exhibits efficient memory usage and requires a limited number of iterations to achieve optimal performance. This approach also simplifies preprocessing of network traffic data, offering high accuracy and low resource consumption.

The choice of the optimal algorithm depends on the context and datasets used. Several studies highlight the importance of feature selection to enhance the performance of supervised intrusion detection models. Techniques such as symmetric mutual information, attribute evaluators and filtering, as well as correlation analysis and Chi-square, help reduce data dimensionality and improve classification accuracy.

While supervised ML and DL offer promising solutions for intrusion detection, several challenges need to be addressed:

- Data Imbalance: Network traffic data often exhibits an imbalance, with normal traffic significantly outnumbering attack traffic. This imbalance can bias the learning process and require specific techniques like oversampling or undersampling.
- Feature Selection: Selecting the most relevant features from network traffic data can significantly improve model performance and efficiency.
- Evolving Attacks: Attackers constantly develop new techniques. Supervised ML models need to be adaptable and regularly retrained with new data to maintain effectiveness against evolving threats.

Unsupervised learning:

Unsupervised learning algorithms are mostly used in intrusion detection to detect anomalies in the data without the need for labeled examples. These algorithms are presented with unlabeled data and must discover underlying structures or patterns without explicit label supervision. In intrusion detection, unsupervised learning algorithms are particularly useful when network traffic examples lack explicit classifications as normal or malicious.

One approach to unsupervised intrusion detection is the use of robust autoencoders (RAE). In [110], Kotani et al. introduce an approach that leverages RAEs to segregate training data into benign and abnormal features in an unsupervised manner. This method is evaluated on the MAWI dataset, where it demonstrates an ability to reduce false positives and effectively detect attacks, particularly SYN port scans.

Another innovative method for detecting anomalies in network traffic using unsupervised machine learning techniques is proposed in [111] by Alam et al. They develop two models: the Pseudo-AE model, which incorporates skip connections to enhance a convolutional neural network (CNN) and mimic the behavior of an autoencoder (AE), and the more traditional CAE (Classical CNN AutoEncoder) model, consisting of an encoder and

a decoder for input reconstruction. These models are trained and tested on the PVAMU-DDoS2020 dataset, showing satisfactory performance with low error rates (0.007 for the Pseudo-AE model) and high accuracies (0.98 for the CAE model). However, the Pseudo-AE model stands out for its simplicity and shorter training time, thus offering an effective solution for anomaly detection in network traffic.

Unsupervised algorithms play a pivotal role in research endeavors, particularly in clustering tasks. Among these, K-Means is extensively employed to categorize network traffic data into distinct clusters based on inherent similarities [112, 113, 114, 115]. Instances of traffic that deviate significantly from established clusters are flagged as potential anomalies or intrusions. In addition, outlier detection techniques pinpoint data points that statistically differ from the majority, effectively highlighting unusual network activity patterns indicative of attacks.

Techniques such as Principal Component Analysis (PCA) have been utilized to reduce the dimensionality of network traffic data while retaining essential information. This facilitates efficient visualization and anomaly detection in high-dimensional datasets. In [116] Khaoula et al. apply this technique with K-Means on the NSL-KDD dataset and achieve an overall accuracy of 94.51%.

However, unsupervised learning for intrusion detection also presents several challenges:

- High False Positive Rates: Unsupervised methods can flag normal network behavior as anomalies, leading to a high number of false positives that require further investigation.
- Difficulty in Anomaly Interpretation: Identifying the root cause of anomalies detected by unsupervised models can be challenging due to the lack of explicit labels associated with the anomalies.
- Tuning Sensitivity Thresholds: Setting appropriate sensitivity thresholds for anomaly detection is crucial to balance the trade-off between false positives and missed attacks.

Semi-supervised and hybrid techniques:

Semi-supervised learning is a technique that combines elements of supervised and unsupervised learning. In this framework, a small amount of data is labeled, while the majority of data remains unlabeled. This approach is used in intrusion detection to leverage both labeled and unlabeled data to build robust detection models. The application of semi-supervised learning techniques in intrusion detection presents significant opportunities to enhance the accuracy of existing detection systems and to identify emerging threats.

In the state of the art, various use cases of this approach are proposed [95, 117, 118]. For example, Zheng et al. introduce a Self-train framework for fine-grained network-based intrusion detection using deep learning. This semi-supervised detection framework incorporates the RI-1DCNN base model for enhanced feature extraction, an uncertainty-based pseudo-label filtering method, and a hybrid loss function for better class separation and more compact intra-class features [119]. Experimental results on the NSL-KDD and CICIDS-2017 datasets demonstrate that SF-IDS significantly improves attack classification performance, even with only 1% labeled samples, achieving accuracy rates of up to 99%.

In [120], Wang et al. propose ATSAD (Attention-Based Semi-supervised Anomaly Detection), a semi-supervised approach to anomaly detection that builds upon the Deep

SAD algorithm. The key innovation of ATSAD is its use of an attention network to focus on important features of the input data. By defining a new objective function, the attention network and the mapping network of Deep SAD are jointly trained. Evaluations conducted on the CSE-CIC-IDS2018 dataset show that ATSAD outperforms Deep SAD and other methods in terms of performance and robustness. Notably, ATSAD requires fewer labeled samples to achieve better performance, with the authors emphasizing the reduced pollution rate of their solution to just 1%.

Other applications of semi-supervised learning are explored in the literature, particularly in cloud and virtualized environments. However, despite their potential advantages, semi-supervised learning techniques also pose significant challenges. The noise in unlabeled data can compromise the quality of pseudo-labels and model performance, the selection of the most appropriate semi-supervised learning model for a given context and available data can be complex, and the interpretability of semi-supervised models may lead to higher false detection rates compared to supervised models.

These solutions facilitate classification, the differentiation between legitimate and malicious flows, and the identification of attack classes within the model. However, they do not ensure zero-day attack detection.

3.3.3 Zero-day detection approaches

According to IBM Security, a zero-day exploit is a cyberattack vector that takes advantage of an unknown or unaddressed security flaw in computer software, hardware or firmware. A zero-day attack is when a malicious actor uses a zero-day exploit to plant malware, steal data or otherwise cause damage to users, organizations or systems [121].
Faced with the unknown nature of these attacks, it is essential to scrutinize appropriate detection techniques. One approach to achieve is by using linear transformations to simplify and reduce the dimensionality of data, which contributes to improved anomaly detection sensitivity and specificity [122]. This technique can improve the sensitivity and specificity of anomaly detection, as the extracted features can better capture differences between normal and abnormal data. By combining linear transformation and anomaly detection techniques with known attack signatures, it is possible to effectively detect zero-day attacks.
One example of this approach is presented in [123], where different discriminant functions are used to calculate the estimated probability of zero-day attacks by analyzing network connection features. The detection of zero-day attacks is done by identifying anomalies using a defined anomaly score and the KNN algorithm. The proposed model has been tested on the NSL-KDD dataset.

Other research efforts have relied on auto-encoders for zero-day attack detection. Auto-encoders are neural network architectures used for data compression, dimensionality reduction, and data generation [124, 125, 126]. An implication of an auto-encoder is presented in [13]. A high recall rate and a low false negative rate have been defined with mean square error (MSE) thresholds. Additional unsupervised techniques involve local detection modules to aggregate detected anomalies [11]. Botnet detection approaches by Blaise et al. focus on identifying significant changes in port usage, utilizing statistical measures to detect anomalies over time, evaluated on two datasets: MAWI and UCSD. In [127], the authors propose an innovative method for network intrusion detection using robust autoencoders (RAEs) in an unsupervised framework. They convert feature vectors into images and apply unsupervised deep learning techniques, including Bidirectional Generative Adversarial Networks (BiGANs), to extract meaningful features. This approach is evaluated on four public datasets (CICIDS2017, UNSW-NB15, NSL-KDD,

and Bot-loT), demonstrating an accuracy improvement of up to 8.25% compared to deep learning models applied to the original feature vectors. The results showcase the method's capability to reduce false positives and effectively detect attacks, while outperforming features extracted by conventional network analysis tools.

Hybrid approaches are also proposed in the state of the art. In [128], Pu et al. introduce an intrusion detection method that combines Sub-Space Clustering (SSC) and One Class Support Vector Machine (OCSVM). By integrating subspace clustering techniques, SSC forms clusters from small subspaces of the original dataset, while OCSVM, an extension of SVM, handles unlabeled data to detect anomalies. The authors evaluated their solution using NSL-KDD and compared it to three state-of-the-art unsupervised solutions.

To overcome the challenge of obtaining large labeled datasets, Mbona et al. have adopted semi-supervised techniques, proposing an innovative method for detecting zero-day attacks in networks [129]. Their approach focuses on analyzing network traffic features to identify potential anomalies. They have also introduced the use of Benford's law to select significant features that differentiate normal network activities from zero-day attacks. By employing machine learning models, particularly the one-class SVM, their solution has yielded the best results, with a Matthews correlation coefficient of 74% and an F1 score of 85%, for detecting zero-day network attacks.

In [130], Arun et al. harness deep learning and LSTM networks for zero-day attack detection. Their innovative approach involves simulating and modeling zero-day attacks by replicating unknown vulnerabilities through a sophisticated development framework. Leveraging recurrent neural networks and gating-based filtering techniques, their method scrutinizes abnormal behaviors to detect subtle deviations from normal patterns. These capabilities enable the identification of zero-day attacks. The results achieved with their DLAD (Dynamic LSTM-based Anomaly Detection) approach are promising, with a detection precision reaching 98.88% and a recall of 98.78% on the NSL-KDD and CI-CIDS2017 datasets.

Facing the diversification of approaches to detecting known and unknown attacks, Mahdi et al. present an innovative framework in [131]. This framework, based on deep learning (DL), is designed to be adaptable to zero-day attacks, thus addressing the challenges of detecting novelties and adapting to detected attacks. It comprises several distinct phases: open set recognition, clustering of unknown samples, supervised labeling, and model updating. The open set recognition phase utilizes deep classification methods to identify unknown samples while simultaneously classifying different types of known attacks. Unknown samples are then clustered, simplifying the labeling task for security experts. The optimized clustering phase aims to enhance the coherence of clusters, thereby facilitating subsequent supervised labeling. Finally, the model is updated with newly labeled data, making it adaptable to emerging attacks and evolving threat landscapes. This framework offers a promising approach for zero-day attack detection in networks.

We present in Table 3.1 a comparative analysis of various solutions from both traditional and recent state-of-the-art approaches.

Attack Detection Approaches	Signature-based	Statistical Anomaly	Supervised Learning	Unsupervis-ed Learning	Hybrid
Ability to Detect Zero-day Attacks	Low	Medium	Medium	High	High
Sensitivity to False Positives	Medium	High	High	Variable	Variable
Complexity and Resource Requirement	Low	Medium	Medium	Variable	Variable
Adaptability to New Attack Types	Low	Medium	Medium	High	High
Ease of Deployment in Industrial Environment	High	Variable	Variable	Variable	Variable

Table 3.1: Comparison of Zero-day Attack Detection Approaches.

Summary and issues:

After conducting a comprehensive analysis of the state of the art, several crucial issues emerge regarding data processing, enhancing attack detection through machine learning and deep learning techniques, and the development of zero-day detection framework.

1. Data processing: Efficiently managing the massive volumes of data generated by security systems remains a major challenge. Traditional data processing architectures have limitations in scalability and real-time data processing capacity. Moreover, the heterogeneity of data sources complicates their integration and analysis, potentially leading to delays in threat and attack detection.

2. Improving attack detection using ML and DL techniques: Despite advancements, challenges persist in enhancing attack detection capabilities. Current models face high rates of false positives or false negatives, compromising their operational effectiveness. Additionally, ensuring the adaptability of these models to emerging and evolving attacks remains a significant issue, necessitating continuous updates to algorithms and training datasets.

3. Zero-Day detection frameworks: Traditional detection methods often rely on recognizing pre-established signatures or patterns, rendering them ineffective against unknown attacks. While progress has been made in developing behavioral detection techniques, zero-day detection frameworks still contend with high rates of false positives, significant computational resource requirements, and the need for adaptability to an increasingly diverse array of attack classes.

Following an in-depth exploration of traditional and contemporary attack detection methods, particularly those based on machine learning and deep learning, we will focus in the subsequent section on studying existing datasets in the literature and the suggested data generation techniques.

3.4 Literature review for network information system data generation

The evaluation of intrusion detection systems (IDS), particularly those based on ML and DL methodologies, is intrinsically linked to the availability of pertinent and inclusive datasets. These datasets serve as the cornerstone for training, validating, and assessing the efficacy of IDS models across diverse security scenarios. This section elucidates the pivotal role of datasets in the evaluation of IDS solutions, highlighting their significance, utilization, and the subsequent necessity for synthetic data generation.

3.4.1 Available datasets

Intrusion detection datasets typically contain network traffic data, with features extracted from packet headers and payloads, along with labels indicating normal or malicious behavior. These datasets can be broadly categorized into two types: real-world datasets and synthetic datasets. Real-world datasets are captured from actual network environments, while synthetic datasets are generated through simulations or artificial means.

In the realm of ML and DL, two primary categories of datasets are commonly encountered: public datasets and private datasets. Public datasets are widely available and easily accessible, thus offering a significant convenience for research endeavors. Some widely-used datasets for evaluating intrusion detection systems include KDD Cup 1999, NSL-KDD, UNSW-NB15, CICIDS2017, CSE-CIC-IDS2018, AWID, ISCX, CTU-13, DARPA 1998, and Kyoto 2006. Private datasets, while often more representative of real-world situations, are characterized by their confidentiality and restricted access. Their more authentic and diversified content renders them valuable resources for research, Some privately-held datasets used for evaluating intrusion detection systems, which are typically not accessible to the public due to sensitive or proprietary information, include proprietary datasets of cybersecurity companies, enterprise network datasets, honeypot datasets, and darknet datasets. Thus, while public datasets are more readily accessible, private datasets offer superior realism but demand additional efforts for acquisition and utilization.

The table 3.2 compares the main characteristics of the 10 public datasets presented such as the year of creation, the number of records, the number of features, the types of attacks represented, as well as the advantages and disadvantages of each dataset.

Dataset	Year	Records	Attrib.	Target	Advantages	Disadvantages
DARPA [132]	1998	4,900,000	41	60	Large, variety of attacks	Obsolete, imbalanced, redundancy
KDD Cup [133]	1999	4,900,000	41	24	Large, variety of attacks	Imbalanced, redundancy, obsolete
Kyoto [134]	2006	24,000,000	24	50	Very large, variety of attacks, real	Imbalanced, fewer features
NSL-KDD [135]	2009	125,000	41	24	Addresses some KDD issues, more realistic	Smaller, some obsolete attacks
ISCX [136]	2012	2,000,000	21	5	Realistic, variety of attacks	Imbalanced, fewer features
CTU-13 [137]	2013	1,600,000	41	7 (botnet)	Specialized in botnets	Imbalanced, lack of diversity
UNSW-NB15 [138]	2015	2,500,000	49	9	Newer attacks, additional features	Imbalanced, complexity of features
AWID [139]	2017	17,000,000	156	4	Specialized in wireless networks	Limited attacks, lack of diversity
CICIDS [140]	2017	2,800,000	80	7	Recent, variety of attacks, realistic	Imbalanced, complexity of features
CSE-CIC-IDS [140]	2018	16,000,000	80	7	Large, variety of attacks, realistic	Imbalanced, complexity of features

Table 3.2: Comparison of intrusion detection datasets with their main characteristics.

Limitations:

- **Lack of diversity and realism:** The available datasets may suffer from various limitations that restrict their representativeness and relevance for evaluating intrusion detection systems. Among these limitations are:
  - Lack of diversity in attack types: Existing datasets may lack diversity in representing various types of attacks, thus neglecting the breadth of possible intrusion scenarios in a real network environment. This lack of diversity can compromise intrusion detection systems' ability to effectively detect novel or evolving threats.
  - Lack of realism in attack scenarios: Some datasets may not accurately reflect real-world attack scenarios, limiting their ability to simulate authentic cyber threat conditions. A lack of realism in the data can skew the performance evaluations of intrusion detection systems and lead to unreliable results.
  - Bias in data distribution: Datasets may exhibit biases in the distribution of data among attack classes and normal activities, skewing the evaluation of intrusion

detection system performance. Biases can result in underrepresentation of rare or novel attacks, compromising the systems' ability to effectively detect such threats.

- **Difficulty in acquiring real-world data:** Acquiring real-world data on attacks and intrusions can be hindered by several challenges, including:

  - Difficulty and cost: Collecting real-world data on attacks and intrusions can be a challenging and costly task, requiring significant resources in terms of time, personnel, and monitoring infrastructure.
  - Privacy concerns: Real-world data on attacks and intrusions may contain sensitive information about users and network activities, raising privacy concerns and compliance issues with data protection regulations.

- **Risk of overfitting:** The use of limited datasets can lead to overfitting of intrusion detection models, where the models overly adapt to the specific characteristics of the training data and lose their ability to generalize effectively to new data. This overfitting can result in poor performance of intrusion detection systems when deployed in real-world environments, where conditions may differ from those in the training dataset.
  To address the limitations of existing datasets, synthetic data generation has emerged as a promising solution.

3.4.2 Synthesis data generation techniques

Synthetic data generation involves creating artificial datasets that mimic real-world network traffic and attack patterns. These data are essential for enriching the limited available datasets and for enhancing the capability of intrusion detection systems to effectively identify and respond to threats. To address the major challenges associated with the use of real-world data, synthetic data plays a crucial role in the field of intrusion detection.

First and foremost, privacy emerges as a significant concern when dealing with authentic data, which may contain sensitive information about individuals or organizations. Synthetic data, being artificially generated, offers a secure alternative that preserves privacy, as it is dissociated from real entities.
Furthermore, authentic attack data is often scarce and challenging to obtain, especially for the latest and most sophisticated attack types. The generation of synthetic data enables the creation of a variety of representative attack examples, thereby enhancing the capability of intrusion detection models to identify a wide range of threats.
Another challenge in intrusion detection lies in class imbalance, where normal events are typically far more common than attack events. Synthetic data helps address this issue by producing additional examples of attacks, thus balancing the classes and improving the performance of detection models.
Lastly, security threats evolve continuously, requiring up-to-date data to train and evaluate intrusion detection systems. Synthetic data can be tailored to reflect new trends and emerging attack techniques, offering greater flexibility and adaptability to changes in the threat landscape.

Different techniques are employed in the state of the art for generating synthetic data in intrusion detection:

Deep generative models:

- **Generative Adversarial Networks (GAN)** have emerged as a significant advancement in the field of machine learning, consisting of two deep neural networks in competition with each other: a generator and a discriminator. The generator produces synthetic data examples, while the discriminator attempts to distinguish between real and synthetic data. Through an iterative training process, the generator progressively improves its ability to generate realistic data, while the discriminator refines its discrimination capabilities. Since its introduction in 2014, GANs have been successfully implemented in several fields, particularly in data generation [141].

  Several studies have proposed different GAN-based models for generating synthetic network traffic data. Ring et al. proposed two models, B-WGAN-GP [142] and E-WGAN-GP [143], to generate realistic synthetic NetFlow traffic using different input data representations from the original CIDDS-001 dataset [144]. Both models use Wasserstein GAN with gradient penalty and the two time-scale update rule (TTUR) to improve training stability. While the proposed models generate realistic synthetic NetFlow traffic, the evaluation method using domain knowledge may be limited and not take into account all possible behaviors in network traffic flows, which could affect the validity of the results. Therefore, it would be interesting to consider other evaluation methods to confirm the quality of the generated synthetic traffic.

  In another study [145], the authors used the B-WGAN-GP model, a variant of WGAN-GP trained to generate network traffic. The experiments used NetFlow records from the CIDDS dataset. The results showed that classifiers performed better when trained on synthetic data generated from longer GAN training. However, none of the classifiers outperformed the baseline trained on the original dataset. The results also indicated that the GAN failed to capture patterns between traffic features and labels, but could effectively capture and replicate patterns between the features of a NetFlow record and its class. However, when the GAN was not sufficiently trained, the classifier's performance was hindered by the synthetic traffic.

  The PAC-GAN model proposed by the Cheng et al. generates packet-level traffic using a convolutional neural network (CNN) GAN [146]. The packet data is encoded as 28×28 binary matrices, taking advantage of the raster data structure used in CNN models. While PAC-GAN was evaluated by measuring the success rate and byte error of decoded packets generated by the model, this evaluation method may not fully capture the complexity and variability of network traffic flows. Therefore, it would be beneficial to consider additional evaluation methods, such as comparing the statistical properties of the generated traffic with real-world traffic, or using the generated traffic in a network simulation to evaluate its impact on network performance.

  Dowoo et al. introduce PcapGAN, a GAN-based model for generating, augmenting, and analyzing network traffic data [147]. The model utilizes real traffic flow packet captures (Pcaps), extracting features and transforming them into graph (source and destination IPs), image (time interval), and layer sequences. These sequences are then used to create packet data for each protocol with augmented option data. The time interval information aids in setting the start time for the first packet, reception time for other packets, and packet sorting at each host, ultimately generating a Pcap file [Citation17]. The evaluation shows a similarity score of 0.5 between original and synthetic data, indicating moderate similarity. However, PcapGAN's limitation

is its requirement for data to be converted into graph, image, and layer sequences before training.

In another study [148], Lu et al. propose a method for generating SQL injection attack samples using deep convolutional generative adversarial networks (DCGAN) combined with genetic algorithms. This approach aims to address the problem of insufficient SQL injection samples for training and testing AI detection models by generating realistic and diverse fake samples. The proposed method expands the initial small number of samples and avoids overfitting of the AI models. Experimental results show that this approach is effective in generating realistic and diverse SQL injection attack samples, thereby improving the performance of AI detection models. The complexity of the DCGAN and genetic algorithm approach may limit its scalability and practicality in real-world applications.

Two GAN architectures, CTGAN and Copula GAN, were presented by Anande et al. for generating synthetic network traffic data [149]. These architectures utilized reversible data transformations to model dependencies between network traffic features. The authors relied on the UNSW-NB15 dataset, and network traffic features considered in this study included continuous features such as flow duration and packet count, as well as categorical features such as the protocol used and service type. Experimental results demonstrated that both GAN architectures were capable of generating synthetic data with statistically similar characteristics to real data in approximately 85% of cases.

In conclusion, while GANs offer promising advantages for synthetic data generation, there are challenges and limitations to their use. These include the need for more comprehensive evaluation methods, consideration of all possible behaviors in network traffic flows, the complexity of the approach and modeling of complex dependencies between traffic features, and the use of real, non-simulated data as input. Nonetheless, GANs have shown great potential in generating synthetic network traffic data, and further research is needed to address these challenges and limitations.

- **Variational Autoencoders (VAE)** are generative models used in deep learning, designed to learn representations of complex data and generate new data from these representations. VAEs combine encoder and decoder techniques with probabilistic concepts to create models that can generate realistic data. The state-of-the-art solutions demonstrate the potential of VAE for generating synthetic data in network intrusion detection [150].

  In [151], Manuel et al. presents a method for generating synthetic data in the field of intrusion detection using a Variational Generative Model (VGM) based on a Conditional Variational Autoencoder (VAE). The VGM uses intrusion class labels as input, enabling the generation of new data using only the labels without relying on specific training samples associated with those labels. The synthetic data generated by the VGM can be used as additional training data to improve classification results for common machine learning classifiers. The VGM demonstrates good performance metrics (average accuracy and F1) when several common classifiers (Random Forest, Logistic Regression, SVM, and Multilayer Perceptron) are trained with the VGM-generated data. The authors use the NSL-KDD dataset to train the VGM model and explore various architectures, including different numbers of layers, nodes, regular-

ization, loss functions, and probability distributions for the output layer. However, the experiments presented in this paper are based on the NSL-KDD dataset. It would be important to verify whether the results obtained with the VGM generalize to other intrusion detection datasets.

In a study, the authors present NeCSTGen, a network traffic generation architecture based on deep learning models such as variational autoencoders (VAEs) and recurrent neural networks (RNNs) [152]. The VAEs are used to learn the underlying distribution of the original network traffic data and generate new data that follows this distribution, while the RNNs are used to model the temporal dependencies in the network traffic data and generate coherent packet sequences. Experimental results show that NeCSTGen is able to generate network traffic with statistical characteristics similar to those of the original traffic data, on various types of networks including a simulated DARPA network, a Google Home voice command capture, and a LoraWAN network deployed in a city center.

In [153], the authors propose the Batched-VAE approach for generating balanced data in network intrusion detection datasets. This approach uses VAEs trained in batches to address the insufficient decoder training problem in the VAE approach. The results show that Batched-VAE provides similar classification accuracy in terms of F1 scores, even when the imbalance ratio changes. The authors use low-dimensional features to train a VAE for each original data sample, which enables obtaining accurate features for generating more favorable balanced samples. However, using low-dimensional features to train a VAE for each original data sample may not capture enough information to generate realistic balanced samples, and the Batched-VAE approach may not be scalable for very large datasets due to the need to train multiple VAEs in batches for each data portion.

In conclusion, VAE are a valuable tool for generating synthetic data in various fields, including network intrusion detection. VAE have shown promising results in generating realistic data by learning complex data representations and combining encoder and decoder techniques with probabilistic concepts. However, there are challenges and limitations to their use, such as the need for more comprehensive evaluation methods, the consideration of all possible behaviors in network traffic flows, the complexity of the approach and modeling of complex dependencies between traffic features, and the use of real, non-simulated data as input. Despite these challenges, VAE have demonstrated great potential in generating synthetic network traffic data, and further research is required to address these limitations and improve their performance.

Rule-based models and profiling models are strategies employed to generate synthetic data in intrusion detection by leveraging knowledge of the information system and its characteristics. These techniques enable the creation of realistic attack scenarios and the generation of data that accurately represents the behavior of attackers, as well as legitimate activities.

Both models facilitate the creation of realistic attack scenarios and address issues related to privacy, scarcity of attack data, and the evolving nature of security threats.

3.4.3 Summary

We have highlighted that existing datasets often lack diversity and realism in attack scenarios, may be biased, difficult to obtain and raise privacy concerns. Meanwhile, synthetic data generation approaches such as GAN and VAE offer promising advantages, but

also face challenges like the need for more comprehensive evaluation methods, consideration of all behaviors in network, the complexity of the approach and modeling of complex dependencies between traffic features, and the use of real, non-simulated data as input.

3.5 Conclusion

In this chapter, we first addressed the major challenges in terms of computer network security faced by enterprises. Subsequently, we examined state-of-the-art solutions, starting with traditional intrusion detection approaches and then focusing on recent techniques based on Machine Learning (ML) and Deep Learning (DL) algorithms. Given that these techniques require data for validation, we also explored existing datasets used in attack detection and synthetic data generation techniques.
Overall, our literature review enabled us to identify the following key challenges:

1. The lack of generic solutions for classification is a significant challenge. Indeed, the diversity of applications and obfuscation techniques hampers the effectiveness of signature-based and payload-based detection methods, leading to a high number of false positives in anomaly detection techniques. Otherwise, existing machine learning (ML) and deep learning (DL) approaches remain tailored to specific datasets, resulting in varying levels of accuracy and false positive rates. Furthermore, the extraction and definition of features utilized by these algorithms heavily depend on the targeted application case. Consequently, solutions designed for one dataset often struggle to generalize to others, especially given the continuously evolving diversity and complexity of cyberattacks.

2. Existing methodologies for zero-day attack detection are often inadequate. Traditional signature-based systems struggle to identify new threats lacking predefined signatures, while new signature detection solutions have limitations in covering various attack categories. Additionally, anomaly detection techniques face challenges in distinguishing between legitimate and malicious anomalies, often resulting in a high rate of false positives. Moreover, ML and DL approaches primarily focus on optimizing hyperparameters to ensure consistent performance across diverse datasets, often neglecting the potential for multiple attack scenarios.

3. Another major issue identified in our state-of-the-art study is the limitations associated with evaluation datasets proposed by the community. After analyzing these datasets, we also investigated synthetic data generation techniques. The increasing use of ML approaches in computer security emphasizes the need for high-quality, realistic, and scalable synthetic data. However, existing datasets often lack diversity and realism in attack scenarios, may be biased, difficult to obtain, and raise privacy concerns. Synthetic data generation approaches like GAN and VAE offer promising advantages but face challenges such as the need for comprehensive evaluation methods, consideration of all behaviors in the network, complexity in modeling dependencies between traffic features, and the utilization of real, non-simulated data as input. Additionally, limited access to real-world datasets for validating proposed methods necessitates the utilization of publicly available datasets, which may have drawbacks in terms of size, coverage of attack types, and potential biases.

In the following chapters, we will propose approaches for the three major points mentioned above.

Chapter 4

Automation and improvement of cyber-attacks detection via an industrial IDS probe

Contents

4.1 Introduction

Given the diversity of approaches proposed for intrusion detection via IDS probes, an analysis and evaluation of the feature engineering and classification techniques proposed in the existing intrusion detection approaches is required.
In their study [62], Maharana et al. highlighted the critical importance of data preprocessing and data augmentation techniques for ML and DL models. One of the most crucial steps in data preprocessing is feature engineering, which involves extracting and selecting the most relevant features for the models.

