A Novel Labeling Method for Intrusion Detection and Evaluating ML Model Generalizability

Johan Andersson
Viktor Rysund

Computer Science
Two-year Master's Thesis
30 ECTS
Spring 2024
Supervisor: Reza Koshkangini

Popular Science Summary

Modern organizations use sophisticated systems to detect cyber threats, which can be understood as malicious attempts to steal or damage data. This research explores how we can improve these systems using machine learning, a type of artificial intelligence that learns from examples.

Imagine you have cameras in your house that learn over time to recognize the faces of family and friends, and also to notice unusual activities or unknown faces. Similarly, Intrusion Detection Systems (IDS) learn from patterns of network behavior to recognize and react to cyber threats. These systems typically work in two ways: some compare network activities to a known list of threats (like a camera recognizing a known face), while others look for unusual patterns that might indicate a break-in attempt (like a movement sensor noticing someone lurking around at night).

Keeping these systems effective is a challenge. They need to be trained with data that accurately represents real network traffic, including the latest types of cyberattack. To address this problem, this thesis presents a new method for creating realistic training examples by using system logs. These are like bookkeeping records of things that happen in the system.

This research examines how well different machine learning models can learn to detect intrusions using these realistic examples, compared to more traditional examples. Our findings revealed significant differences in their performance, highlighting the potential of our approach to provide a more effective training ground for these AI systems, helping them better identify and adapt to new types of cyber threats.

In essence, this work helps lay the groundwork for future research of these systems in cybersecurity defenses, ensuring that they are robust, adaptable, and up to the challenge of protecting our digital lives just as well as our home security systems protect our physical ones.

Abstract

This thesis explores and proposes a method for using aggregated system log messages, combined logs from multiple systems, to create realistic datasets for network intrusion detection systems (NIDS). By integrating these logs with network traffic data, a dataset was developed that enhances the training and testing of machine learning models for the anomaly-based detection component in intrusion detection systems. This work investigates the process of collecting and categorizing system logs into a structured format, facilitating the creation of activity clusters, groups of similar log messages, that aid in labeling network traffic. This approach produces datasets that not only mirror real-world data, but also align with security policies. The research further evaluates the performance of the following machine learning models in anomaly-based detection: Autoencoder, Convolutional Autoencoder, CNN, Isolation Forest, One-Class SVM, and XGBoost; using both artificial and realistic intrusion datasets. It was found that while some models, like the Isolation Forest, generalize well to unseen datasets, others exhibit a significant increase in false-positive rates. This highlights the challenges in model generalization across different datasets. The findings suggest directions for future work, including improving the generalizability of NIDS models, and the creation of realistic intrusion detection datasets using the proposed labeling method.

ACKNOWLEDGEMENTS

First and foremost, we would like to express our deepest gratitude to our supervisor, Reza Koshkangini. His guidance, patience, and expertise were invaluable to the completion of this work. His insightful feedback pushed us to sharpen our thinking and brought our work to a higher level.

We also thank Jesper Rude Selknaes, our contact person at the Data Management and Software Center at European Spallation Source. His insightful comments and continued support throughout this project have been greatly appreciated. His perspective was crucial in shaping our research and ensuring its practical relevance.

We extend our sincere thanks to the members of the opposition for their rigorous review and constructive critiques of our work. Their questions and comments were not only challenging but also deeply enriching, contributing significantly to the depth and breadth of our research discussion.

We are also immensely grateful to the examiner for their time and effort in evaluating our work. Their expertise and thoughtful judgment have been essential in ensuring the academic integrity and quality of our study.

Together, the contributions of our supervisor, contact person, the opposition, and the examiner have been instrumental in bringing this project to fruition. We are truly thankful for their collective wisdom and support.

CONTENTS

I. Introduction

Cybersecurity is the practice of protecting systems, networks, and devices from digital threats. The primary objectives of these cyberattacks include stealing or altering sensitive data, extorting ransom payments, or disrupting business operations [1]. Given the prevalence of digital devices, which now outnumbers humans, expected to reach 29 billion by 2030 [2], and the increasing sophistication of attackers [3], implementing robust cybersecurity measures to mitigate the increased cost and number of attacks [4], [5] is an increasingly complex challenge. Highlighting the importance of businesses and public-facing institutions to protect themselves from intrusions and the need for continued research and development of robust cybersecurity measures.

Organizations are high-value targets for malicious actors who are either attempting to steal valuable information or compromise the day-to-day running of these organizations for either economic or political reasons. Robust intrusion detection systems are an important part of effective cybersecurity solutions to allow organizations to protect themselves.