The literature presents different solution such as using default attributes defined based on datasets [7, 9, 10], statistical features [86], reducing the dimensionality with PCA [97, 116], ML and DL algorithms [8, 101] and applying feature selection techniques (wrapper or filter method) [100, 104]. Combined with a classification algorithm for attack detection, these solutions can be time-consuming to execute and complex. Furthermore, their effectiveness can be limited from one dataset to another. Therefore, it is advisable to classify network flows for the detection of cyberattacks, aiming to reduce false detection rates and model execution time by leveraging general attributes.

In this chapter, we propose an approach to better classify network flows. Our solution involves extracting generic attributes combined with a classification algorithm capable of

identifying additional patterns in the input data, thereby associating network flows with the appropriate classes. For this purpose, we use the 1D-CNN algorithm and its feature detector layer. We also aim to reduce the number of attributes to decrease the execution time of the models in an industrial context.
Our solution is evaluated using different datasets: the first originating from an industrial context, and others from the literature (NSL-KDD, UNSW-NB15), to demonstrate the effectiveness of the model regardless of context and datasets. We highlight the attack classes, the volume of data used, the processing applied to these data, and the classification results, while providing a comparative study with some state-of-the-art works.

In this chapter, we first introduce a background of network flow. After that, we present the architecture of our solution, detailing its two major phases. Subsequently, various evaluations are proposed using data from different contexts.

4.2 Background

As illustrated in Figure 4.1 and discussed in Section 2.2.1, a network flow represents a sequence of packets between a pair of communicating endpoints, such as *Asset A* and *Asset B*, during a specific time interval [17]. Thus, the establishment of a network flow, also known as a session, involves a series of steps that enable *Asset A* and *Asset B* exchange data. First, *Asset A* initiates a connection request to *Asset B*, using a specific communication protocol, such as TCP or UDP. *Asset B* then responds with an acknowledgement, establishing a logical connection between the two endpoints. Subsequently, these two assets engage in a series of data exchanges, with *Asset A* sending requests and *Asset B* sending responses, until the session is terminated, either explicitly by one of the endpoints or implicitly due to a timeout or other network event.

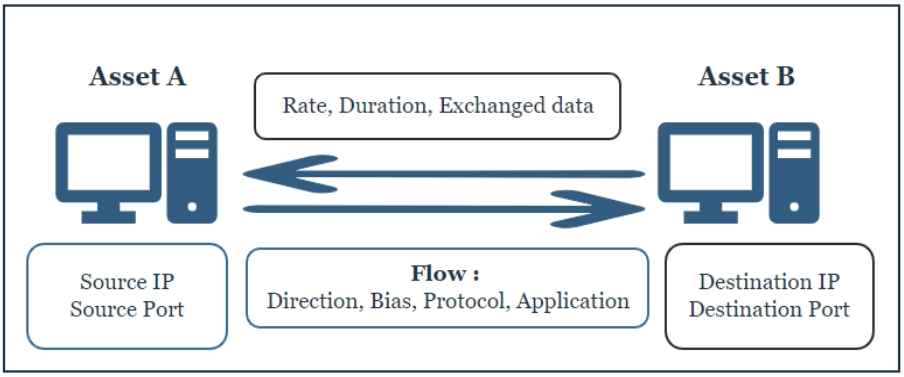

Figure 4.1: Flow definition.

The characteristics of a network flow, including source and destination IP addresses, source and destination ports, and communication protocols, provide critical insights into the nature and behavior of network activity. However, raw network data can be highly complex, noisy, and high-dimensional, with redundant and encrypted information that can significantly impair the performance and accuracy of network activity classification algorithms.

To address these challenge, extracting and analyzing metadata is an appropriate approach. In Figures 4.2, 4.3 and 4.4, we present a metadata (header of a flow), a header with an empty payload, and a payload of encrypted traffic. These metadata features can

effectively capture the essential characteristics of network flows while reducing the complexity and noise of the raw data.

Flow Information

Protocol	tcp_ip	Application	Web.Web.Misc				
Magnitude	(2)	Relevance	1	Severity	1	Credibility	5
First Packet Time	Sep 20, 2019, 11:58:23 AM	Last Packet Time	Sep 20, 2019, 11:58:23 AM	Storage Time	Sep 20, 2019, 11:59:23 AM		
Event Name							
Low Level Category	Web						
Application Determination Algorithm	Flow exporter (7)						
Request URL	1/page/blank.html						
File Hash	10fcc1855126a6ce02c2d6ebdee6b7acc481b50a2de2247d383896263a4b3ef2						
HTTP Referrer	http:// /#						
HTTP Version	HTTP/1.1						
File Entropy	2.85936						
File Size	1180						
HTTP Method	GET						
HTTP Host							
HTTP Server	lighttpd/1.4.13						
Content Type	text/html						
HTTP User Agent	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E; Zoom 3.6.0; wbx 1.0.0						
HTTP Response Code	200						
Flow Direction Algorithm	Single common destination port (1)						
File Name	blank.html						
Domain	Default Domain						

Figure 4.2: Network Flow - Metadata.

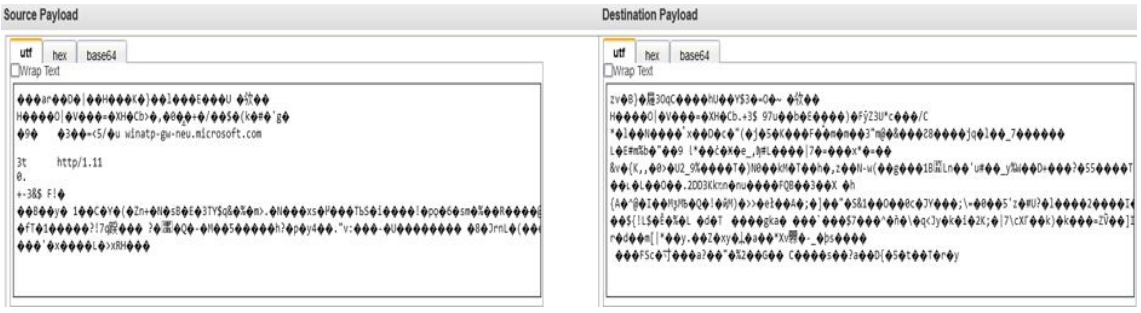

Figure 4.3: Network Flow - Packet Encrypted.

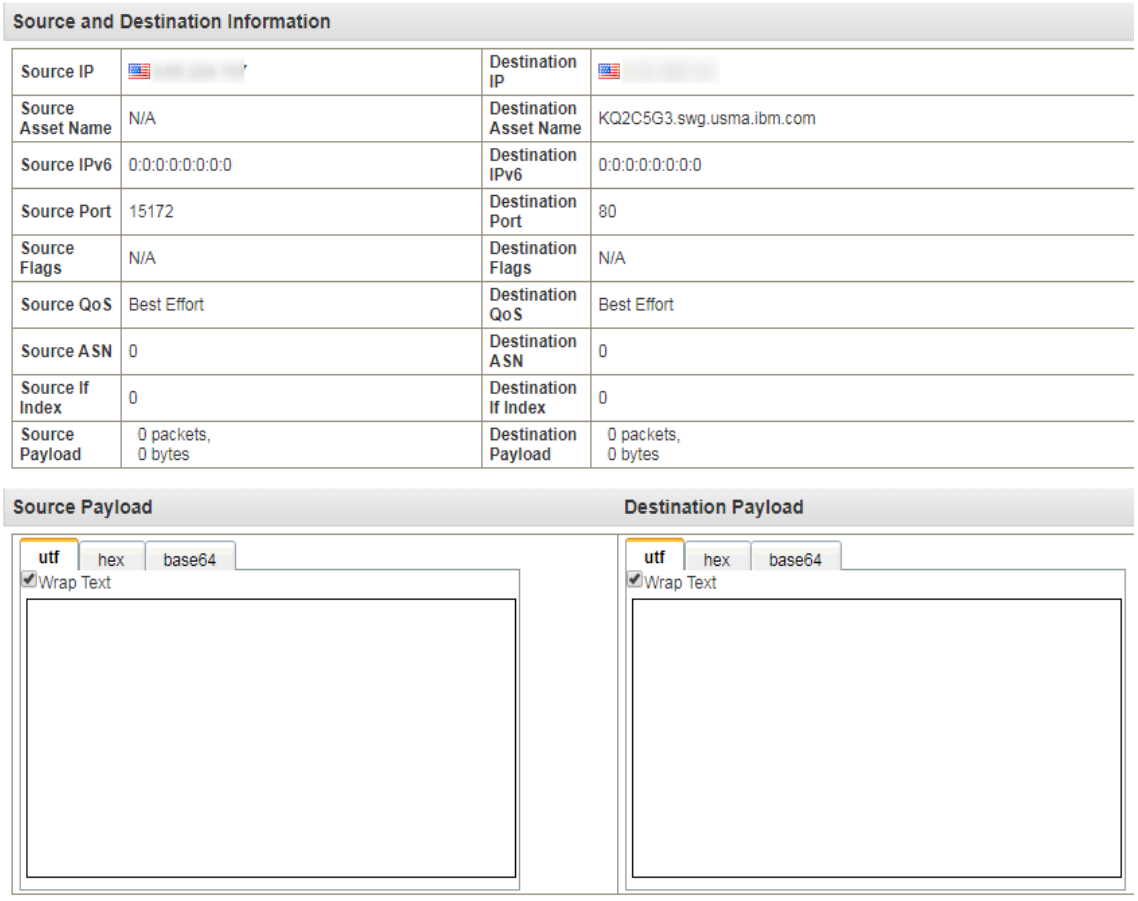

Figure 4.4: Network Flow - Empty Payload.

4.3 Our proposal

In this section, we present our supervised learning approach for network intrusion detection. Network activities (e.g., attempts to connect to a machine, port scanning, data transfers) in our approach are classified based on the features of network flows. Our approach uses CNN due to their feature detector component to achieve a fast and efficient classification and can be applied in any network context. Thus, as we can see in the Figure 4.5 the proposed approach has two main steps:

- the first part will deal with feature engineering which will be developed in Section 4.3.1 and then will focus on the operation of convolution neural networks, in particular the feature detector and feature map proposed in this algorithm;
- the last step will highlight the classification phase of the flows for the identification of the adequate classes to which the flows are associated, as described in Section 4.3.2.

4.3.1 Our feature engineering method

Based on the various phases of communication between the two machines presented in Section 4.2, our feature engineering aims at finding a set of universal minimalist features and facilitating several network flow analysis tasks. Thus, these features can be found and

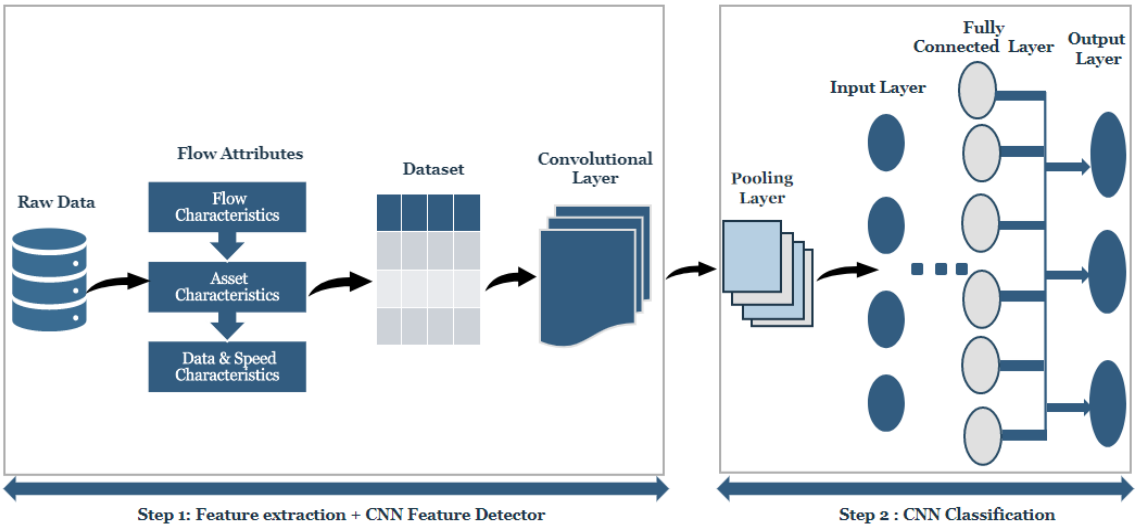

Figure 4.5: Our proposed model.

used to classify network activity. As shown in Table 4.1, the features of any flow can be ranged in three main categories, namely:

- **Asset characteristics:** This category encompasses attributes associated with the lower layers of the OSI model (Network, Transport, and Application), including source and destination IP addresses, port numbers, protocols used, and targeted applications. These attributes aim to characterize and identify the transmitting and receiving equipment involved in communication. Their universality lies in their properties of:
  - Fundamental for packet routing: These attributes are essential for routing data between devices on the network. They enable the identification of transmitting and receiving devices, determine communication paths, and ensure packets are delivered to the correct destination. Their presence is thus guaranteed in every network flow, regardless of the context or type of communication.
  - Independence from protocols and applications: Unlike some attributes that may vary depending on the protocol or application used, attributes in *Asset characteristics* family remain constant and immutable across all network communications. For example, a source IP address will always be a source IP address, whether it is used for an HTTP, FTP, or DNS connection. Similarly, a specific port number will be associated with a particular application, regardless of the protocol used.
- **Flow characteristics:** This category encompasses attributes associated with the session layer of the OSI model (layer 5), including communication duration, number of packets exchanged, average throughput, etc. These attributes serve to characterize the dynamics of communication between equipment, taking into account temporal variations in data exchange. Their universality lies in their ability to:
  - Independence from specific data content: They remain constant regardless of the type of content being exchanged.
  - Applicability to any network communication: Whether it involves file transfers, video streaming, web requests, or any other form of data exchange, they are available and identifiable.

- **Exchanged data and rate characteristics:**

  This category encompasses attributes related to data exchanged between equipment, such as packet size, amount of exchanged data, etc. These attributes aim to characterize the content of communication, considering the specificities of the exchanged data (e.g., file type, language used, etc.). They are essential for evaluating the volume of transmitted data and detecting anomalies in communication patterns.

Flow Characteristics	
Flow-ID	unique identifier of a network flow associated with a session includes source and destination IP addresses, along with source and destination port numbers, and a protocol.
Flow Type	Single standard flow or supeflow (network scans A - DDOS B - Port Scans C).
Flow-Aggregation-Count	Number of session records associated with a specific flow.
Flow Direction	Direction of traffic (Local Only - Local To Remote - Remote To Local).
Flow-Bias	Amount of incoming or outgoing data associated with a specific flow. (In only [100% incoming flows], Out Only [100% outgoing flows], Mostly In [$70-99\%$ incoming flows and $1-30\%$ outgoing flows], Mostly out [$70-99\%$ outgoing flows and $1-30\%$ incoming flows], Nearly Same [$31-69\%$ incoming flows and $31-69\%$ outgoing flows]).
Flow Duration	Difference between the date and time of receiving the last packet and the first packet.
Asset Characteristics	
Source-IP	IP address (Layer 3 of the OSI model) originating the flow.
Source-Port	Source port of the flow.
Destination-IP	IP address (Layer 3 of the OSI model) destined for the flow.
Destination-Port	Destination port of the model assigned by IANA.
Protocol	Protocol assigned to the network flow associated with the transport layer of the OSI model.
Category	Data low level Category
Application	The application name (Layer 7 of the OSI model) assigned based on the protocol and ports that are used for the flow, and the flow content (LDAP, Radius, MSN, NFS, IMAP, POP, BitTorrent, RDP, SSH, ...).
Application-Group	The group name for the application (Authentication, Chat, Web, Data Transfer, Mail, ...)
Exchanged data and rate characteristics	
Source-Bytes	Number of bytes sent from the source host.
Destination-Bytes	Number of bytes sent from the destination host.
Bit-Per-Second	Number of bits sent per second.
Total Bytes	Total number of bytes associated with the flow.

Table 4.1: Features categories and related attributes.

We extract features from these three categories from the network traffic generated by the original information system. The extraction of these universal metadata useful can reduce the processing time of a model to feed a network activity classification algorithm Subsequently, we construct the dataset with these metadata as headers and the values of various attributes extracted from the original traffic.

4.3.2 Our classification model

Once the dataset formed, the corresponding data are injected into a classification algorithm for the accurate identification of attack scenarios present in the traffic. For

this purpose, there are different ML and DL approaches and methodologies, as we have seen in the state of the art, most of which aim to improve the accuracy of the model.

First, some attack categories have similar feature values, which may reduce the effectiveness of ML algorithm-based models. Moreover, we aim to apply our approach in an industrial context with substantial data volumes. Therefore, we propose a classification based on deep learning where some features are learned during the training phase. Hence, in our selection of deep learning approaches, we prioritized convolutional neural networks (CNN) due to their unique ability to detect relevant features in their convolution layer.

Our methodology initially relies on the extraction of a minimalist set of attributes in a network flow (see Section 4.3.1). In a second step, the feature detection component of the CNN is a major asset for accurate traffic classification.

As illustrated in Figure 4.6, the feature detector within the CNN allows to identify the most significant features in the input data, which are reflected by the highest values in the feature map. This detection operation is performed by a convolution between the input data (dataset) and the feature detector, which generates a feature map representative of the most relevant data for our study.

The use of CNN in our approach allows us to effectively combine minimal feature extraction and precise pattern detection in the network flow, which significantly improves the performance of our traffic classification model. Moreover, our approach takes into account the training and execution time of the models. By using a minimalist feature extraction method, we reduce the dimensionality of the input data, which in turn speeds up the training and execution time of the CNN.

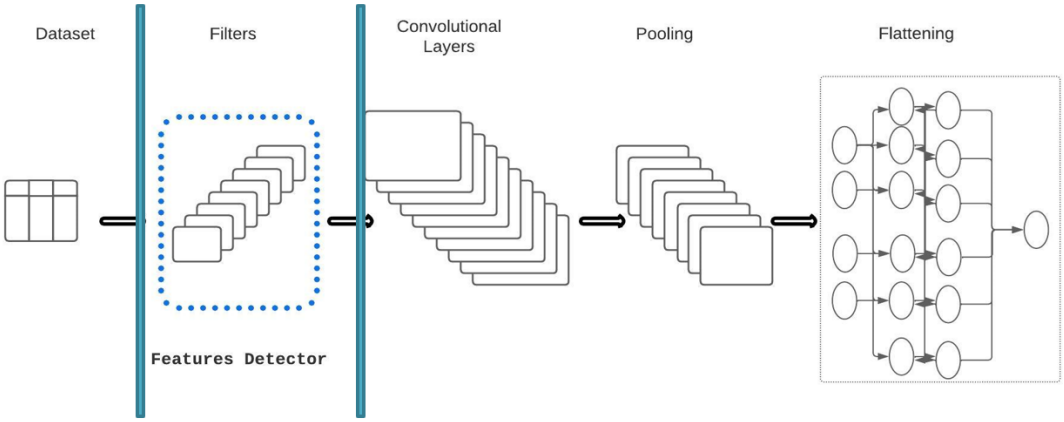

Figure 4.6: Features Detector Layer.

In addition, to improve model efficiency and reduce computational cost, we are focusing on the other layers of CNN algorithm.

In our study, we use the algorithm 1D-CNN an approach based on one-dimensional 1D-CNN with specific parameters to improve the performance of our traffic classification model, without focusing too much on hyperparameters optimization, which considerably limits the reuse of a model in another context or on other datasets.

We choose to use 64 filters in the first convolution layer of the network to extract 64 different features from the input data. This decision is based on a trade-off between

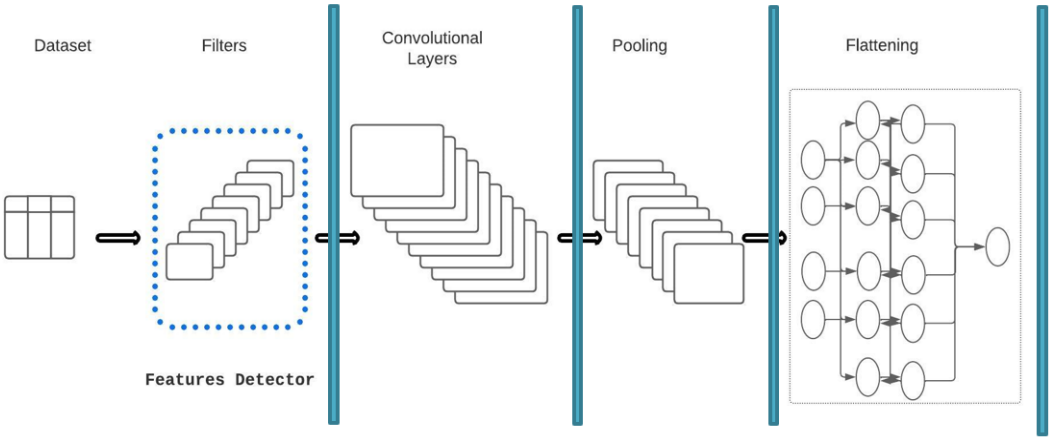

Figure 4.7: Other Layers of Convolutional Neural Networks.

model accuracy and computational cost. Previous studies have demonstrated that a moderate number of filters in the first layer provides a good balance between these two factors [154].

Next, we select a *kernel-size* of 3 (3x3 matrix) for the convolution layer. This kernel size is chosen for the convolution layer based on its ability to detect recurring patterns in the input data while limiting computational cost. This kernel size has been widely used in CNN and has been shown to be effective for large-scale pattern recognition [155].

The *ReLu* activation function is used in the convolution layer to add non-linearity to the model and avoid the problem of saturation. This activation function has been shown to accelerate the convergence of the model and improve its performance [156].

We also use the *max-pooling* layer to ensure spatial invariance property and to reduce the risk of overlearning by removing unimportant information and keeping only the most relevant ones to generalize the model. A *pool-size* of two was chosen as it is a commonly used value in CNN and has been shown to be effective in various applications. [157].

After the pooling layer, we add a *fully-connected* layer with *128 units*, which corresponds to the average number of neurons in the input and output layers while maintaining the *ReLu* activation function.

For the output layer, we define the number of output neurons corresponding to the number of targets present in each of the datasets. We rely on the **cross-entropy** cost function for this network flow classification problem, with an **Adam** optimizer and accuracy to measure the performance of our model. Finally, we use the *Softmax* activation function to evaluate the output probabilities of each class and **Sigmoid** for binary classifications.

To further improve the performance of our 1D-CNN model, we increase its depth while considering the trade-off between model complexity, accuracy and computation time. Specifically, we can add layers to the convolution layer, the fully connected layer, or both.

We define a loop that adds three convolution layers to the network, and apply max-pooling to these new layers while maintaining the same parameters as in the first

Parameter	Value
Convolution Layer	3 layers
Max-pooling	2 layers
Dropout	0.3
Fully Connected	2 layers
Output Layer	Softmax
Optimizer	Adam
Activation Function	ReLu/Softmax
Epoch	200
Batch Size	3

Table 4.2: Our CNN parameters.

convolution layer. This allow us to extract more complex and higher-level features from the input data.

However, to prevent overfitting and minimize the risk of over-training, we use dropout regularization to randomly disable some neurons in the network during training. This help to reduce the co-adaptation of neurons and improve the generalization of the model. In our 1D-CNN model, we set the dropout rate to **0.3**, which means that 30% of the neurons will be disabled at each training iteration.

Table 4.2 summarizes the different settings and parameters of our proposed 1D-CNN model, including the number of layers, the number of filters, the kernel-size, the pooling-size, the number of units in the fully connected layer, the dropout rate, and the activation functions used in each layer.

In summary, the choice of these parameters for our 1D-CNN approach was motivated by previous studies and considerations of accuracy, computational cost, and model generalization. In the next section, we evaluate the performance of our model with these parameters.

4.4 Performance evaluation

In this section, we evaluate the accuracy of our model in network flow classification. To do so, we first apply our feature engineering coupled with our CNN-based model on an IBM dataset extracted from a real industrial context. We compare then our solution to four ML algorithms (see Table 4.3) in terms of binary and multi-class classification. Since we aim to propose a general solution for network flow classification, we show, in the second part of this section, the effectiveness of our solution against existing solutions while considering, this time, the well known NSL-KDD dataset that is widely used in the literature.

4.4.1 Model performance on an industrial context: IBM dataset

IBM QRadar is a security appliance that is built on Linux. QRadar allows to collect, store and correlate logs for the detection of security incidents. To prove the effectiveness of our approach, we extracted raw data from a real-time industrial context, with an IBM proprietary IDS probe (QRadar Network Insight, QNI), which extends QRadar by providing a detailed view of real-time network communications [158].

Algorithms	Parameters
K-Nearest Neighbors	***n_neighbors***: number of targets; ***weights***: uniform; ***algorithm***: auto; ***leaf_size***: default; ***p***:2; ***metric***: euclidean
Naive Bayes	***model***: GaussianNB; ***classes***: number of targets
Decision Tree	***criterion***: gini; ***splitter***: best; ***min_samples_split***: 2; ***max_depth***: none; ***random_state***: none
Random Forest	***n_estimators***: number of targets; ***criterion***: gini; ***max_depth***: none; ***min_samples_split***: 2; ***max_features***: sqrt; ***random_state***: none;

Table 4.3: Machine learning algorithms parameters.

Our dataset includes data collected in 2021, and thus recent attack chains (e.g., exploit log4Shell vulnerability). The flows used in our classification have been extracted during an interval of one week. This extraction (around 36 000 enties) contains legitimate flows and non-legitimate ones that we categorize using the framework MITRE ATT&CK [30]. This content is presented in Table 5.3.

Flow-class	Target-ID	Tactics	Techniques	Size (#rows)
Normal	0	-	-	10.000
Large-Leakage	2	TA0010	T1567	10.000
Stealthy-leakage	1	TA0010	T1030	10.000
Web-Exploit	7	TA0002	T1204	5.000
DOS-Attack	3	TA0040	T1498	150
Indicator of Compromise Inbound	4	TA0001	T1189	500
Indicator of Compromise Outbound	5	TA0011	T1102	300
Malicious-Website	6	TA0011	T1102	15

Table 4.4: IBM Dataset content validated with MITRE ATT&CK.

Data preprocessing

At this phase, we make transformations on our raw data to allow its processing by the learning algorithms that we used. The raw data has an average of 150 features default and custom. However, using all these features for classification is not an optimal approach considering a balance between accuracy and model execution time. Therefore, in our solution, we reduce the number of features to 14 (at most) according to our feature engineering approach (Section 4.3.1) in addition to one target that describes the flow class. After this extraction, all decimal entries (e.g., rate, amount of data, source/destination port numbers) remain unchanged. On the other hand, each byte in a field related to an IP address is converted to its hexadecimal value (excluding the dots that separate the bytes). Then, these values are concatenated which gives us a unique value that will be converted to decimal. The fields related to "flow direction" has at most four possible values in the raw data. Therefore, these values are mapped into integers in the set $\{1,..,4\}$. The fields related to "flow bias" has at most five possible values in the raw data. Therefore, these values are mapped into integers in the set $\{1,..,5\}$. Finally, the type of the communication protocol used is replaced by its corresponding number assigned by the Internet Assigned Numbers Authority (IANA) (i.e., 6 and 17 for TCP and UDP respectively).

For this dataset, we use a distribution of 70% for the train-set and 30% for the test-set.

Evaluation metrics

To determine the quality of our classification model and to evaluate its performance against various existing learning algorithms, we used the following four popular evaluation metrics:

$$Accuracy(A) = \frac{(TP+TN)}{(FP+FN+TP+TN)}, Precision(P) = \frac{TP}{TP+FP}, Recall(R) = \frac{TP}{TP+FN},$$
$$ScoreF1(F1) = 2 * \frac{(Precision*Recall)}{(Precision+Recall)}$$

Where, for a given flow class C: True Positive (TP) represents the number of flows correctly classified in the given class C; True Negative (TN) represents the number of flows correctly classified outside the class C; False Positive (FP) represents the number of flows wrongly classified in the class C; and finally False Negative (FN) represents the number of flows wrongly classified outside the class C.

The evaluation of binary and multi-class classification using two-feature families

As a first step to validate our approach, we started with a binary classification which should allow us to distinguish legitimate flows from illegitimate ones. To do so, we first assign all legitimate flows (Normal flows in Table 4.4) to the target 0 while all illegitimate flows will be assigned to the target 1. After that, we select ten features in each entry in our dataset. These features correspond to two features families among the three highlighted in section 4.3.1, namely, Flow features (the number of records in a flow, the type of flow, the direction of the asset that initiated the communication, the duration of the communication, the data transfer bias) and Asset features (the source IP address, the destination IP address, the source and destination ports, and the protocol used). This data has then been injected into our model and the results have been compared in Table 4.5 with those of four ML algorithms (K-Nearest Neighbors (KNN), Naive Bayesian (NB), Random Forest (RF), Decision Tree (DT)).

Target	KNN			NB			DT			RF			Our model		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
0	1	1	1	1	0.99	0.99	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Accuracy	100%			99%			100%			100%			100%		

Table 4.5: A comparison table in terms of binary classification using 2 families of features.