A. Introduction to Intrusion Detection

Intrusion Detection Systems (IDS) generally exist on the bridge or connection between the Internet and internal networks of an organization to detect potential intrusions or attacks and prevent them from causing harm [6]. The detection component of IDS commonly comes in signature- or anomaly-based versions. Where signature-based detection identifies intrusions depending on whether instances match known attacks stored in databases, and anomaly-based detection identifies intrusions depending on how much they differ from what is considered normal behavior [6].

B. Challenges with Intrusion Detection Systems

Purely signature-based IDS struggle to keep up with the times, as they are increasingly unable to identify novel attacks due to their reliance on referencing a database of known attacks. This requires the exploration of new and innovative approaches capable of dealing with the dynamic nature of cyber-threats of the future [7]. Although anomaly-based IDS are more suitable for detecting novel or unknown attacks, they often require comprehensive and realistic datasets to work in practice and regular realignment of what is considered normal behavior to remain accurate over time [7]. Not doing so may lead to potential problems, such as a high false positive or negative rate, where normal behavior is flagged as intrusions, or actual intrusions are missed altogether.

C. Machine Learning as a Solution

Using machine learning (ML) for anomaly-based detection is highlighted as a solution to the ever-changing nature of intrusions or attacks. The ability of machine learning to process large amounts of data allows it to find patterns that are indicative of malicious activity and be more adaptive to intrusions when new patterns emerge [7]. Furthermore, the different types of learning methods for ML, such as supervised and unsupervised, combined with the different types of ML algorithms, allow the trained ML models to find different patterns and perform differently depending on the characteristics of the dataset. To find out which algorithm performs best requires experimentation and large relevant datasets, preferably collected directly from where it will be used in practice or by using artificial datasets similar in nature.

D. Research Gap

Through review of related research, a gap is identified. Existing research reports promising results for ML models trained on research datasets for intrusion detection [8], [9], [7], [10]. However, critics argue that these datasets are flawed [11], [12], [13], [8], and that the results for models trained on these datasets are limited in their real-world applicability as the data do not reflect the complexities of modern cyberattacks [14]. Although ongoing efforts are intended to produce new and relevant datasets [11], [8], there remains a gap in the use of real-world data to contrast and contextualize research findings [14]. Although research has extensively explored the application of machine learning in IDS, there is a gap in evaluating the effectiveness of different ML algorithms for anomaly detection when considering applicability to real-world scenarios [7]. How well different types of learning methods for ML models, supervised and unsupervised, generalize between intrusion detection datasets [10], and how they generalize from artificial datasets to realistic data remains an open question.

E. Contributions

This thesis contributes to Cybersecurity and Data Science research by introducing a novel method for labeling raw network traffic using system log messages. It shows how machine learning researchers can generate an initial set of labels so that a model can be trained on real-world data with relative ease. This contribution is further strengthened by showing how the performance of machine learning algorithms commonly used in current research, trained on popular research datasets, generalizes when applied to real-world data. This research also contributes to bridging the gap between theory and practice by demonstrating the limitations and challenges of adapting real-world data for machine learning in intrusion detection, thereby urging researchers to explore new potential avenues rather than resorting to benchmarking of models using tried-and-true datasets with reduced real-world relevance.

Thus, the contributions of this work are summarized as follows:

1) A novel and qualitative method for labeling real-world network traffic captures using log messages produced by the same system. Such that the labeled network traffic captures can be used to train machine learning models for intrusion detection research purposes.
2) A comparative study and analysis of generalizability, for two supervised and four unsupervised machine learning models trained and tested on the artificial dataset CIC-IDS-2017 and the custom ESS-DMSC dataset. Hence, enabling the comparison of performance metrics reported in current research in the context of real-world data.

F. Purpose and Objectives

The aim of this research is to investigate the application of machine learning techniques for anomaly-based intrusion detection, specifically focusing on how different algorithms and data sets are used to create accurate and relevant ML models. By exploring the capabilities of machine learning in cybersecurity, this work has the goal of contributing to the ongoing development of more robust and adaptive IDS.

The specific objectives of this thesis are as follows.

1) Review existing research about data science applications for cybersecurity, focusing on the use of machine learning for anomaly-based intrusion detection.
2) Collect two datasets: one commonly used research dataset (CIC-IDS-2017) and create a comparable dataset, using the the necessary data, system logs and network traffic, provided by the external stakeholder (ESS-DMSC) for training the machine learning models in this work.
3) Prepare, describe and apply the processing and labeling method of the created dataset (ESS-DMSC).
4) Implement and evaluate six machine learning models for anomaly-based intrusion detection on both datasets. Covering both supervised and unsupervised learning methods.
5) Compare the performance of the models against each other using the metrics: overall accuracy, precision, recall, F1 score, AUC-ROC, and AUC-PRC.
6) Evaluate the models' generalizability by testing them on the dataset they were not trained on.