As we can see in the results of Table 4.5, our solution can indeed distinguish legitimate network traffic from an abnormal one, where we notice 100% accuracy rate for our model. The other ML algorithms also achieve good results. However, in a real industrial context, it is also important to distinguish the nature of the attack flows in order to take proper counter-measurements. Therefore, we present in Table 4.6 a comparison of a multi-class classification accuracy of illegitimate flows between our model and the above ML algorithms.

As shown in Table 4.6, we notice that our solution offers a better accuracy and a very good precision, recall and F1 scores for all attack targets. As a reminder, the F1

T	KNN			NB			DT			RF			**Our model**		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.74	0.76	0.75	0.94	0.15	0.26	0.70	0.69	0.69	0.71	0.75	0.73	1	0.96	0.97
2	0.75	0.74	0.75	0.55	0.79	0.71	0.68	0.69	0.68	0.72	0.68	0.70	1	0.92	0.94
3	1	1	1	1	1	1	1	1	1	1	1	1	1	0.95	0.92
4	0.83	0.31	0.45	0.40	0.67	0.50	0.93	1	0.96	0.93	1	0.96	0.91	0.91	0.94
5	1	1	1	1	0.45	0.62	1	0.86	0.93	1	0.91	0.94	1	0.96	0.96
6	1	1	1	1	1	1	1	1	1	1	1	1	0.85	0.89	0.92
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Acc	88%			79%			85%			86%			92%		

Table 4.6: Multi-class Classification with two families of features.

score is an arithmetic average between precision and recall, so it allows to measure the ability of the learning algorithms to better associate flows with legitimate traffic classes or corresponding target attacks. The first general observation is a confusion between targets 1 and 2 by almost all ML algorithms and a better classification for targets 3, 6 and 7 compared to the remaining targets. We note that targets 1 and 2 correspond to data leak flows including data leaks in short time intervals, and data leaks in much longer ones. For these flow types, the same IP address can be spotted in both scenarios, which may cause the confusion between the data in these two classes in ML algorithms. With KNN, the F1 score is 0.75 for target 1 and 2, and quite low, especially for target 4. In the Naive Bayesian, we notice the lowest accuracy among all ML algorithms that we tested, which strongly impacts the F1 score which is 0.26 for target 1. A significant confusion was also found in the NB between targets 4 and 5 (corresponding to the inbound and outbound compromise indicator data) with F1 scores of 0.5 and 0.62 respectively. This is explained by the fact that the flow duration in these two attack classes is relatively close, due to the probabilistic operation of NB. The Decision Tree and Random Forest improve these F1 scores at target 4 and 5 since when an attribute has the same approximate value for two given classes in an internal node, these two algorithms dissociate the given targets in the following nodes by comparing other attributes.

Our solution, which combines our feature extraction approach with the feature detector proposed in CNN, solves the confusion observed in ML algorithms. Indeed, a neural layer overlay, including the layer filtering principle, is well adapted to our context of multi-class classification of confused output layers. Convolutional neural networks learn the values of the weights in the same way that they learn the filters of the convolution layer, which can allow it to distinguish between confused classes 1 and 2, and other classes (like target 4 and 5) with medium or low F1 scores which explains the improvement of accuracy highlighted in Table 4.6.

The evaluation of multi-class classification using three-feature families

In order to investigate the impact of the third family of features (highlighted in Section 4.3.1) on the accuracy of our model, we all the three families of characteristics. Therefore, the number of characteristics increases from 10 to 14 features that we use in both our model and ML algorithms to classify malicious network flows. The results of this evaluation are presented in Table 4.7.

As we can observe in Table 4.7, an improvement of the F1 score of all targets is

noticed compared to the Table 4.6 in both our model and ML algorithms, with an exception for NB. In this model, the F1 scores are more or less the same as the ones in Table 4.6 for all targets, except for target 4 (incoming traffic related to indicators of compromise) where a sharp decline is observed. This decreases the accuracy of this model from 79% to 73%.

Overall, we can clearly deduce that the additional information provided by the third family of features gives much more context and allows us to dissociate certain types of attacks. Our solution appears as the best approach with an average F1-score of 0.96 and an overall accuracy of 94.84% which is better than the ones presented in Table 4.6.

T	KNN			NB			DT			RF			**Our model**		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.80	0.78	0.79	0.93	0.17	0.28	0.86	0.85	0.85	0.86	0.91	0.88	1	0.95	0.96
2	0.78	0.81	0.80	0.48	0.74	0.58	0.84	0.85	0.85	0.90	0.94	0.87	0.98	0.98	0.95
3	1	1	1	1	1	1	1	1	1	1	1	1	1	0.91	0.97
4	1	0.60	0.75	0.02	0.82	0.03	1	0.95	0.97	96	0.95	0.96	0.95	0.90	0.89
5	1	1	1	1	0.44	0.62	1	0.73	0.84	0.95	0.82	0.88	1	1	1
6	1	0.67	0.80	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Acc	90%			73%			91%			92%			94.84%		

Table 4.7: Multi-class Classification with all families of features.

The evaluation of epochs according to accuracy and loss values

The convergence of a model is determined by the analysis of the error rate and accuracy curves. Thus, we will focus on determining the optimal number of epochs in order to find a trade-off between the execution time of our CNN model and its overall accuracy because one of the drawbacks of deep learning based solutions is the execution time. In Table 4.2, we have initially defined a baseline value of two-hundred epochs which we believe is more than enough for our model to converge. However, we went through the `early-Stopping` technique to explore the possibility of stopping the learning, as soon as the model starts to overlearn and thus to reduce the execution time. To do so, we present in Figure 4.8, the accuracy and loss in the training data (blue curve) and in the test data (orange curve) according to the number of epochs used in our model. As we can see in Figure 4.8, from the first to the fiftieth epoch, the accuracy increases with the number of epochs and the loss decreases accordingly, which indicates that the model is still continuing to learn and converges. Between the fiftieth and seventieth epochs, both metrics start to stagnate. This stagnation continues until the 200^{th} epoch, with some fluctuations observed in some epochs due to confusions between some neurons.

4.4.2 A comparative analysis with the benchmarking dataset NSL-KDD

Dataset description

There have been many intrusion detection approaches proposed in the literature. However, these approaches are evaluated with different datasets. Therefore, we tested our model on the NSL-KDD dataset (that has been used in several works of the literature) in order to check its efficiency and establish a fair comparison with existing solutions that adopted the same dataset.

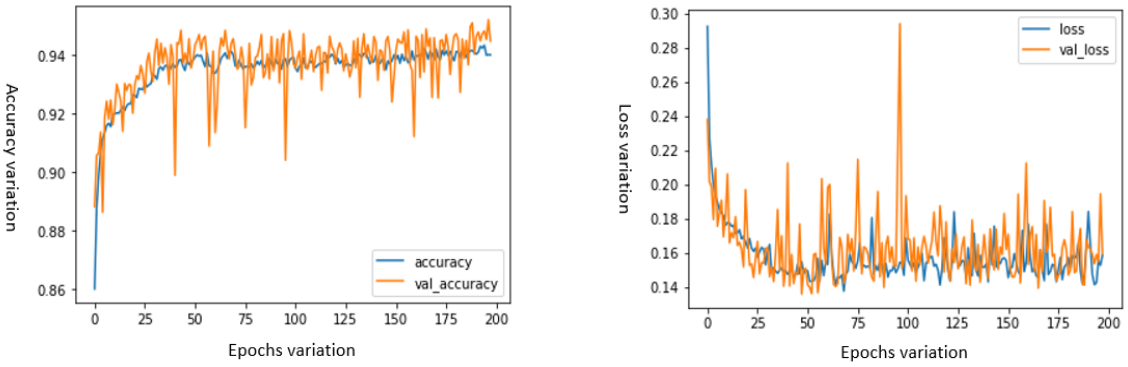

Figure 4.8: Number of epoch with accuracy and loss.

The NSL-KDD dataset [135] is an enhanced version of the KDDCup99 dataset [159] which is a standard dataset consisting of a wide variety of simulated intrusions in a military network environment, defined in 1998 by *MIT Lincoln Labs* of the U.S. DARPA agency. NSL KDD has a KDDTrain+ set (125,973 records) divided into 22 types of attacks and normal traffic and a KDDTest+ test set (22,544 records) which contains 14 additional attacks not included in the KDDTrain+. The set of attack classes is divided into four main families: Denial of Service (DOS), Remote To Local (R2L), User To Root (U2R), Probe. The content of NSL-KDD is presented in the Table 4.8.

Flow-class	Subclass	Tactics	Techniques	Train(#rows)	Test(#rows)
Normal	Normal(67343)	-	-	67 343	9 711
DOS	Back(956), Land(18), Neptune(41214), Pod(201), Smurfs(2646), Teardrop (892)	TA0040	T1498	45 927	7 458
R2L	Guess-Passwd(53), ftp-write(8), Imap(11), Phf(4), Multihop(7), Warezmaster(20), Waresclient(890), Spy(2)	TA0001	T1133/T1199	995	2 754
U2R	Buffer-overflow(30), Loadmodule(9), Perl(3), Rootkit(10)	TA0004	T1548	52	200
Probe	Satan(3633), Lpsweeo(3599), Nmap(1493), Postsweep(2931)	TA0043	T1589/T1590	11 656	2 421

Table 4.8: The NSL-KDD dataset content.

Data processing

NSL-KDD dataset provides 41 features grouped into 3 families: basic features, traffic features to the same host and traffic features to the same service. However, the application of our feature engineering (proposed in Section 4.3.1) will allow us to reduce the number of features used for training to the following 12 features (excluding labels): `duration`, `protocol-type`, `service`, `land`, `src-bytes`, `dst-bytes`, `count`, `srv-count`, `same-srv-rate`, `srv-diff-host-rate`, `flag` and `wrong-fragment`.

Feature	Description
Duration	Duration of the connection in seconds.
Protocol-Type	Type of network protocol used (e.g., TCP, UDP, ICMP).
Service	Type of service on the network (e.g., http, ftp, smtp).
Land	Indicator of whether the connection is from/to the same host/port.
Src- Bytes	Number of bytes transferred from source to destination.
Dst-Bytes	Number of bytes transferred from destination to source.
Count	Number of connections to the same host in the last two seconds.
Srv-count	Connections to the same service on the same host in the last two seconds.
Same-srv-rate	Rate of connections to the same service.
Srv-diff-host-rate	Rate of connections to different hosts for the same service.
Wrong-fragment	Number of incorrect fragments.
Flag	Status and control flag of the connection.

Table 4.9: Our Feature engineering on NSL-KDD dataset.

Moreover, to reduce the impact of inconsistencies between the training and test data in terms of targets, we chose to concatenate the two by keeping only 22 targets common to both sets, then splitting this concatenated set at 80% for the training sets and 20% for the test data. Then, we proceeded to a preprocessing and normalization of the extracted data, in which all numerical values are kept unchanged while attributes with categorical values like services and protocol-type have been mapped to decimal values, as it has been done with IBM dataset. Finally, targets have been labeled as follows: normal:0, DOS:1, Probe:2, R2L:3, U2R:4; and the resulting data is injected into our model. The results of the tests of our model on the NSL-KDD are presented in the Table 4.10.

Model application and results

Target	**Our model (NSL-KDD)**		
	Precision	Recall	F1-score
Normal	0.99	0.99	0.99
DOS	0.99	0.99	0.99
Probe	0.97	0.97	0.97
R2L	0.94	0.79	0.86
U2R	0.50	0.13	0.21
Accuracy	98,7%		

Table 4.10: The evaluation results of our Model on the NSL-KDD dataset.

Overall, as we can see in Table 4.10, our model achieves a global accuracy of 98%. We also notice a high precision and a good F1-Score for the targets `Normal`, `Probe` as well as `DOS` and to a lesser extent `R2L`. However, the target `U2R` has a low score compared to the remaining targets. Actually, the `U2R` target relates to an attempt to elevate privileges which does not correspond to the definition of network flow. Consequently, the classification of this target cannot be based solely on the characteristics of a network flow. Furthermore, the limited number of input samples for this target in the dataset contributes to its lower performance compared to other targets that are indeed network flows and have a significantly larger number of samples.

Moreover, based on the NSL-KDD dataset, we compare in Table 4.11 our model with other solutions proposed in the state of the art according to the confusion matrix

(precision, recall, F1-score and accuracy). Note that the chosen solutions use the following algorithms on NSL-KDD dataset: ANN [7, 104], Deep learning with ANN (Auto-encoder)([8], Naive Bayesian [97], and RNN+LSTM, CNN [10].

Model	A	P	R	F1	Feature Engineering
Our model	0.98	0.88	0.78	0.80	
Vinayakumar et al. [7]	0.78	0.81	0.78	0.76	The default attributes available in this dataset (NIDS).
Zhang et al. [8]	0,79	0,82	0,97	0,76	The encoder of DL auto-encoder (1N-encoding) used to compress the less important features and extract key features without decoder.
Tauscher et al. [9]	0,80	N/A	N/A	0,78	The default attributes available in the dataset.
Liu et al. [10]	0,78	0,78	0,78	0,75	The default attributes available in the dataset.
Sharmila et al. [97]	0.86	N/A	N/A	N/A	Attribute reduction method PCA applied on the dataset.
Prachi et al. [100]	0.99	N/A	N/A	N/A	Wrapper + Filter Methods to feature selection.
Kazi et al. [104]	0.94	N/A	N/A	N/A	Wrapper Method to feature selection.

Table 4.11: Comparison with the state of art based on NSL-KDD dataset.

As we can see in Table 4.11, our solution significantly enhances the results of the existing solutions especially in terms of classification accuracy. In addition, it reduces the preprocessing time applied to the raw data compared to some existing solutions. For instance, in [8], the authors run a deep learning algorithm to extract the most relevant features for classification, which is a time consuming task compared to our preprocessing phase that yet allows to achieve better classification results. Moreover, our solution also reduces the model execution time compared to the remaining solutions, since it uses less features for the training (12 in ours, 17 in [104], 122 in [8], 41 in [7], [10] and [9]).

4.4.3 Complementary experiments with UNSW-NB15 dataset

Dataset description

The UNSW-NB15 dataset serves as a network traffic database designed for intrusion detection. It was developed within the Cyber Range Lab of the Australian Centre for Cyber Security, utilizing the IXIA PerfectStorm tool. The primary aim was to generate normal activities alongside simulations of recent cyber attacks.This dataset encompasses nine attack categories, along with a category dedicated to flows considered as normal. In total, it comprises 2,540,043 entries spread across four CSV files. Furthermore, two additional pre-processed files are also provided, each containing 257,673 entries used in the training and testing sets. The significant distinction between these datasets and the CSV files we possess is that the applied pre-processing has already removed certain features (`srcip`, `dstip`, `sport`, `dsport`, `Stime`, `Ltime`) essential to our feature engineering. Each entry consists of 49 fields detailing various network connection characteristics, including source and destination IP addresses, ports, protocols, etc. Additionally, two fields are specifically designated for labeling (binary and multi-class). The UNSW-NB15 dataset is frequently

employed as a benchmark to assess the performance of intrusion detection systems based on machine learning [138]. It also stands as a valuable resource for research in network security and network traffic analysis.

For the implementation of our approach, we are working with the 4 unprocessed CSV files. We acknowledge that the effectiveness of a model trained within one infrastructure may not be transferable to another, due to factors such as IP addresses and thresholds. However, we operate on the assumption that a model will have a context and will be specifically associated with a given infrastructure. One field, `id`, is clearly redundant as it corresponds to the entry index. Therefore, we have 44 features; 2 of them are labels (binary and multi-class), 3 are nominal, and the remaining 39 are numeric.

Flow-class	Tactics	Techniques	Count(#rows)
Normal	-	-	2218760
Generic	TA0005/TA0011	T1600/T1573	215481
Exploits	TA0002/TA0005/TA0042	T1203/T1211/T1587.004	44525
Fuzzers	TA0042	T1498	24246
DOS	TA0040	T1498 - 1499	16353
Reconnaissance	TA0043	Any	13987
Analysis	TA0001/TA0007	T1566/T1046	2677
Backdoor	TA0042	T1488	2329
Shellcode	TA0005	T1620	1511
Worms	TA0001	T1078-T1133	174

Table 4.12: The UNSW-NB15 dataset content.

Data processing

The highly disparate data requires transformation into binary representations by achieved through *OneHotEncoder* and *LabelEncoder* for object-type fields (`srcip`, `dstip`, `service`, `proto` and `state`), as implemented by Tauscher et al. in [9]. A quick observation reveals that the `attack cat` field has only 22,215 entries, indicating a lack of labels for all attack categories. To address this, entries where `attack cat` is `NaN` are labeled as NORMAL. Anomalies, such as the error in the value of the `ct src ltm` field, are identified.

Fields `sport` and `dsport` containing hexadecimal values and "-", have corresponding entries removed, with hexadecimal values converted to base 10. Three fields have empty or "None" values. To balance the data, only 14% of entries with label 0 (normal) and a quarter of entries with the label `Generic` are retained. The dataset exhibits errors such as inconsistent spaces in labels, for instance, `Shellcode`, `Shellcode  Shellcode`. Corrections, including space removal, are applied to the CSV files. The application of our feature engineering, based on the definition of a network flow, enables us to extract the 17 features presented in Table 4.13, instead of the default 47 attributes

We concatenate the training and test data to ensure a random distribution of entries. Subsequently, we perform a split of 70% for the training set and 30% for the test set, thereby applying the model defined in Table 4.2.

Feature	Description
srcip	source IP of the traffic.
sport	source port.
dstip	destination IP of the traffic.
dsport	destination port
proto	Network protocol used for the communication.
state	State of the connection.
dur	Duration of the connection in seconds.
sbytes	Source to destination bytes.
dbytes	Destination to source bytes.
rate	Data transfer rate.
sttl	Source time to live.
dttl	Destination time to live.
sloss	Source packets retransmitted.
dloss	Destination packets retransmitted.
service	Type of network service.
Sload	Source bits per second.
Dload	Destination bits per second.
ct-srv-src	Source connection count for the same service.
ct-srv-dst	Number of connections observed for the same service and destination IP address.
ct-src- ltm	Number of connections observed for the same source IP address.
ct-dst-ltm	Number of connections observed for the same destination IP address.
attack-cat	The category of the attack.
Label	Binary label indicating whether the record is normal (0) or an attack (1).

Table 4.13: Our Feature engineering on UNSW-NB15.

Binary classification (with LabelEncoder and OneHotEncoder)

Both models yielded very similar results in terms of the figures derived from the classification report. However, a more detailed analysis of the curves reveals significant differences. When using the `OneHotEncoder`, our model encountered an EarlyStop as early as the 8th iteration. The loss function failed to decrease, while the validation performance showed a pronounced upward trend. Regarding accuracy, it stabilized from the 3rd iteration before declining sharply by the 8th (losing 0.001, which is five times the gain observed between the first and 7th iterations).

Similarly, our model employing the `LabelEncoder` stopped at the 10th epoch, swiftly stabilizing its accuracy by the 6th epoch. We noted a significant number of false negatives and very few false positives (5) with the LabelEncoder. Ideally, we aim to achieve zero false positives. This outcome is particularly gratifying as, in our context, minimizing false positives takes precedence over false negatives.

-	Precision	Recall	F1-Score	Support
Normal	1.00	0.99	0.99	47605
Attack	0.99	1.00	0.99	47842
Accuracy			0.99	95447
Macro AVG	0.99	0.99	0.99	95447
Weighted AVG	0.99	0.99	0.99	95447

Table 4.14: Binary Classification with LabelEncoder.

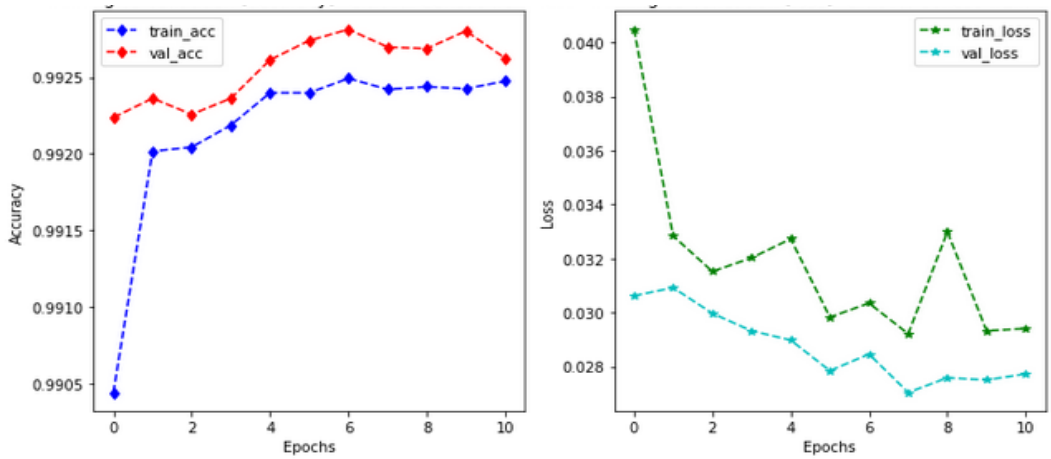

Figure 4.9: Binary Classification with LabelEncoder.

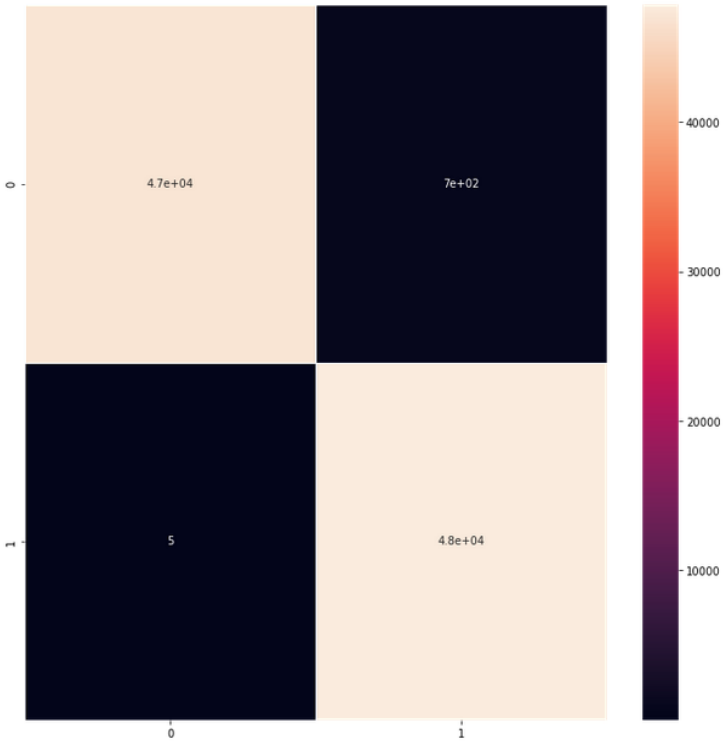

Figure 4.10: Confusion Matrix with LabelEncoder.

Target	Precision	Recall	F1-Score	Support
Normal	1.00	0.99	0.99	47605
Attack	0.99	1.00	0.99	47842
Accuracy			0.99	95447
Macro AVG	0.99	0.99	0.99	95447
Weighted AVG	0.99	0.99	0.99	95447

Table 4.15: Binary Classification with OneHotEncoder.

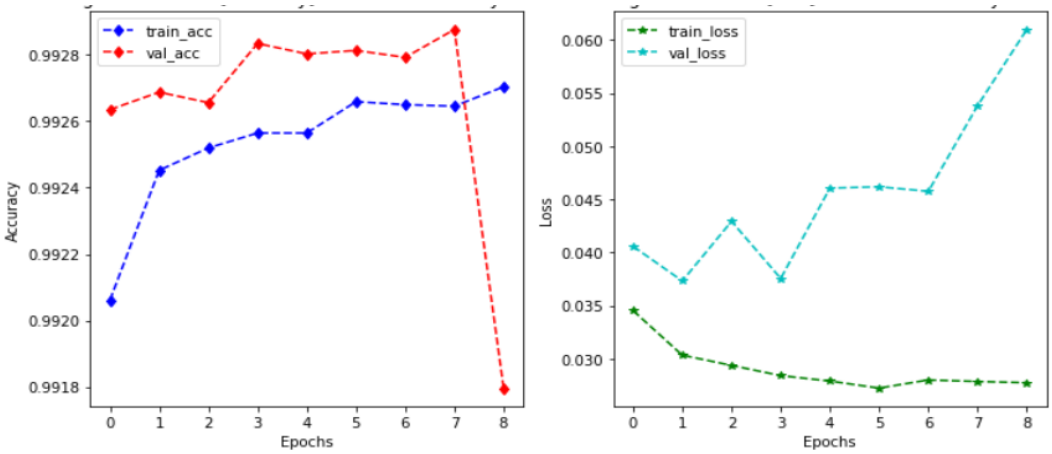

Figure 4.11: Learning Curves of Binary Classification with OneHotEncoder.

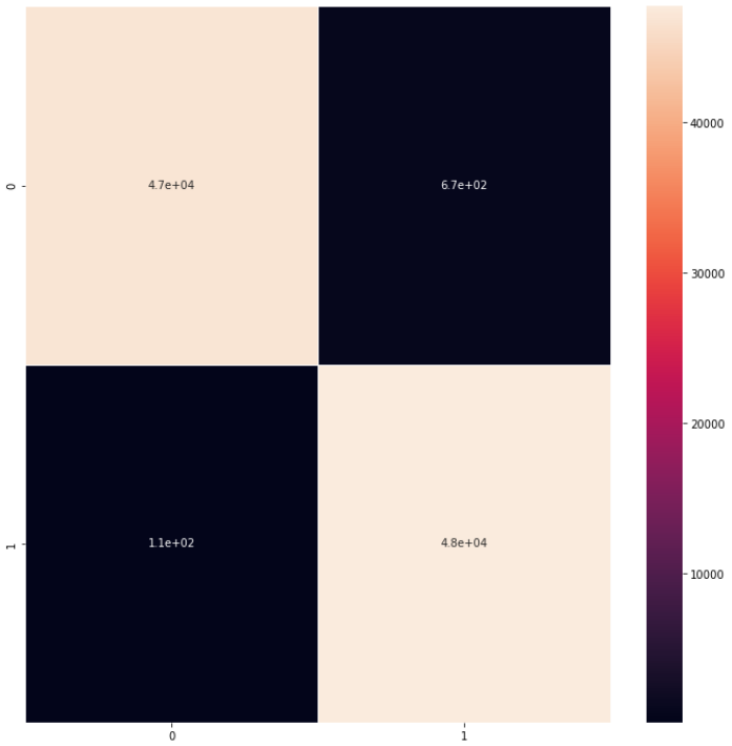

Figure 4.12: Confusion Matrix with OneHotEncoder.

Multi-class classification (with LabelEncoder and OneHotEncoder)

In the realm of multi-class classifications, initial learning stages naturally yield significant gains in both accuracy and loss minimization for both training and validation data. However, a stabilization in accuracy and loss occurs around the 15th epoch, suggesting potential for further learning given the absence of early stopping criteria (defined as three consecutive non-improving losses). We observe no significant difference in accuracy between data processed with LabelEncoder and OneHotEncoder. Generally, F1-Scores for different labels are also very similar. Notably, the 'Normal' and 'Generic' labels exhibit excellent F1-Scores, followed by high scores for 'Fuzzers', 'Reconnaissance', 'Exploits', and 'Shellcode'. Conversely, 'DoS',

'Analysis', and 'Worms' display low F1-Scores, with 'Backdoor' being almost absent.

Target	Precision	Recall	F1-Score	Support
Normal	1.00	0.99	0.99	47605
Generic	1.00	0.97	0.99	16146
Exploits	0.61	0.92	0.74	13355
Fuzzers	0.86	0.85	0.86	7243
DoS	0.43	0.15	0.23	4973
Reconnaissance	0.90	0.66	0.76	4139
Analysis	0.49	0.16	0.24	786
Backdoor	0.35	0.02	0.03	681
Shellcode	0.76	0.75	0.75	457
Worms	0.67	0.06	0.12	62
Accuracy			0.89	95447
Macro AVG	0.71	0.55	0.57	95447
Weighted AVG	0.89	0.89	0.88	95447

Table 4.16: Multi-class Classification with LabelEncoder.

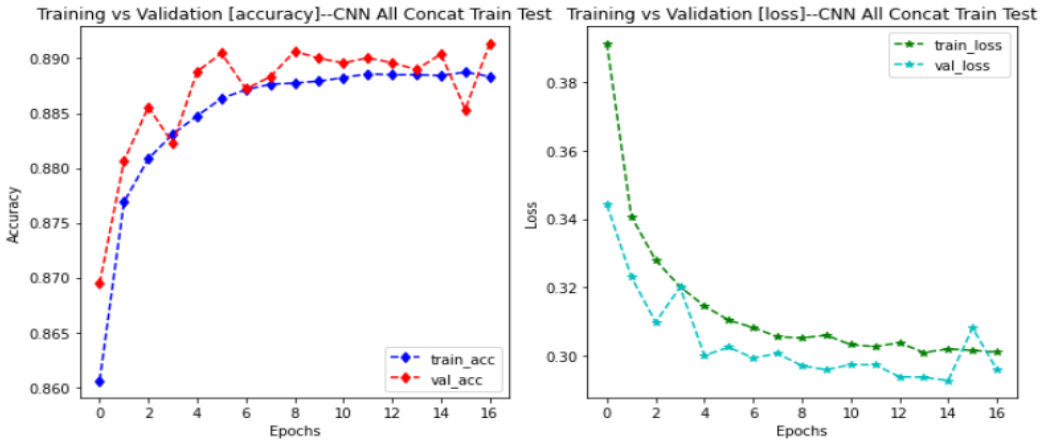

Figure 4.13: Learning Curves of Multi-class Classification with LabelEncoder.

Target	Precision	Recall	F1-Score	Support
Normal	1.00	0.99	0.99	47605
Generic	1.00	0.97	0.99	16146
Exploits	0.61	0.92	0.74	13355
Fuzzers	0.86	0.85	0.86	7243
DoS	0.43	0.15	0.23	4973
Reconnaissance	0.90	0.66	0.76	4139
Analysis	0.49	0.16	0.24	786
Backdoor	0.35	0.02	0.03	681
Shellcode	0.76	0.75	0.75	457
Worms	0.67	0.06	0.12	62
Accuracy			0.89	95447
Macro AVG	0.71	0.55	0.57	95447
Weighted AVG	0.89	0.89	0.88	95447

Table 4.17: Multi-class Classification with OneHotEncoder.

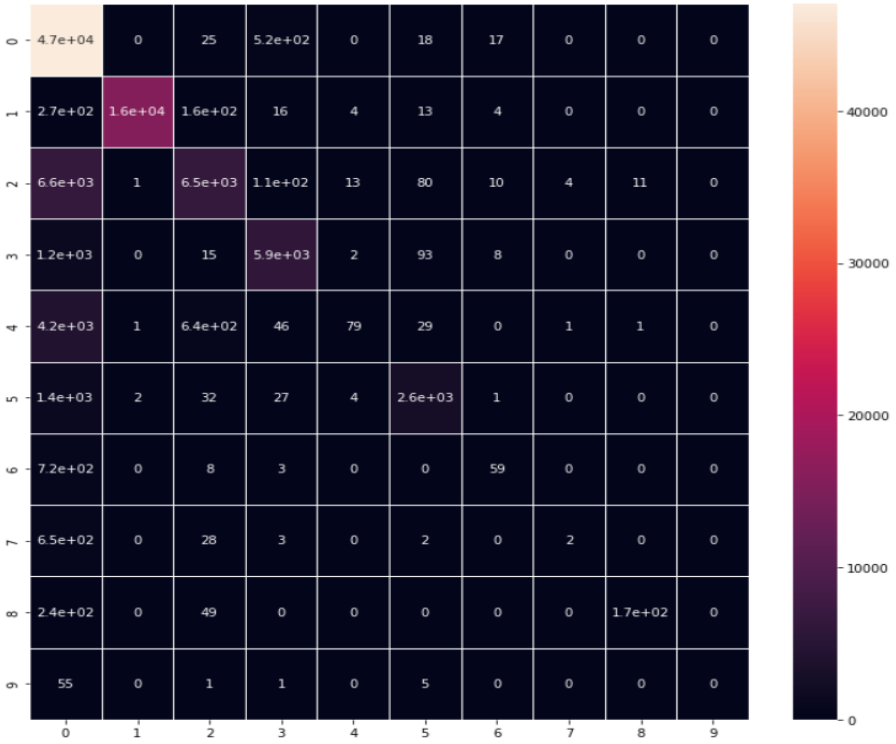

Figure 4.14: Confusion Matrix of Multi-class Classification with LabelEncoder.

It is important to note that the decrease in accuracy does not correlate with the number of corresponding entries, as evidenced by `DoS` with an F1-Score of 0.37 for 4973 predicted entries and 'Shellcode' with 0.78 for 457 entries, despite the same training proportion. This divergence may stem from the predefined thresholds for DOS attacks and the selected features. The threshold defined for DOS attacks may be reached by other attack types.

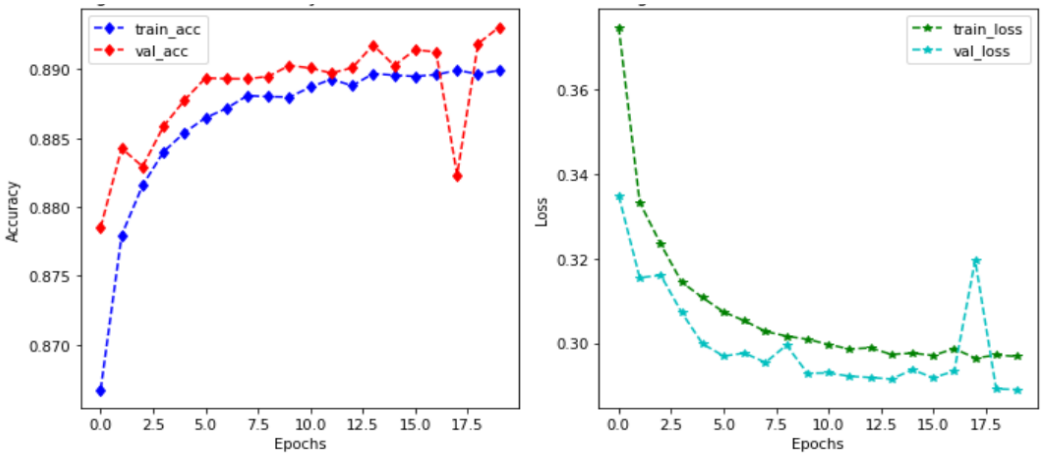

Figure 4.15: Learning Curves of Multi-class Classification with OneHotEncoder.

Regarding confusion matrices, discrepancies emerge between predictions made using LabelEncoder and OneHotEncoder. In the former case, some predictions label attacks as 'Normal', and labels 4, 6, 7, and 9 are rarely predicted correctly, mostly classified as 'Normal'. Predicting 'Normal' instead of another attack label poses a

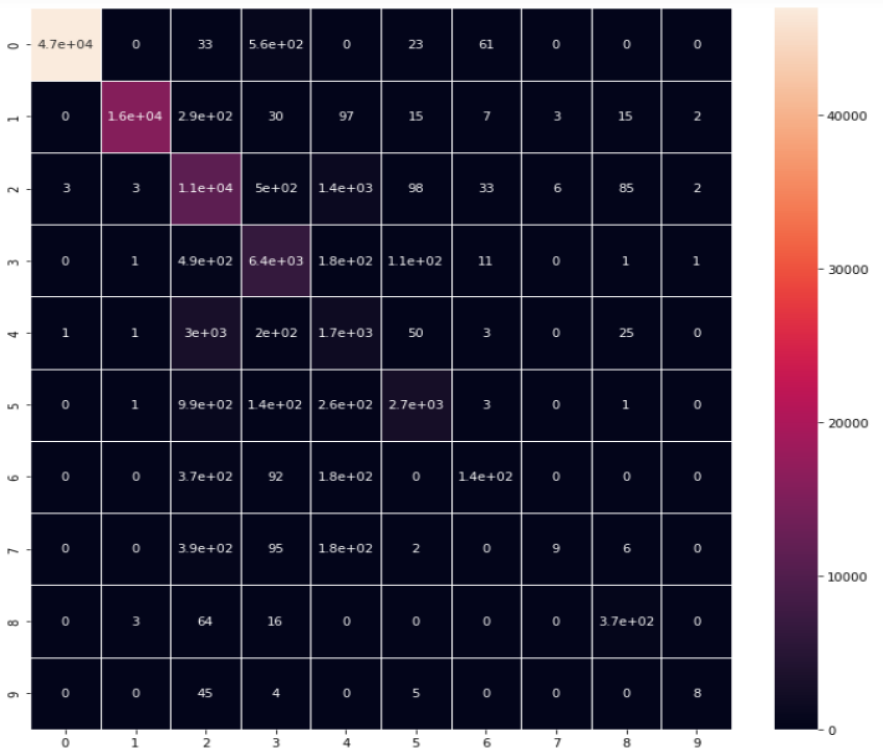

Figure 4.16: Confusion Matrix of Multi-class Classification with OneHotEncoder.

greater problem as it gives no indication of an attack, unlike misclassifying as another type of attack, which still categorizes it as illegitimate.

4.5 Conclusion

In this chapter, we have presented an intrusion detection solution for network flow collectors using 1D-CNN. First, our solution presents a feature engineering for the extraction of data features according to the definition of a network flow. The proposed engineering can be used in any network flow classification, since it is based on criteria that can be identified in any network event. Our feature extraction is then associated with a feature detector functionality defined in CNN, which guarantees an accurate determination of normal network flows and a robust multi-class classification of malicious ones. Our model has been evaluated on different contexts (industrial and public datasets). The first dataset has been extracted from a real IBM industrial environment while the the two others are public: NSL-KDD and UNSW-NB15. Both sets were validated using the MITRE ATT&CK framework, and the results have proved that our model distinguishes normal behaviors from deviant ones, and efficiently identifies the attack classes. Moreover, our model significantly improves the accuracy of the classification process when it is compared to other existing solutions of the state of the art. It also reduces the number of extracted features and thus the execution time of the global model compared to existing solutions. However, it does not ensure unknown attacks detection.

In the upcoming chapter, we intend to focus on zero-day vulnerability exploitation attacks.

Chapter 5

A framework for detecting zero-day exploits in network flows

Contents

5.1 Introduction

Zero-day attacks entail the exploitation of unknown system vulnerabilities within an information system [121]. The proposal of a zero-day attack detection solution aims to proactively identify unknown threats that assets within an information system may encounter.

The zero-day attack detection solutions presented in the literature mainly focus on anomaly detection using various techniques such as linear transformations [122, 123], statistical measures to identify significant deviations in network traffic [11], auto-encoders [13, 127] and deep neural networks [131]. Additionally, semi-supervised methods [129] and hybrid approaches [128] combining clustering techniques and various algorithms are also proposed in the literature. Other solutions

rely on LSTM networks to model unknown vulnerabilities and multi-stage attacks [130].

However, most solutions primarily focus on optimizing the proposed models, which can result in a high false positive rate from one context to another. Moreover, some solutions focus on detecting a specific class of attacks and are not capable of detecting all types of zero-days. Finally, sophisticated attacks may evade anomaly detection. To address these challenges, we present, in this chapter, a framework for detecting zero-day attacks that escape current detection systems. This zero-day detection method relies on enhanced identification and qualification of attacks already cataloged in an existing database, proving crucial in identifying unknown attack patterns. Our second contribution is hybrid and comprehensive, encompassing various intrusion detection phases from data collection to detection, leveraging a combination of supervised and unsupervised algorithms along with anomaly identification methods for real-time data feeds into a network flow collector. We evaluated our protocol using two distinct datasets: the first extracted from a real industrial context, referred to as the IBM dataset and the second sourced from the state-of-the-art, the NSL-KDD. The evaluation results highlight the detection of anomalies present in newly introduced data that were not learned by the detection system.

5.2 Our proposal

In our zero-day detection protocol, we propose a five-phases approach to address the ever-evolving landscape of security threats related to network flows. These phases are mainly designed to play a critical role in data collection, data classification, data clustering, establishing correlation and detecting anomaly.

Figure 5.1 highlights the various phases of our framework outlined in Section 5.2, from data collection to zero-day attack identification.

In our approach, we start by centralizing the network flows to be analyzed, creating training and evaluation datasets, establishing a knowledge base of known attack scenarios, and continually feeding this base with new attack scenarios through anomaly detection for the new data the system continues to receive and process. In what follows, we describe the different phases of our protocol.

5.2.1 Data collection phase

The first phase of our framework involves the collection of data related to the network traffic of the system. This phase requires a flow collector to capture packets passing through the network and its surroundings. The diagram in Figure 5.2 illustrates the network flow collection options (internal and external traffic), as proposed by IBM Security's architecture [160].

The collector retrieves flow data from raw packets collected via monitoring and mirroring ports [161] such as SPAN, TAP, sessions supervising, or from external flow sources such as NetFlow [162], sFlow [163], and jFlow [164]. The data from these various internal and external sources are centralized in a dedicated flow collector. Following this, we apply the feature engineering as presented in Section 4.3.1 to reduce the input data dimensionality.

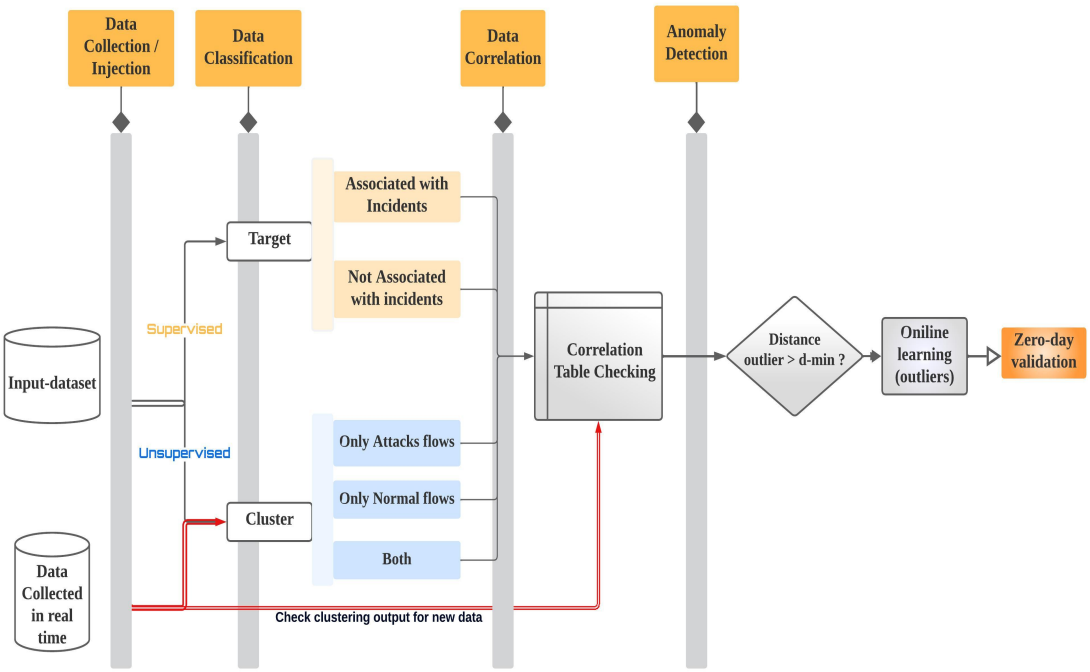

Figure 5.1: Our proposed zero-day detection approach.

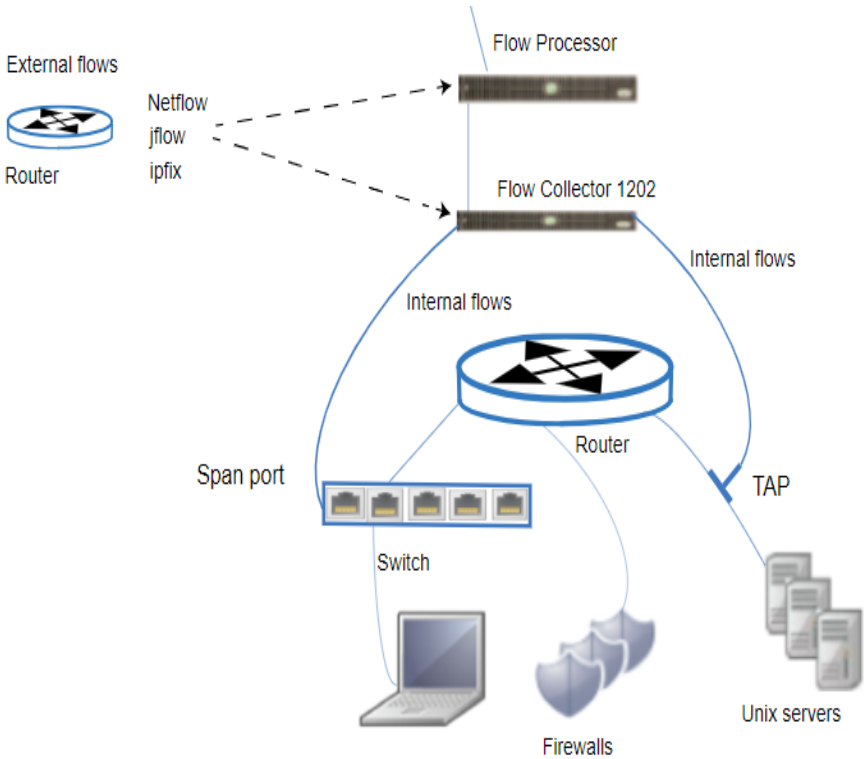

Figure 5.2: Data collection.

Then, we extract multiple pieces of information from the collected data. Consequently, the `input-dataset` consist of attributes based on the three defined families of characteristics needed to classify network flows. In Table 5.1, we present the three feature families described in 4.1.

Flow Characteristics	
Flow-ID	Flow Type
Flow-Aggregation-Count	Flow Direction
Flow-Bias	Flow Duration
Asset Characteristics	
Source-IP	Source-Port
Destination-IP	Destination-Port
Protocol	Category
Application	Application-Group
Data and Speed Characteristics	
Source-Bytes	Destination-Bytes
Bit-Per-Second	Total Bytes

Table 5.1: The classification of network flow features into different families.

Once these characteristics are defined, the `input-dataset` is composed by identifying and selecting attack scenarios flows experienced by the information system, along with a set of legitimate flows identified at various periods (morning activities, business hours, evening activities, weekends, and so on). So, the `input-dataset` must be representative of the various data variants collected and centralized within the flow collector.

As depicted in Figure 5.3, these meticulously curated data fuel both supervised and unsupervised models.

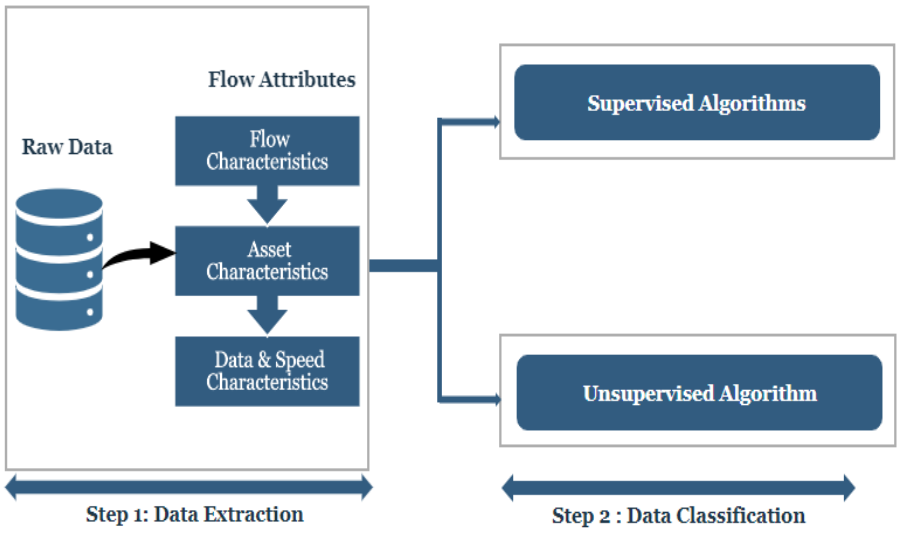

Figure 5.3: Our proposed model for data categorization.

5.2.2 Supervised classification phase

Once the `input-dataset` is formed, the next phase of our framework consists of a supervised classification of network flows. This means associating each flow with the appropriate class (legitimate flow or malicious flow).

We use deep learning-based classification, acquiring specific features during the learning phase. Notably, we leverage the CNN for its feature detection capabilities within the convolutional layer, alongside other DL approaches. This feature detector detects the features that match the highest values of the `input-dataset` in the feature map. A convolution operation is then performed between the `input-dataset` and the feature detector to determine a feature map that is representative of the relevant data. This model aims at coupling this feature map building process with the feature engineering presented in Section 4.3.1, aiming to define a robust method for classifying the network flows.

Thus, the feature map is injected into the CNN's other layers (pooling, flattening and fully connected) in order to determine the output categories known as the `Target-Set`:

- **Target Not Associated with Incident:** which describes a first category of flows not associated with attacks (normal or legitimate traffic).
- **Targets Associated with Incidents:** made up of malicious activities related to different attack scenarios.

We use a One-Dimensional CNN (1D-CNN) approach with parameters for `nb-filters` filters to extract `nb-filters` different features on the first convolution layer of the network.

Once the `Target-Set` is defined, we move to the boosting step to decrease the FDR (false positives, false negatives). To achieve this, the other algorithms used include DT, RF, KNN, and Bernoulli NB. In our model, DT and RF are used because of their effectiveness in classifying nonlinear data, given the highly varied and complex attack patterns of network flows. KNN enable us to define complex boundaries using its proximity-based method, which is useful as we have a variety of normal flows and attacks.

Finally, the NB model, based on its assumption of conditional naivety, presumes that the features used for prediction are all independent of each other. This implies that the presence or value of one feature does not influence the others. This notion enable us to evaluate the relevance of the attributes proposed in the `input-dataset`. Thus, the boosting technique, a set-based learning method that combines a set of weak `Learners` (algorithms) into one strong learner, rely on the four other algorithms.

An implementation of our CNN-based solution and the four other ML algorithms gives us an initial prediction of the different categories of network flows. We rely on the `F1 score` of predictions for each class. Assigning weights to these predictions as shown in Figure 5.4, we compare the weighted F1 scores, class by class, from each algorithm to determine an optimal prediction for each label (normal flow and attack flow). The CNN, serving as the foundational algorithm, holds precedence (a higher weight) over the other algorithms. The `Target-Set` be used to create a correlation Table 5.2.4, which serve as the basis for analyzing the formed clusters in the phase 5.2.3 and for online learning in the phase 5.2.5. This approach avoid running the supervised model entirely during real-time data collection and classification.

Depending on the dataset or the context, the optimal hyper-parameters may change, so we will not focus on this aspect of model optimization. Therefore, the four models defined in addition to the CNN are employed with the parameters commonly used.

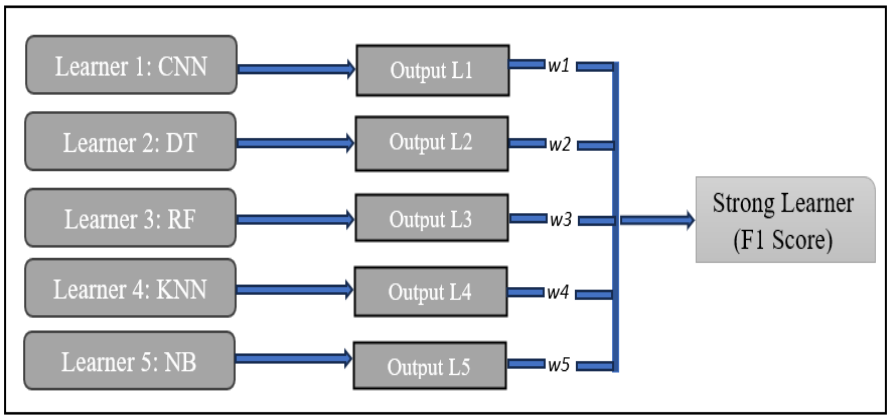

Figure 5.4: Boosting technique.

After completing the supervised classification phase, we transition to the next stage, where we employ unsupervised clustering using the robustness of the K-Means algorithm. This enables the grouping of similar segments within unlabeled data, unveiling concealed patterns within our network data.

5.2.3 Unsupervised classification phase

In this phase, we employ an unsupervised algorithm to identify data clusters in space, offering a visual representation of the input data. The network flows of data contain various attack classes and have a non-hierarchical structure. The K-Means algorithm is more suitable among unsupervised methods to address the clustering of these data types. Our application of the K-means algorithm adopts an approach where the final cluster centroids are defined randomly in the `input-dataset`. This decision was made to explore the algorithm's adaptability to the diverse network flow data. Indeed, choosing a random centroid configuration ensures a more comprehensive exploration of potential structures within the data, providing an unbiased perspective on the network flow clustering process. Consequently, this model allow us to divide our dataset into `K` distinct groups and extract patterns of malicious flows related to attack scenarios, legitimate flows or both. The Elbow method analysis allowed us to identify a point where the precision showed only marginal improvement with the addition of clusters. To enhance anomaly detection and achieve a finer segmentation of the data, we opted to double this point to attain a more detailed distribution, thereby better capturing the data intricacies crucial for anomaly detection.

At the completion of the various implementation phases of this algorithm, the clusters obtained form a set called the `Cluster-Set`.

There are three types of clusters built, considering that a network flow is either normal or abnormal. Thus, this flow could be positioned within one of the following groups:

- **Only Attack Flows** containing only attack data.
- **Only Normal Flows** clusters with only legitimate flows.
- **Both** are made up of normal traffic and attacks.

Next, we move on to the analysis stage of the formed `Cluster-Set` and essential metrics such as the intra-cluster distance `a(i)`, the inter-cluster distance `b(i)`, the

average distances d-mean, the maximum distance d-max, and most importantly, the silhouette score S-sil. All these metrics allow us to determine the minimum distance d-min per cluster that i belonging to a cluster should not exceed. We employ outlier distance analysis to compare this distance threshold for each cluster d-min with the new data that the flow collector continues to receive. Indeed, the set of point distances that exceed this d-min are termed outliers within a cluster. Subsequently, we evaluate Data Collected in real time to identify potential outliers, thus enhancing our ability to detect and respond to zero-day attacks in real-time.

In order to analyze and correlate the Target-Set defined at the end of the supervised approach (phase 5.2.2) and the Cluster-Set created in the phase 5.2.3, we create a join table of the two sets, called *Correlation Table*.

5.2.4 Correlation table phase

To analyze and exploit the results of unsupervised learning (K-Means), and also associate them with of the supervised approach, we create the correlation table. This table allows us to detect which output class from the supervised approach best corresponds to which cluster of points from the unsupervised approach. We recall that in our framework, each communication flow has a unique identifier, Flow-Id, as presented in Figure 5.5. Based on the flow identifiers and associated targets of the supervised approach, we look for the appropriate clusters. Therefore, for each network flow, we identify the target of the supervised approach with the cluster number of the unsupervised approach. Table 5.2 is an example of correlation table. This association of target and cluster, via the Flow-Id enables us to determine the best flow classes for the clusters and also to validate the predictions of the supervised approach. To do this, we determine the similarity ratios between the elements of a cluster and each target.

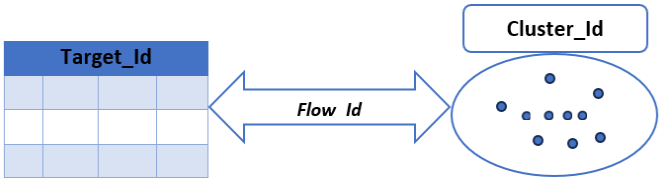

Figure 5.5: Correlation with flow Id.

Cluster-id	Target-id
a	25% target1 + 30% target2 + 45% target3
b	5% target4 + 95% target3
c	1% target2 + 8% target0 + 91% target5
d	100% target0

Table 5.2: An example of a correlation table.

The selection of the number of clusters K is deliberate to optimize the spatial representation of the data. Following the establishment of similarity ratios between the points within a cluster and each defined target, it is noteworthy that each cluster tends to be primarily associated with a single target, even though we have three types of clusters. And each target is represented by at least one cluster. Once this correlation table is defined, we employ outlier distance analysis to identify outliers.

5.2.5 Outlier detection phase

The distance measure serves to assess the proximity between new data points and existing ones. This is crucial in our framework aimed at identifying zero-day attack scenarios, often associated with distant points within established clusters.

Our input dataset comprises real-time data collected by the system, as presented in Figure 5.1. In this new configuration, the `Data collected in real time` is injected into the unsupervised approach without altering any hyper-parameters or the `Cluster-Set` itself. This decision is driven by the need to detect anomalies caused by abnormal distances within clusters.

To understand the data's structure and potential classes, we leverage the correlation table. Subsequently, we calculate the distance `d(i, ol)` between the positioning of the real-time data and the existing cluster elements for classification. As a reminder, any data point significantly distant from others in its assigned cluster is identified as an `outlier`. Determining whether to split a cluster depends on the calculated `d-min` and the presence of new points as outliers with distances exceeding the `d-min`.

Next, the online learning is applied to adapt the model to the remaining data points that are not outliers. Indeed, a zero-day attack class does not solely consist of outliers. It can also encompass data that does not reach the d-min threshold in terms of average distance. This learning approach enables our model to be regularly updated as new data becomes available. In that way, it be adapted in real time to the information provided by the information system [165]. Thus, the outliers in a cluster whose distance respects the defined `d-min` constitute the training data for a new attack class. The other new points in the same cluster that are not outliers are used as evaluation data for the overall model. This last evaluation is used to match the points of the new outlier clusters and validate the detection of zero-day attacks. To do this, we use the same learning algorithms: CNN and the four boosting algorithms described in Section 5.2.

We define `deg-th` the degradation threshold established for base models within the context of online supervised learning. This threshold indicates the acceptable level of performance degradation of the base models in terms of overall model accuracy.

The online learning model is retained if it remains below this degradation threshold, ensuring its stability and efficacy. However, if the base model experiences degradation beyond this threshold, it is not preserved or retained, indicating a significant deterioration in its performance.

5.3 Theorical analysis

As we have seen in Section 5.2.5, the outlier detection in our framework is decided based on whether the distance between the outlier and the cluster centroid exceeds a distance $d - min$. This section presents a method to determine this distance on the basis of the Silhouette score.

First, let us start by presenting some fundamental principles allowing the optimization of intra-cluster coherence and inter-cluster dissimilarity in the K-means

clustering algorithms:

- **The cluster center** is the average of all the features of the points belonging to that cluster. Let **Ci** be the center of cluster C, **Ni** be the number of points in cluster C, and **xij** be the coordinates of the j-th point in cluster C:

$$C_i = \frac{1}{N_i} \sum_{j=1}^{N_i} x_{ij}$$

- **The Silhouette score** which quantifies the separation of clusters.
  1. For an individual data point i:

$$S_{sil}(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

  Where a(i) and b(i) represent the intra-cluster distance and the inter-cluster distances respectively for a given point i.
  2. For a cluster C:

$$S(C) = \frac{1}{|C|} \sum_{i \in C} S_{sil}(i)$$

Building upon the above definitions, we aim to ascertain the relevance of maintaining the outlier points within a specific cluster or not. To do so, we start by identifying the various scenarios for adding new points within a given cluster C.

- **Scenario 1:** The new elements added to the cluster C are very close the old elements of the cluster in terms of distance. Therefore, they do not significantly impact the intra-cluster and inter-cluster distances, causing, at worst, a minor degradation of the silhouette score. In our framework, we treat this scenario as one case where there is no evidence of detecting a new attack scenario. Therefore, we consider these new elements either as normal traffic or as a known attack pattern.
- **Scenario 2:** all the new elements, added to cluster C, are relatively far from the old elements of the cluster. Therefore, adding these relatively distant elements may (depending on their number) impact the intra-cluster distance, causing a significant degradation of the silhouette score. In our framework, we consider this degradation as an indicator of a new attack, which implies a restructuring of these root clusters.
- **Scenario 3:** at least one element added to the cluster C is far from the old elements in terms of distance, while others blend in with the existing elements. This scenario is common in evolving corporate networks due to the use of various business applications and employees not always rigorously adhering to the Information Security Management System (ISMS). Furthermore, the exploitation of zero-day vulnerabilities can also contribute to the occurrence of anomalous and responsive flows in this scenario.

In what follows, we present an analysis of the distance based on scenario 3, where we consider that we have at least one new element in the cluster that is far from the old

ones. Our analysis studies whether it is better to split the cluster or not if scenario 2 occurs. The results of our analysis can, however, be generalized to scenario 2, where all the elements added to the cluster are far from the old ones.

Case 1: Maintaining the cluster is the best solution.
In the case where the addition of a new outlier element ol in the cluster C does not imply the creation of a new cluster, we mainly observe a change in the intra-cluster distance $a(i)$. Therefore, by adding one element ol to the cluster, the silhouette score is represented as follows:

$$S(C) = \frac{1}{|C|} \sum_{i=1}^{|C-1|} \frac{b(i) - a'(i)}{\max\{a'(i), b(i)\}} + S_{sil}(ol) \tag{5.1}$$

with a new intra-cluster score $a'(i)$

$$a'(i) = \frac{|C-1| \cdot a(i) + d(i,ol)}{|C|} \tag{5.2}$$

We note that $|C|$ represents the cardinality of after adding one outlier, $a(i)$ represents the intra-cluster score before adding ol and $d(i,ol)$ is the distance between an old element i and the new element ol.

$$S(C) = \frac{1}{|C|^2} \sum_{i=1}^{|C-1|} \frac{b(i) \cdot (|C|) - a(i) \cdot |C-1|}{\max\{a'(i), b(i)\}} + \frac{d(i,ol)}{\max\{a'(i), b(i)\}} + S_{sil}(ol) \tag{5.3}$$

From the resulting formula (5.3) above, we define the function `f1(X)` of the variable X relative to the distance `d(i,ol)` and $S_{sil}(ol)$. This function represents the variation in the silhouette score of the cluster $S(C)$.

With

$$\max\{a'(i), b(i)\} = b(i) \tag{5.4}$$

$$f1(X) = S(C) = \frac{1}{|C|^2} \sum_{i=1}^{|C-1|} \frac{b(i) \cdot (|C|) - a(i) \cdot |C-1|}{b(i)} + \frac{d(i,ol)}{b(i)\}} + S_{sil}(ol) \tag{5.5}$$

Given (5.4), we draw X the variable in the formula (5.5), dependent on both the distance between each point i and the identified outlier, as well as the silhouette score of this outlier. This is how we obtain the formula (5.6).

$$f1(X) = S(C) = \frac{1}{|C|^2} \sum_{i=1}^{|C-1|} \frac{b(i) \cdot (|C|) - a(i) \cdot |C-1|}{b(i)} + X \tag{5.6}$$

When

$$max\{a'(i), b(i)\} = a'(i) \tag{5.7}$$

$$f1(X) = S(C) = \frac{1}{|C|^2} \sum_{i=1}^{|C-1|} \frac{b(i) \cdot (|C|) - a(i) \cdot |C-1|}{a'(i)} + \frac{d(i,ol)}{a'(i)} + S_{sil}(ol) \quad (5.8)$$

Based on (5.7), we get the variable X in formula (5.8).

$$f1(X) = S(C) = \frac{1}{|C|^2} \sum_{i=1}^{|C-1|} \frac{b(i) \cdot (|C|) - a(i) \cdot |C-1|}{a'(i)} + X \quad (5.9)$$

Case 2: Cluster splitting is the best solution.
This implies that the distance of the outlier significantly affects the silhouette score $S(C)$. Therefore, it is necessary to create a new neighboring cluster. In the case of cluster splitting, a variation is observable at the cluster inter-distance `b(i)`.

Continuously, based on the silhouette score formulas presented in 5.3, we get the silhouette score when the outlier significantly impacts the cohesion of the original cluster.

$$S(C) = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{d(i) - a(i)}{\max\{a(i), d(i,\text{ol})\}} \quad (5.10)$$

In this case, with the cluster splitting into two, the inter-cluster distance of the original cluster becomes the average distance `d(i,ol)` between different points `ii` and the outlier. Hence, Formula 3 is obtained. Function `f2(X)` is defined to assess the variation of the silhouette score in this second scenario of original cluster separation, based on the distance `d(i,ol) = X`.

With

$$\max\{a(i), d(i,\text{ol})\} = d(i,\text{ol}) = X \quad (5.11)$$

When the inter-distance $b(i)$ represents the maximum between $a(i)$ and $b(i)$, as defined in formula (5.11), it results in the following form for `f2(X)` presented in the formula (5.12).

$$f2(X) = S(C) = \frac{1}{|C|} \sum_{i=1}^{|C|} 1 - \frac{a(i)}{X} \quad (5.12)$$

When

$$\max\{a(i), d(i,\text{ol})\} = a(i) \quad (5.13)$$

And in the case where the maximum between the intra-distance and the inter-distance is the intra-distance $a(i)$, as depicted in the formula (5.13), then the function `f2(X)` appears as follows:

$$f2(X) = S(C) = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{X}{a(i)} - 1 \quad (5.14)$$

Once both functions are defined, we subsequently determine the minimum distance from which this cluster separation is appropriate. This d-min corresponds to the point of intersection between the two functions f1(X) and f2(X). Any distance X greater than this intersection point is considered an outlier of the original cluster and is used as the training basis for online learning, as described in Section 5.2.

5.4 Performance evaluation

In this section, we assess the effectiveness of our zero-day attack detection framework while also emphasizing improved detection of known attacks within the collected network flows, aiming to minimize the FDR. To do this, we apply the various phases of our framework, as described in Section 5.2, to two datasets: the IBM dataset extracted from a real industrial context and the NSL-KDD dataset from the literature. Therefore, we identify and collect network data associated with these two contexts. Then, we apply feature engineering coupled with the CNN-based model to these data, and use other algorithms for the boosting technique, in order to constitute the different classes of normal and attack flows. Next, we use an unsupervised approach based on the K-Means algorithm to classify the data in space and thus determine the data clusters. A correlation table is used to link classes of supervised flows and clusters of unsupervised flows. Finally, we analyze the notion of distance for new data injected into our framework and identify zero-day attacks to the model. We validate this anomaly detection using an online learning process, relying on supervised classification and its algorithms.

We define different evaluation cases for our framework that vary according to the evaluation data injected.

5.4.1 Evaluation settings

Our architecture was implemented using Python within the Jupyter Notebook environment, supported by a machine equipped with an Intel® Core™ i5-1135G7 11th generation processor and 16 GB of RAM, running on a 64-bit Windows 11 operating system. For modeling and learning tasks, we relied on scikit-learn for ML algorithms, NumPy and Pandas for data manipulation, Matplotlib and Seaborn for result visualization. TensorFlow was utilized specifically for implementing neural networks.

5.4.2 Framework phase 1: leveraging the IBM dataset

IBM QRadar is a security appliance that is built on Linux. QRadar allows for the collection, storage and correlation logs for the detection of security incidents. To prove the effectiveness of our approach, we extracted raw data from a real-time industrial context, with an IBM proprietary IDS probe (QRadar Network Insight, QNI), which extends QRadar by providing a detailed view of real-time network communications [158].

Our dataset includes data collected in 2023 and, thus, recent attack chains. The flows used in our classification have been extracted over a period of one week. This extraction contains legitimate flows and non-legitimate ones that we categorize using the framework MITRE ATT&CK [30]. This content is presented in Table 5.3.

Flow-class	Target-ID	Tactic	Technique	Size (#rows)
Normal	0	-	-	24854
Large-Leakage	1	TA0010	T1567	4253
Stealthy-leakage	1	TA0010	T1030	4254
Critical Indicator of Compromise	7	TA0001	T1189	2064
Traffic To Anonymization Service	2	TA0011	T1090	288
Traffic to a phishing domain	5	TA0001	T1566	759
High Inbound Emails from External Host	4	TA0011	T1071	481
Access to a certificate expired service	3	TA0042	T1588	305
Remote Flood (TCP)	6	TA0040	T1498	21

Table 5.3: IBM Dataset content validated with MITRE ATT&CK.

To evaluate our zero-day framework, we use two attack classes, the first with ID 6 and the second with ID 7.

For preprocessing and normalization, we perform transformations on our raw data to facilitate its processing by the learning algorithms we employ. Firstly, we do a clean-up by not keeping any data containing null values so as not to introduce bias into the model. The raw data has an average of 150 features by default. However, using all these features for classification is not an optimal approach in terms of execution time. Therefore, in our solution, we reduce the number of features to 14 (at most) according to our feature engineering approach (Section 4.3.1) in addition to one target that describes the flow class.

After this extraction, all decimal entries (e.g., rate, amount of data, source / destination port numbers) remain unchanged. On the other hand, each byte in a field related to an `IP address` is converted to its hexadecimal value (excluding the dots that separate the bytes). After that, these values are concatenated, which gives us a unique value that is converted to a decimal. The field related to the `flow direction` has at most four possible values in the raw data; those relating to the `flow bias` have at most five possible values, and the type of communication protocol used is TCP or UDP. We use *OneHotEncoder* on these few non-numerical features to create a binary vector (composed of 0s and 1s) for each possible category or class in the categorical variable. Although the field values are now all numerical, they have very different ranges, so scaling or normalization is needed to make the data comparable and bring them down to a common scale. We have therefore carried out various normalizations on the data: standardization, minmax, tanh, log-normal. The different experiments performed have shown that normalization with Standardization allows us to obtain the best results later on, because we have better data separation, which is very useful in our case.

To determine the quality of our classification model and to evaluate its performance against various existing learning algorithms, we used the four popular evaluation metrics presented in Section 4.4.1 namely *Accuracy*, *Precision*, *Recall* and $F1-Score$.

Secondly, the silhouette score S_{sil} of a cluster and a set of clusters presented in Section 5.3 is used to assess the quality of the clusters defined, to determine the minimum distance from an outlier necessary to split a cluster, and therefore to identify a zero-day.

Our input-dataset consists of eight classes. We identify attack classes that are used to evaluate the zero-day detection capability of our framework. In the upcoming steps, we exclude the attack class that we define as the zero-day to be detected. This attack class represents our `Data collected in real time`. Thus, instead of eight classes, the model initially processes seven classes: a first class of legitimate flows with ID 0, and seven classes of attack flows with IDs from 1 to 7. Targets 6 and 7 represent `Data collected in real time` that we detect using the framework.

5.4.3 Framework phase 2: building the Target-Set

In the initial stage of evaluating our zero-day detection framework, we begin by forming the `Target-Set`. To achieve this, we apply a multi-class classification to our dataset (Table 5.3), defined after the feature engineering, the preprocessing and the normalization.

As a reminder, the F1 score represents an arithmetic average of precision and recall. It serves as a metric to measure the ability of learning algorithms to effectively associate flows with legitimate traffic classes or their corresponding target attacks.

The `input-dataset` is directly fed into the CNN algorithm, which forms the basis for detecting known attacks, as described in Phase 5.2.3 of Section 5.2 .

Target	**Our CNN model (zero-day 6)**				**Our CNN model (zero-day 7)**			
	Precision	Recall	F1-score	Support	Precision	Recall	F1-score	Support
0	0.97	1	0.98	4491	0.99	0.99	0.99	4466
1	1.00	1.00	1.00	2798	1	1	1	2857
2	0.86	0.94	0.90	64	0.86	0.96	0.91	51
3	0	0	0	69	0.75	0.81	0.78	54
4	0.95	0.65	0.77	150	0.97	0.69	0.80	159
5	1.00	1.00	1.00	95	1	1	1	89
6	x		x	x	1	1	1	7
7	0.99	0.99	0.99	420	x	x	x	x
Accuracy	98%				99%			

Table 5.4: The multi-class classification with our CNN model.

As observed in Table 5.4, the CNN yields an average F1 score of 0.49 for target 3, representing network traffic with an expired digital certificate. The low precision in this class is attributed to the neural network's operation, and the selected generic features lack information about the validity or expiration of a certificate. Consequently, the convolution layers struggle to effectively detect this attack class. A lower F1 score is noted for target 4, with an average of 0.78, indicating potential external phishing emails. The neural networks seem to be confused between these two attack targets. On a positive note, the results for the other targets are very promising, with F1 scores ranging between 0.90 and 1 in both executions and an overall precision ranging from 98% to 99%.

For effective detection of our zero-day attacks, we require improved detection of known attacks, implying a low FDR. Indeed, given the low F1 scores of targets 3 and 4, and that of target 2, we combine the results of the CNN algorithm with four supervised ML algorithms through a boosting technique based on the F1 score.

As observed in Tables 5.5 and 5.6, the results demonstrate that all the algorithms have achieved exceptional performances, with 100% accuracy for DT and RF. This

Target	DT			RF			BernoulliNB			KNN		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
0	1	1	1	1	1	1	0.98	0.98	0.98	0.99	0.99	0.99
1	1	1	1	1	1	1	1	1	1	1	1	1
2	0.98	0.97	0.98	1	1	1	0.73	1	0.84	0.94	0.98	0.96
3	1	0.99	0.99	1	0.99	0.99	0.59	0.97	0.74	0.88	0.97	0.92
4	0.98	0.99	0.98	0.99	0.97	0.98	0.91	0.13	0.23	0.89	0.82	0.85
5	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	0.95	1	0.97	0.98	0.99	0.99
Accuracy	100%			100%			97%			99%		

Table 5.5: Classification with boosting algorithms for zero-day 6.

suggests the remarkable ability of the chosen models to generalize well and correctly classify each category. Bernoulli NB exhibits slightly lower performance, with an overall precision of 97%. This is attributed to the probabilistic nature of the algorithm, which assumes conditional independence between features, an assumption that may not fully align with the complex relationships present in the data. As for KNN, it also displays excellent performance with an overall precision of 99%, showcasing its ability to adapt well to the data structure by utilizing the spatial proximity of instances.

Target	DT			RF			BernoulliNB			KNN		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
0	1	1	1	1	1	1	0.97	0.98	0.98	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	0.71	1	0.83	0.91	0.96	0.93
3	0.98	1	0.99	1	0.98	0.99	0.57	0.96	0.72	0.93	0.94	0.94
4	0.97	0.98	0.98	0.99	0.99	0.99	0.97	0.21	0.35	0.94	0.89	0.92
5	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1
Acc	100%			100%			97%			99%		

Table 5.6: Classification with boosting algorithms for zero-day 7.

The boosting technique aims to leverage different classification algorithms to reduce mispredictions and capture the specificities of each class, which can vary from one algorithm to another. Combining the results from Tables 5.4, 5.5 and 5.6, we achieve a maximum F1 score of 1 for targets 0, 1, 2, 5, and 6, and a score of 0.99 for the remaining targets 3 and 4. These classification results ensure a better distinction between legitimate flows and attacks, leading to the effective detection of known attacks with negligible FDR.

5.4.4 Framework phase 3: Cluster-Set building

In the second phase of our framework, we form the `Cluster-Set`. To achieve this, we apply the unsupervised K-Means algorithm to our `input-dataset`, with two executions corresponding to each zero-day attack.

For both scenarios, we obtain a cluster count of 25 using the Elbow method. The S_{sil} is 0.53 when the zero-day is the `Critical Indicator of Compromise`

attack and 0.54 when the zero-day is the *DOS* attack. In an effort to better utilize space and cluster learning, the cluster count is doubled from 25 to 50 in both scenarios. Therefore, our final `K` value is 50.

In Table 5.7, we present the top 5 clusters with the highest number of entries `Count`. Subsequently, we determine the intra-cluster distances `a(i)` and inter-cluster distances `b(i)`, from which the silhouette score `Score Sil` of each cluster in our unsupervised learning is calculated. These results highlight the data's structure.

Top 5 Cluster - zero day 6					Top 5 Cluster - zero day 7				
Cluster	Score Sil	a(i)	b(i)	Count	Cluster	Score Sil	a(i)	b(i)	Count
C4	0.766	0.448	2.058	3698	C7	0.781	0.466	2.297	3740
C12	0.528	0.650	1.403	2386	C10	0.316	1.168	1.754	2543
C1	0.281	1.100	1.659	2350	C3	0.487	0.697	1.393	2484
C35	0.337	1.094	1.699	2289	C30	0.296	1.195	1.746	2441
C0	0.364	0.958	1.479	2065	C22	0.326	1.536	2.346	2263

Table 5.7: Clustering scores for zero-day 6 and 7.

The silhouette score provides an initial indication of the clustering quality. Clusters C4 (ZD-6) and C7 (ZD-7) stand out with high scores (0.766 and 0.781, respectively), indicating a clear separation of elements within the clusters and a distinct demarcation between clusters. These results suggest a well-defined intrinsic structure within the data.
Furthermore, the intra-cluster and inter-cluster distances provide additional insights into the internal cohesion of clusters and their external separation. Despite having similar silhouette scores, clusters C4 and C7 exhibit distinct characteristics. C4 (ZD-6) displays lower internal cohesion a(i) but a more pronounced external separation b(i), while C7 (ZD-7) shows higher internal cohesion and equivalent external separation. These nuances reflect the complexity of the underlying data structures.
The size of clusters, reflected by the number of elements they contain, is a crucial aspect. Clusters C4 and C7 stand out not only for their metric performance, but also due to their significant size (3698 and 3740 elements, respectively). This indicates a strong representation of these clusters within their respective datasets. These values per cluster subsequently allow us to calculate the maximum distance `d-min` from which it is necessary to decide whether to separate a cluster or not, as defined in the theoretical analysis (Section 5.3).

We recall that function `f1(X)` represents the variation in the silhouette score of the original cluster when new elements are retained within it, while function `f2(X)` describes the variation when the original cluster must be split. Figure 6 illustrates an example of identifying the minimal distance d-min (the intersection point of the two functions `f1(X)` in orange and `f1(X)` in blue) between clusters 31 and 40 for zero-day 6, and clusters 12 and 32 for zero-day 7.

5.4.5 Framework phase 4: correlation table

In Table 5.8, we display the clusters in which the two evaluation data were positioned and also the targets corresponding to these clusters.

Now, we can proceed to detect zero-day attacks by injecting new data or streams that have never been learned by the models in the previous phases. Two classes of attacks have been defined, and these are tested one by one.

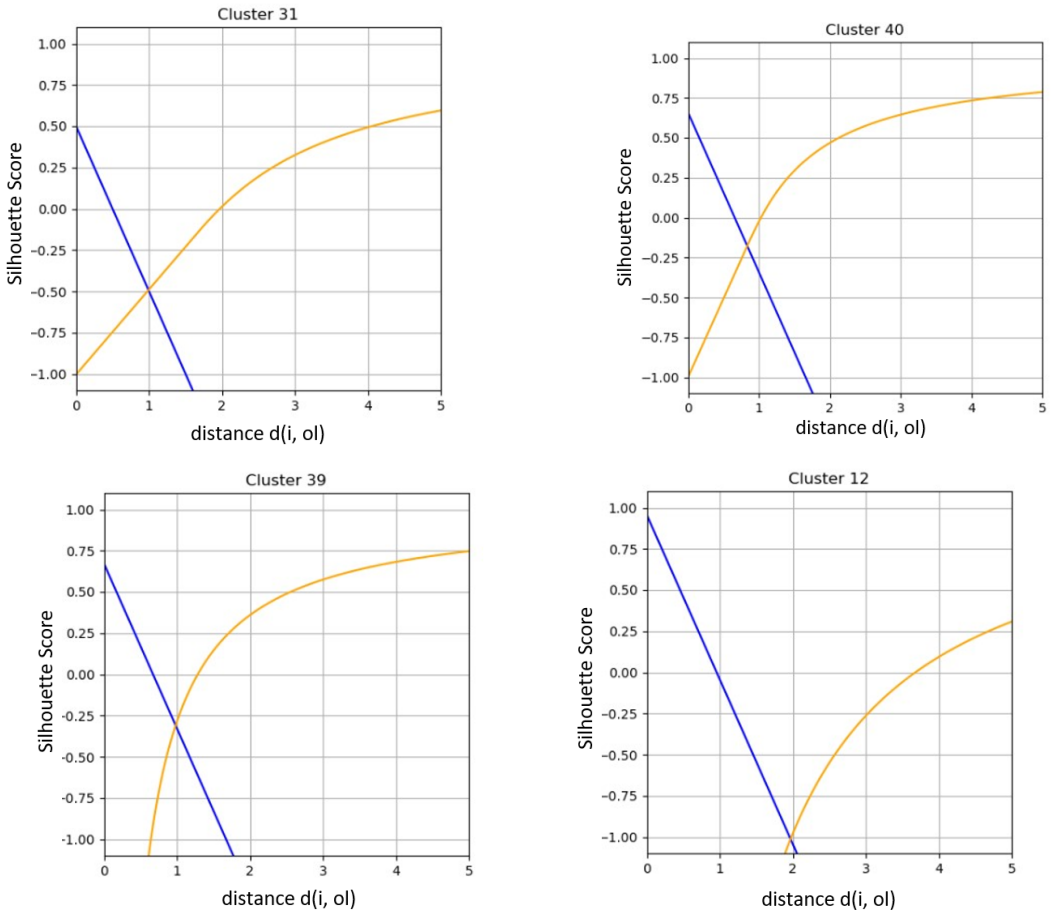

Figure 5.6: Determining the minimum distance (d-min) based on the cluster scores calculated from the functions f1 and f2 for the IBM dataset.

Data with ZD 6		**Data with ZD 7**	
Cluster-Id	Target-Id	Cluster-Id	Target-Id
31	0	12	4
40	1	31	1

Table 5.8: The correlation table for the evaluation data.

Zero-day 6	**Cluster-ID 31**			**Cluster-ID 40**		
	Before	Outliers Only	All data	Before	Outliers Only	All data
Count	184	3	187	1556	18	1574
Mean	1.45	153.67	3.89	0.75	42.03	1.22
Max	4.46	170.45	170.45	2.52	87.25	87.25
d-min	1	132,6	x	0.85	3.81	x
Score Sil	0.45	x	0.44	0.45	x	0.44
Online learning	YES			YES		

Table 5.9: Data distribution for zero-day 6.

5.4.6 Framework phase 5: distance analysis for detecting zero-day

As described in the architecture of the zero-day framework, the new data to be evaluated are injected into the unsupervised approach without any variation in the pa-

rameters of this model. Therefore, we do not modify the number of clusters or the hyper-parameters of the K-Means algorithm.

In this phase of the framework, we analyze the placement of the new data for evaluation, the clusters in which they are positioned, the distance measure of these data concerning the previous points in the target clusters, and most importantly, the variation in the silhouette score of each cluster with the addition of new points. This variation is crucial in deciding whether to retain these elements in the cluster or choose to split it.

Use case 1: Zero-day attack target ID 6

It involves DOS data, defined as an attack class never previously learned by our model. Therefore, technically, this class represents a zero-day for our model. The DOS data has been injected into the K-Means clustering, and all the points have been placed into two different clusters: C31 and C40.

As we observe in Table 5.9, before the introduction of zero-day data, the identified clusters (C31 and C40) had distinct characteristics, with notable differences in the number of elements, averages, and maximum distances. C31 contained 184 elements with an average of 1.45 and a maximum distance of 4.46, while C40 had 1556 elements with an average of 0.75 and a maximum distance of 2.52. The addition of DOS data significantly impacted the structure of the clusters. The number of elements increased for C31 (rising to 187) and C40 (reaching, 1574). Furthermore, the values of averages and maximum distances underwent significant changes. For C31, the average increased to 3.89 and the maximum distance to 170.45, whereas for C40, the average rose to 1.22 and the maximum distance to 87.25. The silhouette score, prior to the addition of zero-day data, for both clusters C31 and C40 decreased from 0.45 to 0.44.
These results suggest that the addition of zero-day data has altered the configuration of existing clusters, increasing the dissimilarity between the newly introduced data and the data already present.
The *Online learning* field, based on theoretical analysis of distance, and the variation in silhouette score, confirms the transition to the online learning stage when its value is set to YES. Table 5.10 showcases the results from online learning of various classification algorithms on the `input-datatset`, including the identified outliers within the clusters. In these findings, the label F1 score corresponds to a split on the zero-day data. Hence, predictions are made on zero-days that are either *outliers* and the threshold value for acceptable degradation of the base models, denoted as `deg-th`, has been set at 0.005.

Model	**DT**	**RF**	**KNN**	**NB**	**Our CNN**
Normal score	1.0	1.0	0.99	0.97	0.98
Online score	0.998	0.999	0.994	0.975	0.986
Label F1	1.0	1.0	1.0	1.0	1.0
Model to kept	YES	YES	YES	YES	YES

Table 5.10: Online learning for zero-day 6.

As observed in Table 5.10, all models remain below this threshold after online training, affirming their stability and justifying the retention of all models for zero-day attack detection. Therefore, the outliers represent zero-day flows, effectively

detected by our framework.

Use case 2: Zero-day attack target ID 7

For the second use case, we utilize the data from the `Critical Indicator of Compromise` class as never-before-learned data and inject it into the model.

Zero-day 7	Cluster-ID 12			Cluster-ID 39		
	Before	Outliers Only	All data	Before	Outliers Only	All data
Count	7	1	8	179	98	277
Mean	2.42	8.21	3.14	0.89	6.61	2.91
Max	2.79	8.21	8.21	6.15	7.23	7.23
d-min	1.96	6.79	x	0.99	6.37	x
Score Sil	0.44	x	0.42	0.44	x	0.42
Online learning	YES			YES		

Table 5.11: Data distribution for zero-day 7.

As observed in Table 5.11, prior to the introduction of zero-day data, the identified clusters (C31 and C40) displayed notable differences in the number of elements, means, and maximum distances. C31 contained 184 elements with a mean of 1.45 and a maximum distance of 4.46, while C40 comprised 1556 elements with a mean of 0.75 and a maximum distance of 2.52. The addition of D0S data significantly impacted the cluster structure because the number of elements increased for C31 (rising to 187) and C40 (reaching, 1574). Moreover, both the mean and maximum distances underwent substantial changes. For C31, the mean escalated to 3.89, and the maximum distance surged to 170.45, while for C40, the mean rose to 1.22 and the maximum distance reached 87.25.

Similar to the first use case, these findings suggest that the introduction of zero-day data altered the configuration of the existing clusters, increasing the dissimilarity between the newly introduced data and the pre-existing ones.

Model	**DT**	**RF**	**KNN**	**NB**	**Our CNN**
Normal score	1.0	1.0	0.97	0.972	0.99
Online score	0.998	0.999	0.993	0.974	0.982
Label F1	1.0	1.0	0.96	0.72	0.96
Model to kept	YES	YES	YES	YES	YES

Table 5.12: Online learning for zero-day 7.

In Table 5.12, we observe that the outliers do not lead to degradation in the base models. This observation underscores the models' strong ability to generalize well to new online data, particularly zero-day instances.

5.4.7 Exploring the NSL-KDD dataset and conducting comparative analysis

A functional approach that remains independent of specific datasets, hyper-parameters, and context serves as a reference model. Following the evaluation of our detection framework on an industrial dataset, we intend to leverage a public dataset to test its applicability. Our selection has been directed towards NSL-KDD dataset [166] presented in Section 4.4.2.

In this dataset comprising 41 features, we apply the detailed feature engineering outlined in Section 4.3.1 to retain only the features: `duration`, `protocol-type`, `service`, `land`, `src-bytes`, `dst-bytes`, `count`, `srv-count`, `same-srv rate`, `flag`, `srv-diff-host-rate`, and `wrong-fragment`.

The NSL-KDD dataset exhibits inconsistencies primarily in data variability, where characteristics between the training and test sets differ significantly, potentially impacting model generalization. Additionally, variations in attack definitions between training and test sets may lead to inconsistent evaluation results. Given these inconsistencies between the test and training data in the NSL-KDD dataset, we decide to solely focus on the training data. From this, we exclude a few subclasses of each attack family: neptune, satan, perl, spy, phf. Similar to the approach used in industrial datasets, we chose a subclass from the DOS family as a zero-day attack, specifically the teardrop with 892 entries. Additionally, 5000 random entries from the normal class are included. This constitutes the input dataset. Subsequently, data normalization is performed to proceed with the classification steps (using a 70% 30% train-test split for supervised classification) as outlined in our framework.

Model	DT			RF			BernoulliNB			KNN			Our CNN		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
macro avg	0.92	0.98	0.93	0.94	0.97	0.94	0.78	0.96	0.80	0.94	0.97	0.95	0.90	0.98	0.91
weighted avg	1.00	1.00	1.00	1.00	1.00	1.00	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99
Accuracy	100%			100%			98%			99%			99%		

Table 5.13: Classification with supervised algorithms.

As evidenced by Table 5.13, we present the overall results of various boosting algorithms and the base CNN algorithm. The results demonstrate remarkably high accuracy for all algorithms, with scores nearing or reaching 100 in the weighted measure. However, upon closer examination of the macro-avg and weighted-avg measures, there are some noteworthy points to highlight. The DT displays very good overall performance, with precision, recall, and an F1 score of 0.92, 0.98, and 0.93, respectively, for the macro-avg measure, showcasing its strong ability to correctly identify classes. RF shows slightly better performance than DT, with higher scores in precision (0.94), recall (0.97), and F1 score (0.94) for the macro-avg measure. NB exhibits slightly lower precision compared to DT and RF, with an F1 score of 0.80 for the macro-avg measure, indicating challenges in classifying certain classes. The KNN algorithm demonstrates robust performance with F1 scores of 0.95 for the macro-avg measure, showing a similar capability to DT but with a slight decrease in precision. These algorithmic results further solidify those of the CNN, which demonstrates high performance, with F1 scores of 0.99 for the macro-avg measure.

Once the `Target-set` for this dataset is established, we present in Table 5.14 only the variation in metrics of the cluster impacted by the addition of zero-day elements: C0. Our zero-day data corresponds to the teardrop subclass of the DOS family. The data from this subclass has been placed within cluster C0, with 94 outliers identified concerning the d-min distance of this cluster, as shown in the Figure 5.7.

In Table 5.14, we observe that the minimum outlier distance, d(i,ol), is significantly smaller than the theoretical `d-min` distance of cluster C0. Additionally, the inclusion of outliers within C0 notably degrades the silhouette score, decreasing from 0.651 to 0.428, considering the weight of 94 outliers.

Zero-day Teardrop	Cluster-ID 0		
	Before	Outliers Only	All data
Count	290	94	384
Mean	1.455	7.630	2.966
Max	5.764	9.116	9.116
d-min	1.09	6,98	x
Score Sil	0.651	x	0.428
Online learning	YES		

Table 5.14: Data distribution for zero-day attack Teardrop.

Model	**DT**	**RF**	**KNN**	**NB**	**Our CNN**
Normal score	1.0	1.0	0.99	0.98	0.99
Online score	0.997	0.955	0.74	0.989	0.966
Label F1	0.996	0.955	0.739	0.849	0.81
Model to kept	YES	YES	NO	YES	YES

Table 5.15: Online learning for zero-day attack Teardrop.

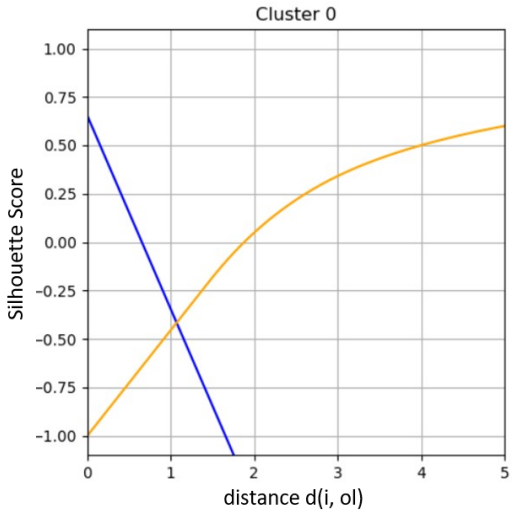

Figure 5.7: Determining the minimum distance (d-min) based on the cluster scores calculated from the functions f1 and f2 for the NSL-KDD dataset.

This degradation is further supported by the average distance within the new concatenated set of old elements and outliers, which increases from 1.455 to 2.966, more than doubling. This necessitates transitioning to the phase of online learning (Online learning = YES) to validate the detection of a new attack class that the model has not encountered before.

Approach	Evaluation Dataset	MITRE Tactic Covered	Learning Model	Model Perf.
Our framework	IBM + NSL-KDD	TA0001 - TA0002 - TA0003 - TA0008 - TA0010 - TA0011 - TA0040 - TA0042	Hybrid (Supervised + Unsupervised + Online learnning)	98.4% IBM and 96.6% NSL-KDD
Agathe et al. [11]	MAWI + UCSD	TA0011	Statistical measures	78.6% MAWI and 85.7% UCSD
Zhou et al. [12]	CIC-AWS-2018	TA0001 - TA0002 - TA0006 - TA0011 - TA0040	Supervised	$96-100\%$
Hindy et al. [13]	CICIDS2017 + NSL-KDD	TA0001 - TA0004 - TA0040 - TA0043	Unsupervised (Auto-Encoder)	90.01 – 99.67% CICIDS and 92.96% NSL-KDD
Pu et al. [14]	NSL-KDD	TA0001 - TA0004 - TA0040 - TA0043	Unsupervised	$52-99\%$
Comar et al. [15]	Private data from Internet service provider	TA0002	Hybrid (Supervised + Probabilistic Profiling)	$88,54\%$
Shamsul et al. [16]	Data from CA Technologies VET Zoo	TA0002	Hybrid (Semi-Supervised)	100%

Table 5.16: Comparing Approaches: Attack Coverage, Models, Accuracy.

Based on the results from Table 5.15, online learning highlights the detection of a zero-day, as no model degrades significantly compared to the defined threshold. However, the KNN model is not retained because the F1 score for the new zero-day class is very low, standing at 0.739.

Therefore, having completed all the steps of our framework, we can conclude that the teardrop class, previously defined as a zero-day, has been successfully detected as a new class of anomalies, never previously learned by our model.

In Table 5.16, we provide a comparative analysis between the performance of our framework and that of several state-of-the-art solutions for zero-day attack detection. We highlight key characteristics of the presented solutions, such as the utilized datasets, specific types of targeted zero-day attacks, detection algorithms employed, and the precision rates of the models.

5.5 Discussion and perspectives

In line with our ongoing work, several points of discussion can be explored to further enhance the detection and response to known and zero-day threats:

- **Online learning and continuous adaptation:** pose challenges regarding the ideal frequency of model updates to capture new threat patterns effectively without overwhelming the data. Implementing mechanisms for regular and real-time updates is crucial for maintaining the model's relevance.
- **Managing the increasing volume of data:** Addressing the continuous surge in data flow without compromising the model's accuracy requires considering mechanisms for data selection, processing, storage and optimizing the framework's deployment.
- **Anomaly detection:** Determining the threshold for segregating outliers with appropriate clusters and handling the potentially high number of newly generated zero-day classes demands thorough exploration.
- **Other case studies:** Exploring additional potential scenarios, such as the presence of outliers within the training data or validation data of a new zero-day class residing in distinct clusters, would be intriguing.
- **Updating the KB:** Incorporating new zero-day classes into the KB enhances detection but raises concerns about maintaining reliability amid continuous updates.

  Hence, all these discussions underscore the need for future research for improved detection of zero-day attacks in general and specifically for our framework for detecting these attacks.

5.6 Conclusion

In this chapter, we present a novel and effective approach for detecting both known attacks and zero-day exploits within network flows. By employing data collection techniques, data preprocessing, and supervised learning through 1D-CNN, we successfully identify these threats. The astute utilization of supervised algorithms such as DT, RF, KNN and NB, coupled with a boosting method, ensures an enhanced detection rate with a high detection accuracy of $98-100\%$. Furthermore, the incorporation of the unsupervised K-Means algorithm to extract flow patterns and generate a correlation table provides cross-validation between supervised and unsupervised methodologies. Our framework identifies anomalies by pinpointing outliers within clusters and correlation tables, thereby offering a dual validation mechanism for new data. Moreover, the validation of zero-day attacks using online learning demonstrates the capability of our solution to identify previously unlearned attack classes. This validation is substantiated by test cases on IBM and NSL-KDD datasets, where malicious activities such as DOS attacks and compromise indicators were identified, including the teardrop class within the NSL-KDD dataset. Looking ahead, we contemplate a more exhaustive exploration of diverse data sources to enhance detection and encompass a broader spectrum of attacks and botnets. In addition, the implementation of our proposition within an industrial environment holds promise for automating correlation rule creation, zero-day vulnerability detection, and real-time attack mitigation.

Chapter 6

Synthetic data generation

Contents

6.1 Introduction

The lack of diversity and realism in available datasets [132, 135, 138, 140], biases in data distribution, difficulty and cost of acquiring real-world data, and the risk of overfitting are significant scientific challenges that need to be addressed to improve the evaluation and effectiveness of intrusion detection systems. To address the limitations of existing datasets, synthetic data generation has emerged as a promising solution.
Data generation techniques using Generative Adversarial Networks (GAN) [144, 145, 146, 147] and Variational Auto-Encoders (VAE) [151, 152, 153] have been widely employed to produce synthetic and diverse data. However, scientific challenges remain, including the need for more comprehensive evaluation methods, consideration of all possible behaviors in network traffic flows, the complexity of the approach and modeling of complex dependencies between traffic features, as well as the use of real, non-simulated data as input.

In this chapter, we introduce a new approach to generate synthetic data using network activities profiling. We then delve into the anonymization process of the generated

data, ensuring confidentiality and protection of sensitive information. We present the validation techniques used to evaluate the quality of the generated data and compare it to the real original data. Finally, we illustrate the practical use of these synthetic data through data extraction cases and discuss the impacts and potential applications in various fields.

6.2 Our proposal

In Figure 6.1, we introduce our synthetic data generation model, designed based on real-world data without any implementation of an information system or simulation of attack scenarios. Our Network Information System Synthetic Data Generator `NIS-SDG` is composed of 5 major phases, is also applied in a real industrial context.

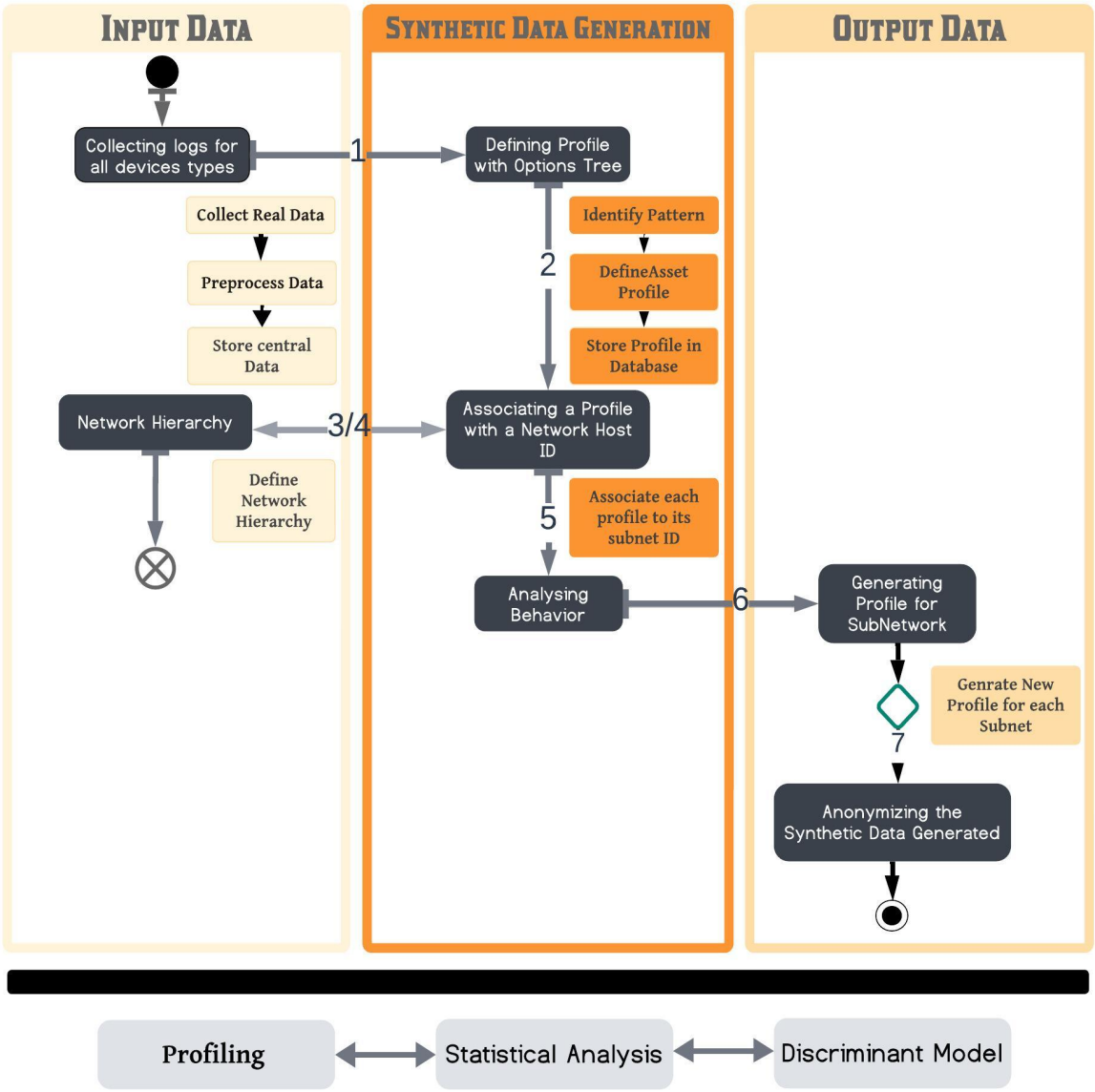

Figure 6.1: Our Model for Synthetic Data Generation.

6.2.1 Phase 1: Data collection and processing

During this phase, our primary objective is to comprehensively identify all assets and products within the information system, regardless of their source or vendor like operating system, network devices (routers), firewalls, applications, proxies, security sensors, etc. This full data source identification ensures optimal data diversity and accuracy in constructing asset profiles.

Following asset identification, we conduct a detailed analysis of data formats and structures for each type of log source in order to define the main event categories generated by each data source. We call these log families, the High-Level Categories (see Table 6.1). For each category, we then identify the necessary attributes, distinguishing between default attributes inherent to data extracts and custom attributes derived from specific device log investigations.

Category	**Description**
Authentication	Events related to authentication controls, group, or privilege change, for example, log in or log out.
Access	Events resulting from an attempt to access network resources, for example, firewall accept or deny.
System	Events related to system changes, software installation, or status messages.
Audit	Events related to audit activity.
Application	Events related to application activity.
Flow	Network Activities.

Table 6.1: Event Categories and Descriptions.

Then, we collect real data over an extensive timeframe, ideally covering various time slots (different days for instance) for all log source types that have been identified. Upon data extraction, data pre-processing is required. Firstly, we identify attributes containing only null values and remove them from the dataframe. Additionally, columns containing only one value are reviewed, with the recorded value noted before deletion. This approach provides efficient handling of null or unique values during result generation, as they do not contribute to the definition of profiles within the tree. Thus, we reduce the runtime of the algorithms.

6.2.2 Phase 2: Defining profiles with decision tree

In this phase, we employ a tree-based approach to define activity profiles covering the different behaviors present in the information system. Each profile encapsulates the specific end-to-end behavior of elements such as IP addresses, users, machines, programs, data sources, or files.

First, we establish a hierarchical data structure from the multidimensional dataset representing the logs of each data source collected in Phase 6.2.1. This structure relies on identifying the unique values of each attribute within the data source. Using these values, we construct a tree-like data structure where each level corresponds to an attribute of the data source, with the leaves representing unique combinations of attribute values.

Subsequently, we construct the tree recursively, starting with the most significant attribute, using the algorithm 1.

The recursive algorithm, `ConstructDict` takes a Dataframe D, an attribute identifier a, the number of remaining attributes n and an integer s representing the current stage as input. The algorithm returns a tree data structure T, where internal nodes represent unique attribute values and leaves contain counters for attribute value combinations.

With multiple log sources in an information system, we have chosen to construct profiles per log source. Thus, the identifier of a log source characterizes the first hierarchical level of all profiles. Once a log source is selected, the data categories within that log source form the second hierarchical level of profiles for that asset. By continuing this process, we can identify the entire hierarchy of the various data. At each level of the hierarchy, we create a node representing the corresponding attribute and its unique values. Subsequently, we generate sub-nodes for each unique value of the attribute, employing the same recursive steps for subsequent attributes.

The resulting data structure gives us a hierarchical representation of relationships between different attributes of the data source, facilitating data analysis and exploration. It also allows for the identification of the most frequent attribute value combinations, as well as aberrant or rare values. This approach offers an effective method for understanding the structure and characteristics of an element's log data and identifying potential trends and anomalies.

Algorithm 1 `ConstructDict`

procedure CONSTRUCTDICT(D, a, n, s)
  Retrieve the unique values U of attribute a in D.
  if $n = 1$ **then**
    Construct a dictionary T by associating each unique value $u \in U$ with a counter initialized to 1.
    return T.
  end if
  Initialize an empty dictionary T.
  for all unique values $u \in U$ **do**
    Recursively call CONSTRUCTDICT with arguments:
    $D' = \{x \in D \,|\, x[a] = u\}$: the subset of D where the attribute a equals u.
    $a' = A[s+1]$: the next attribute identifier in the hierarchy.
    $n' = n - 1$: the number of remaining attributes decremented by one.
    $s' = s + 1$: the updated stage integer.
    Associate the result with key u in T.
  end for
  return T.
end procedure

6.2.3 Phase 3: The definition of the network hierarchy associating a Profile with its Subnet ID

The network hierarchy (NH) refers to an organized structure of network resources and groups, facilitating in-depth analysis of network activity. In this phase, we associate each IP profile with a hierarchical network identifier to generate output profiles based on their original network segment. As a result, the generated profiles retain consistent behaviors and activities within the information system.

The `IdentityNetworkID` algorithm employs an iterative approach, comparing each IP address to predefined ranges until the most specific range is identified.

It enriches a Dataframe D by incorporating additional columns identifying network ID, start and end ranges for both source and destination IP addresses. We consider

that D contains network traffic records and the DataFrame C contains network ranges defined by CIDR notation. The `IdentityNetworkID` algorithm is suggested as follow:

Algorithm 2 `IdentityNetworkID`

procedure IDENTITYNETWORKID(D,C)
    Append columns for network IDs, start and end IP ranges.
    for each record r in D **do**
        Initialize maximum network mask and corresponding network ID, start and end IP range.
        for each record c in C **do**
            Identify the most specific network range containing the source and destination IP addresses in r.
            if network mask is greater than maximum network mask **then**
                Update maximum network mask and corresponding network ID, start and end IP range.
            end if
        end for
        Store the identified network ID, start and end IP range in the appropriate columns of D.
    end for
end procedure

6.2.4 Phase 4: Profile generation for sub-network

In this phase, we conduct an exhaustive analysis of the behavior of generated profiles, encompassing various attributes such as IP addresses for firewall data, usernames for system logs, file paths for process executions, etc., along with their associated values within each sub-network. This analysis involves the creation of a mapping that correlates relevant attribute values with corresponding subsets of data. Such an approach facilitates the grouping of data based on key attributes, thereby aiding in data analysis and exploration. By associating attribute values with data subsets, we can discern patterns and trends, facilitating informed decision-making and enhancing security system performance.

With D containing N attributes, we introduce the `CreateMapping` algorithm, which takes D and M as input, where M is an empty dictionary. Each key in M represents a specific combination of attribute values, and its corresponding value is a data subset associated with that key.

Algorithm 3 `CreateMapping`

procedure CREATEMAPPING(D, attributes)
  Count the unique combinations of attribute values from specified attributes in D.
  for all unique combination of attribute values (denoted as a key) **do**
    Filter D to obtain the corresponding data subset S.
    Associate the key with the data subset S in the dictionary M.
  end for
end procedure

We then proceed to a selection of real event and its associated asset profile. Utilizing the mapping table generated by the algorithm 3, we create a similar asset profile, thereby generating realistic synthetic data that mirrors observed behavior patterns, using the `GenerateEvents` algorithm which preserves attribute relationships as specified in the mapping dictionary. We consider D_{syn} the synthetic dataset with n random entries based on D and M.

Algorithm 4 `GenerateEvents`

procedure GENERATEEVENTS(n, M)
  for $i = 1$ to n **do**
    Randomly select a key k from the mapping dictionary M.
    Retrieve corresponding values v from M using the selected key.
    Generate a single random entry using function ONE_ENTRY(k, v).
    Append the generated entry to D_{syn}.
  end for
end procedure

Next, we compare the generated asset profile with the fundamental asset profile based on the original tree. This step determines the feasibility of the generated events based on observed behavior in real data.

A random IP profile within the specified range and other event-related attributes obtained from input values is then generated by a function.

6.2.5 Phase 5: Anonymizing the generated synthetic data

In this last phase, we deal with anonymization which is a crucial step in the processing and generation of sensitive data, as required by privacy and data confidentiality.

Indeed, some attributes may contain personally identifiable information (PII) such as IP addresses, MAC addresses, ports, dates, usernames, sent and received data, etc. These information must be handled with care to avoid any unauthorized disclosure or misuse.

According to Phase 6.2.1, we identify these attributes in the source logs and proceeded with their anonymization while preserving the structure and logic of the network. We preserve the usefulness of the data for analysis and research while ensuring confidentiality and privacy protection. To do this, we have developed an anonymization method for each attribute. In Table 6.2, we have compiled a list of attributes requiring anonymization based on our study of data sources, and propose some techniques to anonymize data.

Finally, for sent and received data, we applied a linear function to transform the real data. We also analyzed the value intervals for reception and transmission and generated new values in these intervals while staying close to the initial quantity. This approach preserves the usefulness of the data for analysis while masking the real data.

6.3 Assessment and validation of generated data

Evaluation and validation of synthetic data are essential steps to ensure that the generated data is reliable and useful for further analysis. Indeed, the generation of synthetic data aims to create artificial datasets that reproduce the characteristics and properties of real data, while preserving confidentiality and privacy of the individuals concerned. However, it is important to verify that the generated data is consistent and representative of the real data, and that it does not present any bias or errors that could affect the results of the analyses. In this section, we present three different methods of evaluation and validation of synthetic data that we have used: profiling, similarity measurement, and discriminant classification. These techniques allow us to measure the quality and reliability of the generated data and to guarantee its relevance for further analysis.

6.3.1 Profiling validation technique

Profile validation involves ensuring that each entry in the generated dataset corresponds to the possible values defined in the dictionary of real data, except for modified fields, which are now intervals (such as IP addresses, for example). This technique guarantees that all generated data complies with authorized values, ensuring the quality and reliability of the generated data while preserving confidentiality and privacy.

Let D be the original dataset, D_s be the synthetic dataset, and P be the dictionary of authorized values. Our objective is to verify that each entry in D_s corresponds to the possible values defined in P.

We define an algotihm `fcheck_dict` : $D_s \times P \rightarrow \{true, false\}$ that takes as input a synthetic dataset D_s and a dictionary P and returns a boolean value indicating whether all entries in D_s are present in P. The algorithm `fcheck_dict` is defined as follows:

For each entry e in D_s:

Types	**Attributes**	**Risk**	Techniques To anonymize
OSI Model Layer 3	IP Source v4, IP Destination v4,	IP address leakage, enabling tracking of network communications.	Divide them into four octets and apply a mapping that assigns an 8-bit value to each octet while avoiding transforming a public IP address into a private IP address and vice versa (based on RFC 1918 and CIDR).
OSI Model Layer 2	Source MAC, Destination MAC	MAC address leakage, enabling device identification.	Apply a map that associates each MAC address with another randomly generated address.
Path	File Path, Parent Process Path, Process Path, Share Path	File system and process structure leakage, enabling identification of used applications and files.	Maintain all default hierarchical paths across Windows, Linux, etc., and replace any other identifying elements within a path with alphanumeric generic names.
File & process name	Filename, Creator Process Name, New Process Name, Parent Process Name, Process Name	Disclosure of information about used applications and manipulated files.	Maintain machine process names as they are, and utilize random file names.
Username & Domain	Username, Initiator Username, Target Username, Initiator User Name, Domain, Account Domain, Group Domain, Target Computer Domain, Target User Domain, User Domain	User identifier and domain leakage, enabling identification of users and systems.	For unique usernames and domains, employ a 15-character alphanumeric substitution. For email-style usernames, generate distinct 15 character strings for each part before and after the "@" symbol
Asset Identifier	Log Source Identifier, Machine ID, Machine Identifier, Destination Asset Name, Source Asset Name	Infrastructure leakage, enabling identification of assets and data sources.	Generate unique identifiers per device
Timestamp	Storage Date, Log Source Date, Start Date	Event timeline leakage, enabling activity correlation.	Increment by 24 hours to preserve the information on working and non-working hours.
Mails	Sender, Receiver, Object	Email content leakage, enabling disclosure of private or sensitive information.	Generate distinct 15 character strings for each part before and after the ”@” symbol.

Table 6.2: Data to anonymize.

Algorithm 5 `fcheck_dict`

procedure FCHECK_DICT(e, P)
  for each element x in e **do**
    if the type of x is not a string **then**
      continue to the next element.
    end if
    if x is not in P **then**
      return false.
    end if
    if x is in P **then**
      Update P to be the sub-dictionary associated with the key x, i.e., $P \leftarrow P[x]$.
    end if
  end for
end procedure

If all entries in D_s are present in P, return true.

The algorithm `fcheck_dict` ensures that all entries in the synthetic dataset D_s comply with the authorized values defined in the dictionary P. By employing this function, we can verify the quality and reliability of the generated synthetic data while preserving the confidentiality and privacy of the individuals concerned.

6.3.2 Statistical analysis validation technique

To evaluate the quality of data generation, we utilized a similarity measure between the distributions of the generated data and the real data. We chose to use the Euclidean distance as the similarity metric, a commonly used measure in statistics and machine learning. This metric allows us to compare the similarity between two distributions effectively.

To calculate the Euclidean distance between the two distributions, we first identified the unique values in each dataset. Then, we computed the frequency distribution of each unique value in both the real data and the generated data. Next, we calculated the sum of the squares of the differences between the frequencies of each unique value in the two distributions. Finally, we obtained the Euclidean distance by taking the square root of this sum.

It is important to note that a distance of 0 would indicate that the generated data is identical to the real data, which is not our objective. Our goal is to enrich the data and explore the space of possibilities to improve the understanding of the network architecture for the model.

Let U be the set of unique values in the original dataset, and V be the set of unique values in the synthetic dataset. Let D be the frequency distribution of the unique values in the original dataset, and S be the frequency distribution of the unique values in the synthetic dataset.

We define an algorithm $\texttt{fsimilarity} : D \times S \to \mathbb{R}$ that takes as input the frequency distributions D and S and returns a similarity score between them. The algorithm `fsimilarity` is defined as follows:

Algorithm 6 `fsimilarity`

procedure FSIMILARITY(U, V, D, S)
Initialize a similarity score, `score` $\leftarrow 0$.
for each unique value u in U **do**
if u is also in V **then**
Calculate the squared difference between the frequencies of u in the original and synthetic datasets, i.e., $(D[u] - S[u])^2$.
Add the squared difference to the score.
end if
end for
Calculate the square root of the score.
end procedure

The algorithm `fsimilarity` returns a non-negative real number representing the similarity score between the original and synthetic datasets' distributions. A lower score indicates a higher similarity between the distributions, while a higher score indicates a lower similarity. By employing this function, we can quantitatively assess the quality of the generated synthetic data and compare it to the original dataset.

In summary, we used the Euclidean distance to measure the similarity between the distributions of the real and generated data. This similarity measure allows us to evaluate the quality of data generation and ensure that the generated data is sufficiently different from the real data to explore the space of possibilities while being sufficiently similar to be useful for our model.

6.3.3 Discriminant model validation technique

In this final validation technique, a discriminant machine learning model is employed to distinguish between real data and synthetically generated data. Specifically, a decision tree is trained on a labeled dataset comprising both real and synthetic data to predict whether a given observation is real or synthetic.

To evaluate the performance of the model, the dataset is divided into a training set and a test set. The model is trained on the training set using the decision tree classification algorithm. Subsequently, the model's performance is assessed on the test set by calculating the precision score, which measures the proportion of correctly classified observations by the model.

Let D be the original dataset and S be the synthetic dataset. We define an algorithm `fdiscriminant` : $D \times S \rightarrow \mathbb{R}$ that takes as input the original dataset D and the synthetic dataset S and returns a performance score of a discriminative machine learning model. The algorithm `fdiscriminant` is defined as follows:

Algorithm 7 `fdiscriminant`

procedure FDISCRIMINANT(D, S)
Concatenate the original dataset D and the synthetic dataset S into a single dataset T.
Label the entries in T with a binary target variable y where:
Entries from D are assigned the label 1 (indicating real data),
Entries from S are assigned the label 0 (indicating synthetic data).
Preprocess the dataset T by encoding categorical variables using a label encoder.
Split the preprocessed dataset T into training set T_{train} and testing set T_{test}.
Train a discriminative machine learning model using the training set T_{train}.
Evaluate the performance of the trained model on the testing set T_{test} using appropriate performance metrics.
end procedure

The algorithm `fdiscriminant` returns a performance score indicating the ability of the discriminative model to distinguish between real and synthetic data. A lower score implies that the synthetic data is more similar to the original data, making it harder for the model to differentiate between them. This suggests that the synthetic data preserves the privacy of the original data while maintaining its utility.

This validation technique allows us to evaluate the quality of the synthetically generated data by comparing their distribution to that of the real data using a discriminant machine learning model. If the model can distinguish real data from synthetic data with high precision, this indicates that the synthetic data is sufficiently different from the real data to be used in applications such as generating training data for deep learning models. Conversely, if the model fails to distinguish real data from synthetic data, this suggests that the synthetic data is too similar to the real data and may not be useful for improving model performance.

6.4 Data extraction cases

6.4.1 Case 1: Firewall events

For the firewall data generation, we extracted 450k entries from this information system, ensuring comprehensive coverage across various timeframes to capture different types of activities on this equipment. Subsequently, we executed our approach to generate 2000 entries with the same profiles identified in the original data. Figure 6.2 provides an example of the hierarchy of a profile linked to a specific high-level category. Table 6.3 showcases five generated firewall data entries linked to the *High Level Category: Access*. We note that the IP addresses (OSI model layer 3) are contained in the last two fields, which represent identifiers for each subnet. As a reminder, each identifier is associated with a dedicated network address range in accordance with the Network Hierarchy.

1. For result validation, all generated entries are within the dictionary of possible values and profiles
2. In Figures 6.3 and 6.4, we note that the generated data closely matches the synthetic data in terms of event types and destination ports, with similarity scores of 0.0292 for destination ports and 0.01426605 for event ID. These scores indicate that the generated data is sufficiently similar to the real data to be useful, while being different enough to explore the space of possibilities

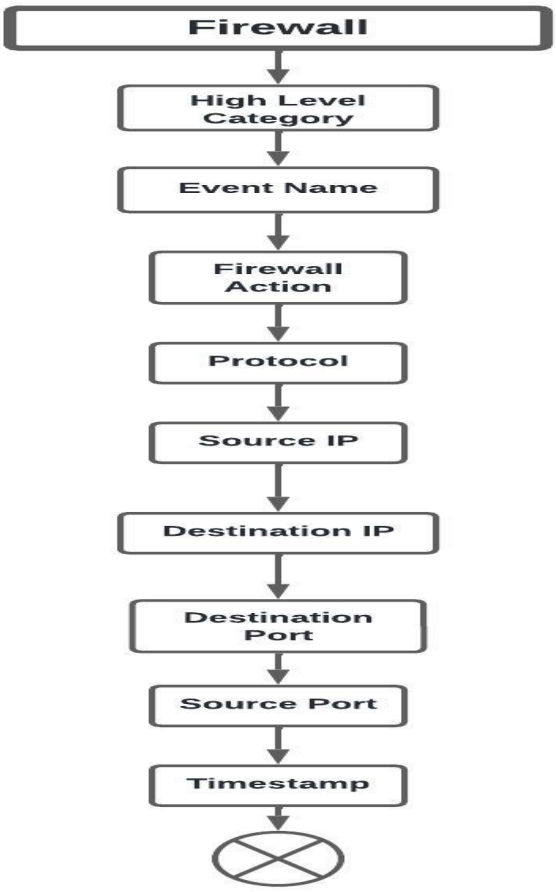

Figure 6.2: Tree Structure of the firewall profiles.

Dest. Port	Event Category	Event ID	Log Source Time	Protocol	Source Port	Action	Event Name	id_src	id_dest
389.0	VPN-1 & FireWall-1	Accept	Jan 30, 2024, 2:00:00 PM	tcp_ip	60991.0	Accept	Firewall Permit	436	369
53.0	VPN-1 & FireWall-1	Decrypt	Jan 30, 2024, 2:02:43 PM	udp_ip	54122.0	Decrypt	Decrypt	430	369
135.0	VPN-1 & FireWall-1	Accept	Jan 30, 2024, 1:59:29 PM	tcp_ip	57054.0	Accept	Firewall Permit	369	443
443.0	VPN-1 & FireWall-1	Accept	Jan 30, 2024, 2:01:05 PM	tcp_ip	57519.0	Accept	Firewall Permit	438	397
443.0	VPN-1 & FireWall-1	Accept	Jan 30, 2024, 2:00:52 PM	tcp_ip	58082.0	Accept	Firewall Permit	305	335

Table 6.3: Example 2 of firewall logs.

3. The DT model was trained on a labeled dataset containing real (class 1) and synthetic (class 0) data. The results of the confusion matrix showed a precision of 0.45 for synthetic data and 0.36 for real data, indicating that 45% of synthetic data were correctly classified as synthetic and 36% of real data as real, with an overall accuracy of 0.42. However, the model's performance is moderate due to low precision for both classes and low recall for class 1, indicating that many real data points were misclassified as synthetic. This justifies the need for statistical analysis in terms of similarity and difference.

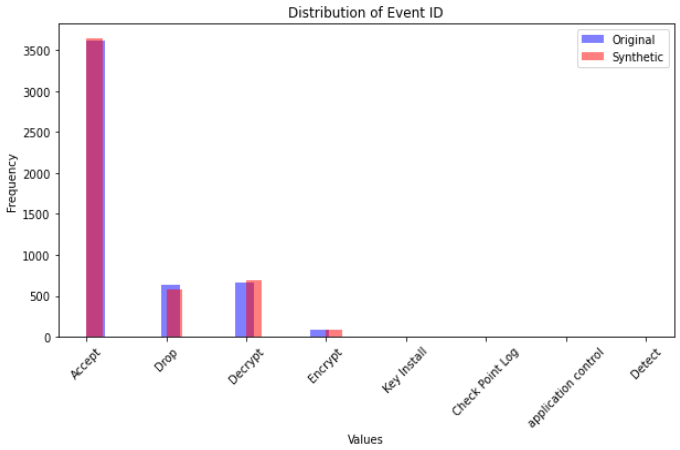

Figure 6.3: Distribution of Event ID.

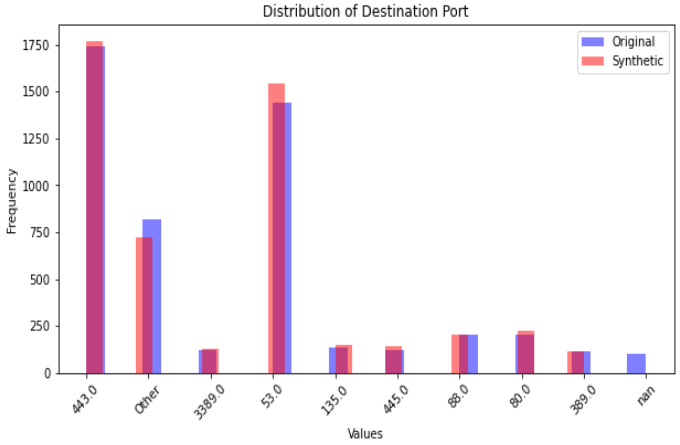

Figure 6.4: Distribution of Destination Port.

Class	Precision	Recall	F1-score	Support
0	0.45	0.64	0.53	1007
1	0.36	0.20	0.26	993
Accuracy			0.42	2000
Macro avg	0.40	0.42	0.39	2000
Weighted avg	0.40	0.42	0.39	2000

Table 6.4: Classification report.

6.4.2 Case 2: Microsoft Security events

Faced with the diversity of Microsoft Security Event logs, our second log extraction case using the `NIS-SDG` data generator will focus on user account authentication attempts. This encompasses the High-level category of Authentication and Microsoft events IDs such as: 4624, 4672, 4648 and 4625. We anticipate generating 4000 events corresponding to these event IDs. In Table 6.5, we present an example of authentication data, highlighting some relevant attributes.
Figure 6.5 depicts the hierarchy of various profiles that can be generated from Microsoft Security Event logs. Initially, we identified attributes common to all high-level categories to define the first hierarchical levels of the tree. Subsequently, addi-

tional attributes are progressively added based on the high-level category.

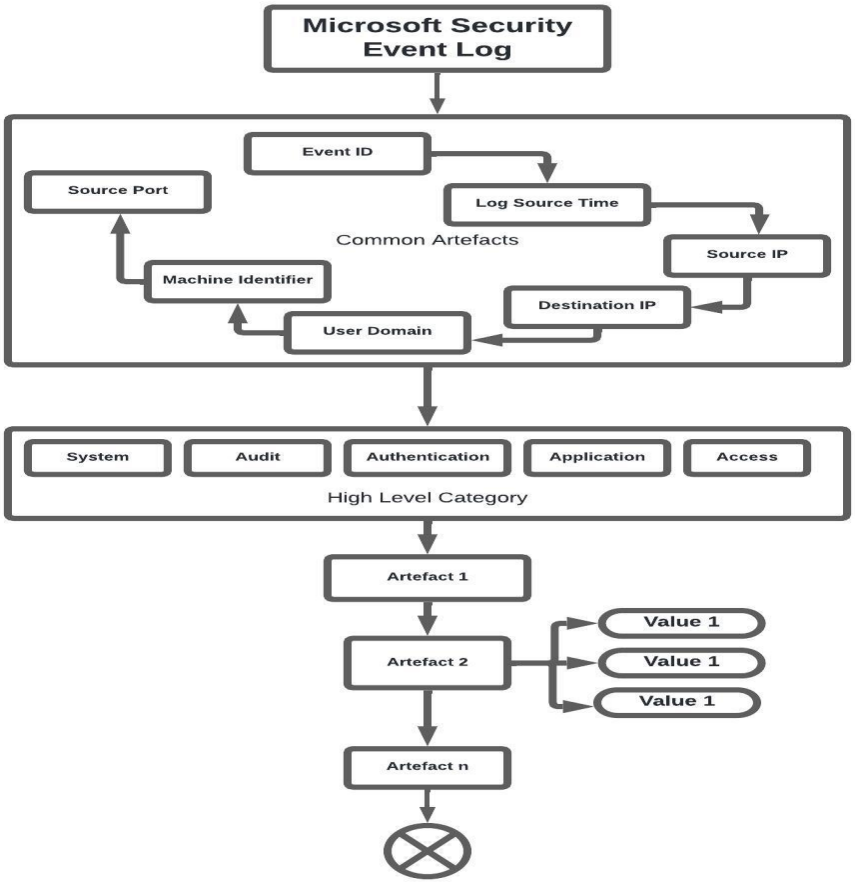

Figure 6.5: Tree Structure of the Microsoft Security Events.

1. The `fcheck_dict` algorithm returns the value `true`, validating that all generated entries are within the dictionary of possible values and profiles, similar to the firewall events case.
2. The validation algorithm `fsimilarity` returns low scores for all compared distributions, particularly for Event-ID and Process-Name, with scores of 0.0186 and 0.0192 respectively. This confirms the high similarity and coherence of the generated Microsoft authentication data, as shown in Figure 6.7.
3. The model's moderate score underscores the similarity between generated and real data, with an overall classification accuracy and recall of 0.61. Specifically, the F1 score is 0.68 for real data (class 0) and 0.52 for syntehtic data (class 1). These scores further confirm the challenge the decision tree (DT) faces in distinguishing between the two classes, with a better balance of precision and recall for real data compared to the generated ones.

Logon Type	Package	User Initiator	Machine ID	Process Name	ID-Src	User Target	Event ID	Log Source Time
3.0	NTLM	VTIHnGS80p	Host900	-	457	fF7uqIstCM	4624	2024-04-23 14:26:26
3.0	Kerberos	VTIHnGS80p	Host2	-	457	PXSYTmjIBt	4624	2024-04-23 13:53:02
3.0	NTLM	VTIHnGS80p	Host5	-	457	9grK454sTK	4624	2024-04-23 12:13:40
3.0	Negotiate	vZYc3v2WCo	Host1300	CMS.exe	454	FbawQpPVUZ	4624	2024-04-23 11:48:09
NaN	NaN	grzT73UtAF	Host115	NaN	454	TNJ5XYmb1Y	4672	2024-04-23 14:02:28
3.0	NTLM	VTIHnGS80p	Host93	-	457	CTy2qui1Bi	4624	2024-04-23 12:32:47
3.0	NTLM	VTIHnGS80p	Host831	-	457	BI3u3yN3Rx	4624	2024-04-23 14:12:28
NaN	NaN	grzT73UtAF	Host111	NaN	454	TNJ5XYmb1Y	4672	2024-04-23 12:36:15
NaN	NaN	grzT73UtAF	Host785	NaN	444	TNJ5XYmb1Y	4672	2024-04-23 12:14:06
NaN	NaN	grzT73UtAF	Host27	NaN	443	TNJ5XYmb1Y	4672	2024-04-23 12:54:24

Table 6.5: Example of Microsoft Security event logs.

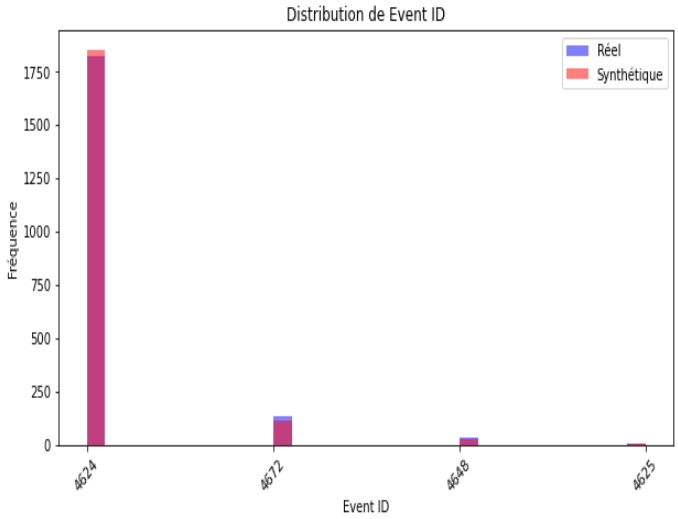

Figure 6.6: Distribution of Event ID.

Class	Precision	Recall	F1-score	Support
0	0.58	0.81	0.68	1990
1	0.69	0.41	0.52	1990
Accuracy			0.61	3980
Macro Avg	0.63	0.61	0.60	3980
Weighted Avg	0.63	0.61	0.60	3980

Table 6.6: Microsoft Authentication Events.

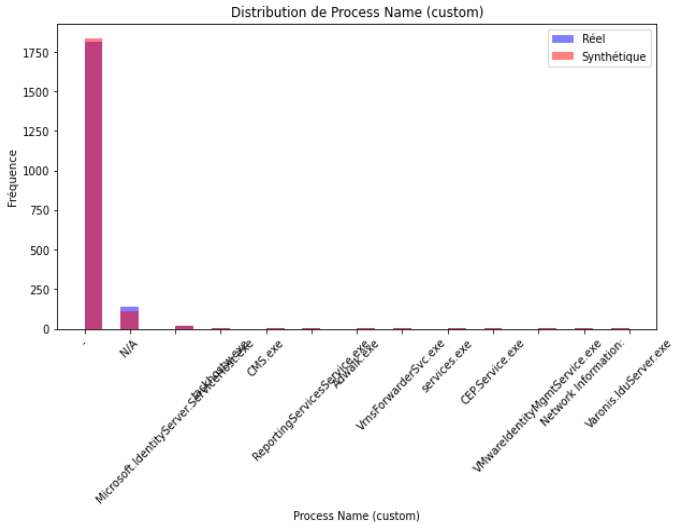

Figure 6.7: Distribution of Event ID and Event Process Name.

6.5 Conclusion and perspectives

The synthetic data generator proposed in this chapter aims to address a crucial need in the cybersecurity scientific community: access to reliable, diverse, and representative datasets of the real world. Indeed, existing datasets often suffer from bias, limitations in terms of diversity, and lack of representativeness of real environments. Moreover, data confidentiality issues are a major obstacle to their sharing and use. Our proposal is based on a rigorous and transparent methodology, relying on proven technologies and recognized practices in data science. The collection and preparation of source data, anonymization of sensitive information, generation of synthetic data that respects the structure and logic of the network, and rigorous validation of data quality are the key steps of our approach.

The results obtained so far are promising. The tests and evaluations conducted demonstrate the generator's ability to produce synthetic data statistically similar to real data, while ensuring scenario diversity, robustness to changes, and efficient anomaly management. Moreover, sensitive data anonymization is ensured, thus respecting confidentiality requirements.

The synthetic data generator developed has a high potential impact on research and applications in cybersecurity. It will enable the feeding of machine learning models with labeled and diversified data, contributing to the improvement of the accuracy and performance of anomaly, intrusion, and cyberattack detection solutions. In terms of future perspectives, we plan to deploy the application's source code, which will allow the scientific community and companies to collaborate and enrich knowledge in the field of cybersecurity. The use of this generator will enable the exploration of new avenues and expand the scope of synthetic data applications.

At the same time, we are considering the development of a user interface that will facilitate the generator's use by a wide audience, mainly targeted at researchers, security professionals, and students, or integrating it directly into Watsonx.data.

Finally, extending the methodology to other types of security data, such as application logs or incident reports, will enable the generator's impact to be extended to

other areas of cybersecurity. Following this methodology, the development of synthetic data for a specific data type can be relatively time-consuming, especially when numerous validation criteria are desired.

Chapter 7

Conclusion, impacts and future work

Contents

7.1 Conclusion

The collection and analysis of communication flows is a reliable method for detecting cyberattacks within an information system. These flows, collected via dedicated probes, may be encrypted. This large volume of data collected and centralized through a network flow collector is used by detection and response teams to manually create and update correlation rules. These rules ensure the identification of abnormal and deviant behaviors within an information system. However, this approach often generates high rates of false positives and its implementation is costly in terms of time and human resources. Moreover, these teams face difficulties in proactively detecting unknown attacks. Thus, SOC and response teams require dynamic approaches to effectively detect cyberattacks of various types and characteristics in order to respond to them promptly.

Various works in the literature address the issue of attack detection and optimization. Traditional approaches for improving the detection strategy, based on signature detection, payload detection, and anomaly detection, generate many false positives and are limited by obfuscation techniques. Furthermore, machine learning and deep learning algorithms are increasingly being used. However, existing solutions are specific to each case in feature extraction, focus on optimizing hyperparameters and model comparisons do not have the same basis.

In addition to the approaches for detecting known attacks, zero-day attack detection solutions are proposed in the literature. The state-of-the-art in attack detection often focuses on enhancing accuracy for specific attack types, overlooking the potential for multiple attack scenarios.

Moreover, to evaluate the effectiveness of the proposed cyberattack detection methods, it is essential to generate up-to-date synthetic and anonymized data that is based on a solid foundation of recent and non-simulated attack scenarios. Indeed, the datasets and synthetic data generation techniques used are limited, outdated in terms of scenarios, and based on simulated attack scenarios.

In this thesis, we propose novel approaches to optimize and strengthen security network incident detection and synthetic data generation, with a particular focus on incidents related to network traffic in the industrial context of corporate networks. The three major contributions presented in this thesis offer significant advances in the fields of detecting attacks related to network flows, detecting zero-day attacks, and generating realistic and up-to-date synthetic data for effective model evaluation.

Considering the static approach of creating correlation rules via network flow collectors for cyberattack detection, we leveraged machine learning and deep learning algorithms to automate and improve the detection of attacks related to network flows. Thus, we designed an approach based on convolutional neural networks (CNN) that aims to identify a universal feature engineering, applicable to any network traffic analysis context. Therefore, we defined three families of network characteristics based on the definition of a network flow and the different steps of traffic exchange between assets. These families include information about communicating assets, data on the session established between these assets, and information about the speed and data exchange rate. Moreover, the CNN's feature detector ability to extract deeper patterns from the data has allowed the extraction of relevant information from the initial dataset and thus improved the classification of network flows. Therefore, our two-step solution thus guarantees better classification of network flows without feature engineering specific to a dataset or particular context. Our approach has been evaluated on a dataset from our IBM work context and two public datasets (NSL-KDD and UNSW-NB15).

After that, we addressed the issue of proactive detection of unknown attacks. In the face of the constant increase in the number and surfaces of attacks, unknown vulnerabilities are increasingly exploited in information systems. We have therefore focused our efforts on this issue, as existing approaches generally target specific attack classes to detect or optimize model precision rates. Our contribution was to propose a zero-day attack detection framework that covers the entire chain of data collection, processing, analysis, and classification up to the detection of an anomaly in the classification. Our framework validates these unknown attacks using online learning to confirm the new attack class. Our solution has also been validated on different datasets, including an extract from our work context and the public dataset, NSL-KDD. Our results confirm the detection of a zero-day attack class for different use cases studied.

Finally, to address the lack of realistic and up-to-date data necessary for effective model evaluation, we addressed the issue of synthetic data generation. After a thorough study of the different types of data sources in an IS, we identified the essential attributes to retain from a specific data payload and those containing sensitive and critical information to anonymize. Then, by extracting a large volume of data covering all assets of a given information system and all necessary time windows, we used decision trees to generate the different profiles available by data source, which we stored in a database. These profiles were then analyzed to determine behaviors, and the network architecture was also examined. Each profile was associated with

a specific network identifier, which allowed us to generate data based on these profiles associated with a network identifier. A thorough verification was carried out on the generated data using three mechanisms: profiling, statistical analysis, and discriminant method. Finally, we proceeded with the anonymization of the critical fields identified at the beginning to avoid disclosing information about the original information system.

7.2 Impacts on the IBM ecosystem

7.2.1 Detection of known attacks

For this approach, we plan to seamless integration of classification models, notably CNN, into the user interface of the SIEM through a dedicated AI tool. Therefore, SOC analysts, correlation engineers, and detection and response specialists can easily select models and adjust hyper-parameters. By utilizing labeled datasets, experts can train these models based on existing correlation rules and associated data flows. Once trained, these models are evaluated in real-time on the collected flows within the SIEM.

The classification results, distinguishing between normal and malicious flows, are then integrated into the SIEM interface via this dedicated application or AI tool. Detection and response experts can intervene to optimize existing rules by analyzing false positives and negatives. They adjust detection thresholds and incorporate specific knowledge that may have caused these false detection (administrative tasks, internal IP addresses originating from internal audits, scheduled vulnerability scans). This feedback loop ensures the tool's constant adaptation to new threats, thereby enhancing the organization's security posture. By closely integrating supervised classification and correlation rule creation, our approach enables the rapid identification and response to known and emerging threats.

7.2.2 Detection of zero-day attacks

Continuing the automation of detection, and with a tool to assist security analysts and incident response experts, the zero-day detection protocol can provide significant value in an incident detection and response tool.

The integration of the zero-day protocol into a SIEM or dedicated AI tool provides a comprehensive solution for proactive detection of zero-day threats and dynamic creation of new correlation rules. Leveraging advanced anomaly detection techniques, the protocol enables analysts to swiftly identify suspicious behaviors, thereby reducing response time to potential attacks. Its continuous adaptation to emerging threats ensures that correlation rules remain relevant and up-to-date, offering improved visibility into network activities and faster threat detection. By facilitating event correlation and optimizing detection accuracy through a combination of different techniques, the protocol empowers analysts to create more precise and reliable correlation rules, thereby reducing false positives and negatives. Validated on real-world data, it ensures the reliability of results, enabling analysts to make confident decisions to strengthen network security.

7.2.3 Data generation

Our data generator addresses a real need in the scientific community by providing a generator of synthetic data based on carefully diversified real data.

Our generator enables the provision of datasets that constitute stable sets for learning and verifying different models in machine learning. Moreover, through collaboration, it is possible to enrich our knowledge base with known, recent attack scenarios and incorporate them into the datasets provided by the generator. The methodology we have adopted also guarantees easier data sharing through integrated anonymization. Thus, we can provide datasets that meet the expectations of researchers in the field of machine learning applied to cybersecurity and the many applications that result from it.

By integrating our data generation mechanism into Watsonx.data, we offer a recognized, reliable, and shared platform for the entire scientific and industrial community to collect, store, clean, and transform data on a large scale. This integration enables users to generate high-quality synthetic data for training and validating AI models based on real and diverse data. Our integrated anonymization methodology also ensures the confidentiality and security of the data used to generate the synthetic data, in addition to the basic security mechanisms provided by the Watsonx.data solution.

Furthermore, this integration would ensure better evaluation of the AI models developed, trained, and deployed via Watsonx.ai, thereby facilitating the simulation of the security use cases proposed by this solution, such as attack detection, anomaly detection in an IS, improvement of the performance of AI models, etc.

7.3 Perspectives

In this section, we present perspectives for continuing the initiated work and explore additional research avenues.

Firstly, we have outlined several future research directions in zero-day exploit detection. We will aim to refine our detection approach by dynamically updating the model with previously identified zero-day attacks. This involves employing advanced techniques to analyze and integrate new information, thereby improving the model's accuracy and efficacy.

Moreover, once a zero-day exploit has been confirmed, we plan to extract the patterns and behaviors are extracted to create new detection rules. Thus, maintaining a dynamic, accurate model while ensuring swift detection poses several scientific challenges. One major challenge is developing efficient data structures for incremental updates and prompt rule access. Another challenge is ensuring that the model adapts to new data and unknown attacks without discarding prior knowledge. To address this, we will deploy meta-learning and transfer learning algorithms to enhance the system's adaptability.

Therefore, to facilitate timely detection of new threats and keep defenses current, we propose to develop a module that autonomously generates detection rules based on the characteristics of detected zero-day attacks. This involves automatically translating attack patterns into actionable detection rules, which is another significant

scientific challenge. To address this challenge, we will explore Natural Language Processing (NLP) methods for log analysis and rule generation once a new detection scenario is confirmed.

Another promising research direction would be to aim for, we the detection of complex cyber threats by developing attack detection models for multi-stage attacks. Using Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) techniques.

A given attack scenario can be simple or complex. Simple attacks are characterized by a single execution phase, whereas complex attacks are a sequence of actions carried out over time, commonly referred to as the cyber kill chain. Leveraging the cyber kill chain framework, we will model attack stages to identify patterns and anomalies indicative of multi-stage attacks. LSTM enables the detection of long-term dependencies between attack stages, enhancing the accuracy and efficiency of our detection framework and offering better protection against advanced cyber threats.

However, enhancing detection using RNN with LSTM poses significant challenges. Modeling complex attack sequences is challenging due to intricate temporal dependencies in multi-stage attacks. Advanced LSTM architectures like bidirectional LSTM and attention mechanisms are necessary to capture these nuances. Real-time detection and scalability are additional challenges. They require LSTM model optimization for low-latency processing through techniques like model pruning and integration with high-performance computing frameworks. Additionally, reducing false positives and negatives is crucial, achievable through ensemble learning methods and hybrid models.

Finally, considering the insider threats while building attack detection mechanisms is also an interesting research direction. Indeed, unlike external attackers, insiders have legitimate access to the organization's systems and data, enabling them to bypass traditional security measures. These attackers can learn operational models within the information system and manipulate them to evade detection, presenting significant challenges for maintaining robust security.

Challenges related to these insider attackers include detection evasion, bias and manipulation of models, real-time monitoring and response, behavioral analysis and context awareness, as well as integration with existing security frameworks. Thus, to address these challenges, research avenues include leveraging adversarial learning to identify and mitigate attempts to bias models, using behavioral and contextual analyses to establish baselines of normal user behavior (profiling). Indeed, adversarial learning to enhance the resilience of security models and the implementation of continuous learning systems capable of adapting to new patterns of insider threats as they emerge. We aim to enhance the detection and response capabilities against insider threats, thereby ensuring more robust and adaptive security measures.

Bibliography

[1] "Ibm security report: Data breach cost 2023." https://www.ibm.com/fr-fr/reports/data-breach.

[2] C. Zhang, G. Wang, S. Wang, D. Zhan, and M. Yin, "Cross-domain network attack detection enabled by heterogeneous transfer learning," *Computer Networks*, vol. 227, p. 109692, 2023.

[3] W. Tounsi and H. Rais, "A survey on technical threat intelligence in the age of sophisticated cyber attacks," *Computers Security*, vol. 72, pp. 212–233, 2018.

[4] Blessing Guembe, Ambrose Azeta, Sanjay Misra, Victor Chukwudi Osamor, Luis Fernandez-Sanz, and Vera Pospelova, "The emerging threat of ai-driven cyber attacks: A review," *Applied Artificial Intelligence*, vol. 36, no. 1, p. 2037254, 2022.

[5] P. Lin, K. Ye, Y. Hu, Y. Lin, and C.-Z. Xu, "A Novel Multimodal Deep Learning Framework for Encrypted Traffic Classification," *IEEE/ACM Transactions on Networking*, pp. 1–16, 2022.

[6] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "A review on machine learning-based approaches for Internet traffic classification," *Annals of Telecommunications*, vol. 75, no. 11, pp. 673–710, 2020.

[7] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[8] C. Zhang, F. Ruan, L. Yin, X. Chen, L. Zhai, and F. Liu, "A deep learning approach for network intrusion detection based on nsl-kdd dataset," in *2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, pp. 41–45, 2019.

[9] Z. Tauscher, Y. Jiang, K. Zhang, J. Wang, and H. Song, "Learning to detect: A data-driven approach for network intrusion detection," in *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pp. 1–6, 2021.

[10] L. Liu, P. Wang, J. Lin, and L. Liu, "Intrusion detection of imbalanced network traffic based on machine learning and deep learning," *IEEE Access*, vol. 9, pp. 7550–7563, 2021.

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

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

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

[14] G. Pu, L. Wang, J. Shen, and F. Dong, "A hybrid unsupervised clustering-based anomaly detection method," *Tsinghua Science and Technology*, vol. 26, no. 2, pp. 146–153, 2021.

[15] P. M. Comar, L. Liu, S. Saha, P.-N. Tan, and A. Nucci, "Combining supervised and unsupervised learning for zero-day malware detection," in *2013 Proceedings IEEE INFOCOM*, pp. 2022–2030, 2013.

[16] S. Huda, S. Miah, M. Mehedi Hassan, R. Islam, J. Yearwood, M. Alrubaian, and A. Almogren, "Defending unknown attacks on cyber-physical systems by semi-supervised approach and available unlabeled data," *Information Sciences*, vol. 379, pp. 211–228, 2017.

[17] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network flows," 1988.

[18] B. Li, J. Springer, G. Bebis, and M. Hadi Gunes, "A survey of network flow applications," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 567–581, 2013.

[19] B. Claise, "Cisco systems netflow services export version 9," tech. rep., 2004.

[20] P. Aitken, B. Claise, and B. Trammell, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information." RFC 7011, Sept. 2013.

[21] S. Panchen, N. McKee, and P. Phaal, "InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks." RFC 3176, Sept. 2001.

[22] Í. Cunha, F. Silveira, R. Oliveira, R. Teixeira, and C. Diot, "Uncovering artifacts of flow measurement tools," in *Passive and Active Network Measurement: 10th International Conference, PAM 2009, Seoul, Korea, April 1-3, 2009. Proceedings 10*, pp. 187–196, Springer, 2009.

[23] D. Zhou, Z. Yan, Y. Fu, and Z. Yao, "A survey on network data collection," *Journal of Network and Computer Applications*, vol. 116, pp. 9–23, 2018.

[24] C. So-In, "A survey of network traffic monitoring and analysis tools," *Cse 576m computer system analysis project, Washington University in St. Louis*, 2009.

[25] P. Sahoo, R. Chottray, and S. Pattnaiak, "Research issues on windows event log," *International Journal of Computer Applications*, vol. 41, no. 19, 2012.

[26] B. S. Nayak, "A practical logging solution using syslog,"

[27] C. Lonvick, "The bsd syslog protocol," tech. rep., 2001.

[28] D. New and M. Rose, "Reliable delivery for syslog," tech. rep., 2001.

[29] D. Sulistyowati, F. Handayani, and Y. Suryanto, "Comparative analysis and design of cybersecurity maturity assessment methodology using nist csf, cobit, iso/iec 27002 and pci dss," *JOIV: International Journal on Informatics Visualization*, vol. 4, no. 4, pp. 225–230, 2020.

[30] "MITRE ATTA&CK."

[31] P. Rajesh, M. Alam, M. Tahernezhadi, A. Monika, and G. Chanakya, "Analysis Of Cyber Threat Detection And Emulation Using MITRE Attack Framework," in *International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, pp. 4–12, 2022.

[32] M. Lezzi, M. Lazoi, and A. Corallo, "Cybersecurity for industry 4.0 in the current literature: A reference framework," *Computers in Industry*, vol. 103, pp. 97–110, 2018.

[33] S. Gupta, B. S. Chaudhari, and B. Chakrabarty, "Vulnerable network analysis using war driving and security intelligence," in *2016 International Conference on Inventive Computation Technologies (ICICT)*, vol. 3, pp. 1–5, 2016.

[34] "Ibm product: Security qradar ndr." https://www.ibm.com/products/qradar-ndr.

[35] "Ibm product: Security qradar edr." https://www.ibm.com//products/qradar-edr.

[36] "Ibm product: Security qradar soar." https://www.ibm.com/products/qradar-soar.

[37] "Ibm plans to make llama 2 available within its watsonx ai and data platform, pr newswire us, 9 august. 2023."

[38] "Ibm product: Watsonx[.]ai."

[39] "Ibm product: Watsonx[.]data."

[40] F. Hou, Z. Wang, Y. Tang, and Z. Liu, "Protecting integrity and confidentiality for data communication," in *Proceedings. ISCC 2004. Ninth International Symposium on Computers And Communications (IEEE Cat. No. 04TH8769)*, vol. 1, pp. 357–362, IEEE, 2004.

[41] S. Duggineni, "Data integrity and risk," *Open Journal of Optimization*, vol. 12, no. 2, pp. 25–33, 2023.

[42] O. A. Fayayola, O. L. Olorunfemi, and P. O. Shoetan, "Data privacy and security in it: A review of techniques and challenges," *Computer Science & IT Research Journal*, vol. 5, no. 3, pp. 606–615, 2024.

[43] J. B. Bernabe and A. Skarmeta, "Introducing the challenges in cybersecurity and privacy: The european research landscape," in *Challenges in Cybersecurity and Privacy-the European Research Landscape*, pp. 1–21, River Publishers, 2022.

[44] M. Toussaint, S. Krima, and H. Panetto, "Industry 4.0 data security: a cybersecurity frameworks review," *Journal of Industrial Information Integration*, p. 100604, 2024.

[45] K. Nova, "Security and resilience in sustainable smart cities through cyber threat intelligence," *International Journal of Information and Cybersecurity*, vol. 6, no. 1, pp. 21–42, 2022.

[46] "Ibm report - a focus on europe: X-force threat intelligence index 2024."

[47] "Ibm report - a focus on europe: X-force threat intelligence index 2024."

[48] F. Ghaffari, K. Gilani, E. Bertin, and N. Crespi, "Identity and access management using distributed ledger technology: A survey," *International Journal of Network Management*, vol. 32, no. 2, p. e2180, 2022.

[49] I. Indu, P. R. Anand, and V. Bhaskar, "Identity and access management in cloud environment: Mechanisms and challenges," *Engineering Science and Technology, an International Journal*, vol. 21, no. 4, pp. 574–588, 2018.

[50] N. Dissanayake, A. Jayatilaka, M. Zahedi, and M. A. Babar, "Software security patch management-a systematic literature review of challenges, approaches, tools and practices," *Information and Software Technology*, vol. 144, p. 106771, 2022.

[51] V. Markkanen and T. Frantti, "Patch management planning - towards one-to-one policy," in *2023 10th International Conference on Dependable Systems and Their Applications (DSA)*, pp. 60–69, 2023.

[52] A. A. Ganin, P. Quach, M. Panwar, Z. A. Collier, J. M. Keisler, D. Marchese, and I. Linkov, "Multicriteria decision framework for cybersecurity risk assessment and management," *Risk Analysis*, vol. 40, no. 1, pp. 183–199, 2020.

[53] A. Marotta and S. Madnick, "Perspectives on the relationship between compliance and cybersecurity.," *Journal of Information System Security*, vol. 16, no. 3, 2020.

[54] M. Mirtsch, "The role of conformity assessment in the digital transformation: Focusing on cybersecurity," in *EURAS Proceedings 2018-Standards for a Smarter Future*, vol. 2018, pp. 183–202, Verlagshaus Mainz GmbH Aachen, 2018.

[55] K. Stine, S. Quinn, G. Witte, and R. Gardner, "Integrating cybersecurity and enterprise risk management (erm)," *National Institute of Standards and Technology*, vol. 10, 2020.

[56] A. Marotta and S. Madnick, "Convergence and divergence of regulatory compliance and cybersecurity.," *Issues in Information Systems*, vol. 22, no. 1, 2021.

[57] A. Ambre and N. Shekokar, "Insider threat detection using log analysis and event correlation," *Procedia Computer Science*, vol. 45, pp. 436–445, 2015.

[58] N. Khan, I. Yaqoob, I. A. T. Hashem, Z. Inayat, W. K. Mahmoud Ali, M. Alam, M. Shiraz, A. Gani, *et al.*, "Big data: survey, technologies, opportunities, and challenges," *The scientific world journal*, vol. 2014, 2014.

[59] S. Bhatt, P. K. Manadhata, and L. Zomlot, "The operational role of security information and event management systems," *IEEE security & Privacy*, vol. 12, no. 5, pp. 35–41, 2014.

[60] M. Cinque, D. Cotroneo, and A. Pecchia, "Challenges and directions in security information and event management (siem)," in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pp. 95–99, IEEE, 2018.

[61] O. Podzins and A. Romanovs, "Why siem is irreplaceable in a secure it environment?," in *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*, pp. 1–5, IEEE, 2019.

[62] K. Maharana, S. Mondal, and B. Nemade, "A review: Data pre-processing and data augmentation techniques," *Global Transitions Proceedings*, vol. 3, no. 1, pp. 91–99, 2022. International Conference on Intelligent Engineering Approach(ICIEA-2022).

[63] S. Gupta and A. Gupta, "Dealing with noise problem in machine learning data-sets: A systematic review," *Procedia Computer Science*, vol. 161, pp. 466–474, 2019. The Fifth Information Systems International Conference, 23-24 July 2019, Surabaya, Indonesia.

[64] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on big data: Opportunities and challenges," *Neurocomputing*, vol. 237, pp. 350–361, 2017.

[65] M. Conti, T. Dargahi, and A. Dehghantanha, *Cyber threat intelligence: challenges and opportunities*. Springer, 2018.

[66] A. A. Salih and M. B. Abdulrazzaq, "Cyber security: performance analysis and challenges for cyber attacks detection," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 31, no. 3, pp. 1763–1775, 2023.

[67] A. A. Mughal, "The art of cybersecurity: Defense in depth strategy for robust protection," *International Journal of Intelligent Automation and Computing*, vol. 1, no. 1, pp. 1–20, 2018.

[68] S. E. Whang and J.-G. Lee, "Data collection and quality challenges for deep learning," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3429–3432, 2020.

[69] S. E. Whang, Y. Roh, H. Song, and J.-G. Lee, "Data collection and quality challenges in deep learning: A data-centric ai perspective," *The VLDB Journal*, vol. 32, no. 4, pp. 791–813, 2023.

[70] J. McHugh, "Intrusion and intrusion detection," *International Journal of Information Security*, vol. 1, pp. 14–35, 2001.

[71] Y.-H. Choi, M.-Y. Jung, and S.-W. Seo, "L+ 1-mwm: A fast pattern matching algorithm for high-speed packet filtering," in *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*, pp. 2288–2296, IEEE, 2008.

[72] D. E. Knuth, J. H. Morris, Jr, and V. R. Pratt, "Fast pattern matching in strings," *SIAM journal on computing*, vol. 6, no. 2, pp. 323–350, 1977.

[73] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," *Communications of the ACM*, vol. 20, no. 10, pp. 762–772, 1977.

[74] S. Wu, U. Manber, *et al.*, *A fast algorithm for multi-pattern searching*. University of Arizona. Department of Computer Science Tucson, AZ, 1994.

[75] A. V. Aho and M. J. Corasick, "Efficient string matching: an aid to bibliographic search," *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, 1975.

[76] M. Aldwairi and K. Al-Khamaiseh, "Exhaust: Optimizing wu-manber pattern matching for intrusion detection using bloom filters," in *2015 2nd World Symposium on Web Applications and Networking (WSWAN)*, pp. 1–6, IEEE, 2015.

[77] M. Aldwairi, K. Al-Khamaiseh, F. Alharbi, and B. Shah, "Bloom filters optimized wu-manber for intrusion detection," *Journal of Digital Forensics, Security and Law*, vol. 11, no. 4, p. 5, 2016.

[78] I. Obeidat and M. AlZubi, "Developing a faster pattern matching algorithms for intrusion detection system," *International Journal of Computing*, vol. 18, no. 3, pp. 278–284, 2019.

[79] A. Hnaif, K. M. Jaber, M. A. Alia, and M. Daghbosheh, "Parallel scalable approximate matching algorithm for network intrusion detection systems.," *Int. Arab J. Inf. Technol.*, vol. 18, no. 1, pp. 77–84, 2021.

[80] E. Papadogiannaki, G. Tsirantonakis, and S. Ioannidis, "Network intrusion detection in encrypted traffic," in *2022 IEEE Conference on Dependable and Secure Computing (DSC)*, pp. 1–8, IEEE, 2022.

[81] B. Bayoğlu and İbrahim Soğukpınar, "Graph based signature classes for detecting polymorphic worms via content analysis," *Computer Networks*, vol. 56, no. 2, pp. 832–844, 2012.

[82] A. Tongaonkar, R. Torres, M. Iliofotou, R. Keralapura, and A. Nucci, "Towards self adaptive network traffic classification," *Computer Communications*, vol. 56, pp. 35–46, 2015.

[83] A. Zand, G. Vigna, X. Yan, and C. Kruegel, "Extracting probable command and control signatures for detecting botnets," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, (New York, NY, USA), p. 1657–1662, Association for Computing Machinery, 2014.

[84] A. S. Dina and D. Manivannan, "Intrusion detection based on machine learning techniques in computer networks," *Internet of Things*, vol. 16, p. 100462, 2021.

[85] R. Samrin and D. Vasumathi, "Review on anomaly based network intrusion detection system," in *2017 international conference on electrical, electronics, communication, computer, and optimization techniques (ICEECCOT)*, pp. 141–147, IEEE, 2017.

[86] Z. Liu, N. Japkowicz, R. Wang, Y. Cai, D. Tang, and X. Cai, "A statistical pattern based feature extraction method on system call traces for anomaly detection," *Information and Software Technology*, vol. 126, p. 106348, 2020.

[87] M. Wurzenberger, G. Höld, M. Landauer, and F. Skopik, "Analysis of statistical properties of variables in log data for advanced anomaly detection in cyber security," *Computers Security*, vol. 137, p. 103631, 2024.

[88] I. Friedberg, F. Skopik, G. Settanni, and R. Fiedler, "Combating advanced persistent threats: From network event correlation to incident detection," *Computers & Security*, vol. 48, pp. 35–57, 2015.

[89] S. B. Shamsuddin and M. E. Woodward, "Modeling protocol based packet header anomaly detector for network and host intrusion detection systems," in *Cryptology and Network Security* (F. Bao, S. Ling, T. Okamoto, H. Wang, and C. Xing, eds.), (Berlin, Heidelberg), pp. 209–227, Springer Berlin Heidelberg, 2007.

[90] P. Satam and S. Hariri, "Wids: An anomaly based intrusion detection system for wi-fi (ieee 802.11) protocol," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 1077–1091, 2021.

[91] T. J. Parvat and P. Chandra, "Performance improvement of deep packet inspection for intrusion detection," in *2014 IEEE Global Conference on Wireless Computing Networking (GCWCN)*, pp. 224–228, 2014.

[92] C. Togay, A. Kasif, C. Catal, and B. Tekinerdogan, "A firewall policy anomaly detection framework for reliable network security," *IEEE Transactions on Reliability*, vol. 71, no. 1, pp. 339–347, 2022.

[93] J. Kuri, G. Navarro, L. Mé, and L. Heye, "A pattern matching based filter for audit reduction and fast detection of potential intrusions," in *Recent Advances in Intrusion Detection: Third International Workshop, RAID 2000 Toulouse, France, October 2–4, 2000 Proceedings 3*, pp. 17–27, Springer, 2000.

[94] Y. Zhou and G. Ding, "Research of multi-pattern matching algorithm based on characteristic value," in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pp. 1657–1661, 2017.

[95] J. Wu, W. Wang, L. Huang, and F. Zhang, "Intrusion detection technique based on flow aggregation and latent semantic analysis," *Applied Soft Computing*, vol. 127, p. 109375, 2022.

[96] M. A. Jabbar and S. Samreen, "intelligent network intrusion detection using alternating decision trees," in *International Conference on Circuits, Controls, Communications and Computing (I4C)*.

[97] B. S. Sharmila and R. Nagapadma, "Intrusion Detection System using Naive Bayes algorithm," in *IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, 2019.

[98] L. Koc and A. D. Carswell, "Network Intrusion Detection Using a HNB Binary Classifier," in *17th UKSim-AMSS International Conference on Modelling and Simulation (UKSim)*, pp. 81–85, 2015.

[99] P. Owezarski, "Investigating adversarial attacks against random forest-based network attack detection systems," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–6, 2023.

[100] H. M. Prachi and P. Sharma, "Intrusion detection using machine learning and feature selection," *International Journal of Computer Network and Information security*, vol. 11, no. 4, pp. 43–52, 2019.

[101] H. Sanusi, Z. Utic, and J. Kim, "Improving network intrusion detection using supervised learning for feature selection," in *2023 IEEE/ACIS 8th International Conference on Big Data, Cloud Computing, and Data Science (BCD)*, pp. 42–48, 2023.

[102] N. H. Kumar and R. Dhanalakshmi, "A novel host based intrusion detection system using supervised learning by comparing svm over random forest," in *2023 Eighth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*, pp. 1–4, 2023.

[103] M. Macas, C. Wu, and W. Fuertes, "A survey on deep learning for cybersecurity: Progress, challenges, and opportunities," *Computer Networks*, vol. 212, p. 109032, 2022.

[104] K. A. Taher, B. Mohammed Yasin Jisan, and M. M. Rahman, "Network intrusion detection using supervised machine learning technique with feature selection," in *2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST)*, pp. 643–646, 2019.

[105] H. Zhao, Y. Feng, H. Koide, and K. Sakurai, "An ANN Based Sequential Detection Method for Balancing Performance Indicators of IDS," in *7th International Symposium on Computing and Networking (CANDAR)*, pp. 239–244, 2019.

[106] S. Sivamohan, S. Sridhar, and S. Krishnaveni, "An Effective Recurrent Neural Network (RNN) based Intrusion Detection via Bi-directional Long

Short-Term Memory," in *International Conference on Intelligent Technologies (CONIT)*, 2021.

[107] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve Intrusion Detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

[108] M. Azizjon, A. Jumabek, and W. Kim, "1D CNN based network intrusion detection with normalization on imbalanced data," in *International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pp. 218–224, 2020.

[109] S. Amutha, K. R, S. R, and K. M, "Secure network intrusion detection system using nid-rnn based deep learning," in *2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, pp. 1–5, 2022.

[110] G. Kotani and Y. Sekiya, "Unsupervised scanning behavior detection based on distribution of network traffic features using robust autoencoders," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 35–38, 2018.

[111] S. Alam, Y. Alam, S. Cui, and C. M. Akujuobi, "Unsupervised network intrusion detection using convolutional neural networks," in *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0712–0717, 2023.

[112] J. V. Anand Sukumar, I. Pranav, M. Neetish, and J. Narayanan, "Network intrusion detection using improved genetic k-means algorithm," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 2441–2446, 2018.

[113] H. M. Tahir, A. M. Said, N. H. Osman, N. H. Zakaria, P. N. A. M. Sabri, and N. Katuk, "Oving k-means clustering using discretization technique in network intrusion detection system," in *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*, pp. 248–252, 2016.

[114] M. Kherbache, D. Espes, and K. Amroun, "An enhanced approach of the k-means clustering for anomaly-based intrusion detection systems," in *2021 International Conference on Computing, Computational Modelling and Applications (ICCMA)*, pp. 78–83, 2021.

[115] R. Younisse and Q. A. Al-Haija, "An empirical study on utilizing online k-means clustering for intrusion detection purposes," in *2023 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pp. 1–5, 2023.

[116] R. Khaoula and M. Mohamed, "Improving intrusion detection using pca and k-means clustering algorithm," in *2022 9th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 1–5, 2022.

[117] S. Saheel, A. Alvi, A. R. Ani, T. Ahmed, and M. F. Uddin, "Semi-supervised, neural network based approaches to face mask and anomaly detection in surveillance networks," *Journal of Network and Computer Applications*, vol. 222, p. 103786, 2024.

[118] S. Li, Y. Cao, S. Liu, Y. Lai, Y. Zhu, and N. Ahmad, "Hda-ids: A hybrid dos attacks intrusion detection system for iot by using semi-supervised cl-gan," *Expert Systems with Applications*, vol. 238, p. 122198, 2024.

[119] X. Zheng, S. Yang, and X. Wang, "Sf-ids: An imbalanced semi-supervised learning framework for fine-grained intrusion detection," in *ICC 2023-IEEE International Conference on Communications*, pp. 2988–2993, IEEE, 2023.

[120] X. Wang and X. Qiu, "A novel semi-supervised anomaly detection method for network intrusion detection," in *2022 IEEE 22nd International Conference on Communication Technology (ICCT)*, pp. 1276–1280, 2022.

[121] "Ibm security topic: What is a zero-day exploit?."

[122] M. Mohammadi, B. Raahemi, A. Akbari, and B. Nassersharif, "Class dependent feature transformation for intrusion detection systems," in *2011 19th Iranian Conference on Electrical Engineering*, pp. 1–1, 2011.

[123] A. Aleroud and G. Karabatis, "Toward zero-day attack identification using linear data transformation techniques," in *2013 IEEE 7th International Conference on Software Security and Reliability*, pp. 159–168, 2013.

[124] D. Jin, J. Xie, S. Chen, J. Yang, X. Liu, and W. Wang, "Zero-day traffic identification using one-dimension convolutional neural networks and auto encoder machine," in *2020 IFIP Networking Conference (Networking)*, pp. 559–563, 2020.

[125] K. Roshan and A. Zafar, "An optimized auto-encoder based approach for detecting zero-day cyber-attacks in computer network," in *2021 5th International Conference on Information Systems and Computer Networks (ISCON)*, pp. 1–6, 2021.

[126] B. Kızıltaş and E. Gül, "Network anomaly detection with convolutional neural network based auto encoders," in *2020 28th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2020.

[127] R. Hosler, A. Sundar, X. Zou, F. Li, and T. Gao, "Unsupervised deep learning for an image based network intrusion detection system," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, pp. 6825–6831, 2023.

[128] G. Pu, L. Wang, J. Shen, and F. Dong, "A hybrid unsupervised clustering-based anomaly detection method," *Tsinghua Science and Technology*, vol. 26, no. 2, pp. 146–153, 2020.

[129] I. Mbona and J. H. P. Eloff, "Detecting zero-day intrusion attacks using semi-supervised machine learning approaches," *IEEE Access*, vol. 10, pp. 69822–69838, 2022.

[130] A. Arun, A. S. Nair, and A. G. Sreedevi, "Zero day attack detection and simulation through deep learning techniques," in *2024 14th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pp. 852–857, 2024.

[131] M. Soltani, B. Ousat, M. Jafari Siavoshani, and A. H. Jahangir, "An adaptable deep learning-based intrusion detection system to zero-day attacks," *Journal of Information Security and Applications*, vol. 76, p. 103516, 2023.

[132] K. K. R. Kendall, *A database of computer attacks for the evaluation of intrusion detection systems*. PhD thesis, Massachusetts Institute of Technology, 1999.

[133] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection: Results from the jam project.-1999," *http://kdd. ics. uci. edu*.

[134] J. Song, H. Takakura, and Y. Okabe, "Description of kyoto university benchmark data," *Available at link: http://www. takakura. com/Kyoto_data/BenchmarkData-Description-v5. pdf [Accessed on 15 March 2016]*, 2006.

[135] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, 2009.

[136] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers Security*, vol. 31, no. 3, pp. 357–374, 2012.

[137] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers Security*, vol. 45, pp. 100–123, 2014.

[138] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, pp. 1–6, 2015.

[139] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 184–208, 2015.

[140] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," *ICISSp*, vol. 1, pp. 108–116, 2018.

[141] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[142] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Creation of flow-based data sets for intrusion detection," *Journal of Information Warfare*, vol. 16, no. 4, pp. 41–54, 2017.

[143] M. Ring, A. Dallmann, D. Landes, and A. Hotho, "Ip2vec: Learning similarities between ip addresses," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 657–666, IEEE, 2017.

[144] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Flow-based benchmark data sets for intrusion detection," in *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS)*, pp. 361–369, ACPI South Oxfordshire, UK, 2017.

[145] P. Zingo and A. Novocin, "Can gan-generated network traffic be used to train traffic anomaly classifiers?," in *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 0540–0545, 2020.

[146] A. Cheng, "Pac-gan: Packet generation of network traffic using generative adversarial networks," in *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 0728–0734, IEEE, 2019.

[147] B. Dowoo, Y. Jung, and C. Choi, "Pcapgan: Packet capture file generator by style-based generative adversarial networks," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 1149–1154, IEEE, 2019.

[148] D. Lu, J. Fei, L. Liu, and Z. Li, "A gan-based method for generating sql injection attack samples," in *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, vol. 10, pp. 1827–1833, 2022.

[149] S. A.-S. Tertsegha J. Anande and M. S. Leeson, "Generative adversarial networks for network traffic feature generation," *International Journal of Computers and Applications*, vol. 45, no. 4, pp. 297–305, 2023.

[150] S. Suh, D. H. Chae, H.-G. Kang, and S. Choi, "Echo-state conditional variational autoencoder for anomaly detection," in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 1015–1022, 2016.

[151] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Variational data generative model for intrusion detection," *Knowledge and Information Systems*, vol. 60, pp. 569–590, 2019.

[152] F. Meslet-Millet, S. Mouysset, and E. Chaput, "Necstgen: An approach for realistic network traffic generation using deep learning," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pp. 3108–3113, 2022.

[153] P.-J. Chuang and P.-Y. Huang, "B-vae: a new dataset balancing approach using batched variational autoencoders to enhance network intrusion detection," *The Journal of Supercomputing*, vol. 79, no. 12, pp. 13262–13286, 2023.

[154] H. Naseri and V. Mehrdad, "Novel cnn with investigation on accuracy by modifying stride, padding, kernel size and filter numbers," *Multimedia Tools and Applications*, vol. 82, no. 15, pp. 23673–23691, 2023.

[155] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[156] M. Maithem and G. A. Al-Sultany, "Network intrusion detection system using deep neural networks," in *Journal of Physics: Conference Series*, vol. 1804, p. 012138, IOP Publishing, 2021.

[157] W.-F. Zheng, "Intrusion Detection Based on Convolutional Neural Network," in *International Conference on Computer Engineering and Application (ICCEA)*, pp. 273–277, 2020.

[158] S. S. Sekharan and K. Kandasamy, "Profiling SIEM tools and correlation engines for security analytics," in *International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 717–721, 2017.

[159] B. Shah and B. H. Trivedi, "Reducing features of kdd cup 1999 dataset for anomaly detection using back propagation neural network," in *2015 Fifth International Conference on Advanced Computing Communication Technologies*, pp. 247–251, 2015.

[160] "Ibm security topic: Qradar architecture and deployment."

[161] S. Jeong, J.-H. You, and J. W.-K. Hong, "Design and implementation of virtual tap for sdn-based openstack networking," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 233–241, 2019.

[162] W. Zhenqi and W. Xinyu, "Netflow based intrusion detection system," in *2008 International Conference on MultiMedia and Information Technology*, pp. 825–828, 2008.

[163] S. U. Rehman, W.-C. Song, and M. Kang, "Network-wide traffic visibility in of@tein sdn testbed using sflow," in *The 16th Asia-Pacific Network Operations and Management Symposium*, pp. 1–6, 2014.

[164] N. Chen and R. E. Johnson, "Jflow: Practical refactorings for flow-based parallelism," in *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 202–212, 2013.

[165] A. Shahraki, M. Abbasi, A. Taherkordi, and A. D. Jurcut, "A comparative study on online machine learning techniques for network traffic streams analysis," *Computer Networks*, vol. 207, p. 108836, 2022.

[166] N. Solekha, "Analysis of nsl-kdd dataset for classification of attacks based on intrusion detection system using binary logistics and multinomial logistics," *Seminar Nasional Official Statistics*, vol. 2022, pp. 507–520, Nov. 2022.