G. Research Questions

Based on the purpose and objectives of the thesis, two research questions are defined, the first to propose and apply a method to create a dataset, which is relevant in the requested need for more real-world or relevant datasets for intrusion detection research. The second question uses the created dataset to evaluate and compare machine learning model performance in terms of how well models generalize from research datasets to real-world data, in order to answer relevant questions raised in research of the transferability of research to practice.

- **Research Question 1:** How can system log messages be integrated with network traffic in order to create a dataset capable of training machine learning models for the task of intrusion detection?
- Hypothesis: It it possible to label the log messages such that the log messages can be used to label the packet payloads from the raw network traffic.
- Hypothesis: The labels can be used to create a realistic dataset that is comparable in format to an existing intrusion detection dataset for training machine learning models.
- **Research Question 2:** How does the performance metrics of machine learning models for intrusion detection compare when trained and tested on the same dataset versus when trained on one dataset and tested on an unseen dataset?
- Hypothesis: The performance metrics of the models differ depending on the model's suitability for the characteristics of the dataset.
- Hypothesis: There will be a loss in performance metrics for all models when tested on the unseen dataset as they are not able to fully generalize to the unseen dataset.

H. Research Justification

This research is motivated by the potential of data science, particularly machine learning, to address the limitations of current IDS. Machine learning algorithms have the ability to analyze large amounts of security data, identify complex patterns, and detect anomalies that can indicate potential intrusions. By investigating the effectiveness of different machine learning approaches for anomaly-based detection, this research contributes to the development of new and innovative IDS solutions.

In a practical sense, this research is also motivated by collaboration with the Data Management and Software Center (DMSC) [15] at the European Spallation Source (ESS) [16]. ESS is a European Research Infrastructure Consortium (ERIC) with 13 member states that facilitates multidisciplinary research and research collaborations internationally. DMSC is responsible for Scientific Computing and Scientific Data Management at ESS. Part of that responsibility is to ensure the integrity, availability, and confidentiality of computers, servers, systems, and data. Therefore, any investigation and development of the cybersecurity of their infrastructure is of great interest to them and shows the practical relevance of this thesis.

I. Significance of Research

The findings in this research can potentially benefit cybersecurity research by:

- **Bridging the gap between research and practical application.**
  The use of a realistic dataset alongside a commonly used research dataset allows for evaluating generalizability and contextualizing the findings for real-world implementation.
- **Informing the selection of appropriate machine learning algorithms for the development of anomaly-based IDS.**
  By comparing the effectiveness of different algorithms, this research provides valuable information for practitioners in the field.
- **Enhancing the adaptability of IDS to new attacks.**
  By identifying algorithms that excel at identifying unknown intrusions, research can contribute to the development of future-proof IDS solutions.
- **Contributing to the development of more robust anomaly-based IDS models.**
  Through the evaluation process, the research will provide valuable information on the strengths and weaknesses of different machine learning approaches. This knowledge can be used to design and develop more robust and effective anomaly detection systems.

J. Target Audience

This thesis is aimed at a broad audience with expertise in Computer Science, Cybersecurity, and Data Science. The primary target audience for this thesis includes;

1) **Data Scientists:** This group will be particularly interested in the following areas; exploration and evaluation of machine learning algorithms, data processing, and feature engineering.
2) **Cybersecurity Professionals:** This group will benefit from the insights gained on the effectiveness of machine learning for anomaly-based intrusion detection systems.
3) **Computer Scientists:** This group will find value in how novel representation is used to solve the problem of qualitative labeling.

The explanations in this thesis assume a foundational understanding of these fields. However, key concepts and terminology specific to anomaly detection or the chosen machine learning algorithms are clearly defined to aid comprehension and conceptual understanding. The language is clear and concise, with well-structured sections for easy navigation and reference.

K. Structure of the Thesis

This thesis is structured to guide the reader through a comprehensive exploration of anomaly-based machine learning techniques in the field of intrusion detection systems. Each chapter builds on the preceding content, providing a cohesive and structured presentation of the research.

- **Chapter 1: Introduction** - This chapter introduces the motivation behind the work, and the need for advanced IDS solutions in Cybersecurity and the limitations of research datasets. It describes the research questions, objectives and scope of the thesis, the intended audience, along with a short overview of scientific contributions.
- **Chapter 2: Background** - Is a continuation of Chapter 1, and serves as an introduction to Chapter 3. It provides a broader contextual background to the research and introduces fundamental terminology and concepts associated with intrusion detection systems.
- **Chapter 3: Related Work** - Constitutes an extensive review of existing research within the domain. Focusing on intrusion detection systems and the applicability of machine learning techniques. This chapter details the gaps in current research and establishes a theoretical foundation for the subsequent analysis in Chapter 4.
- **Chapter 3: Methodology** - Provides a detailed description of the applied method, including data collection, the preprocessing steps undertaken, selection and implementation of machine learning models, and the criteria for evaluation.
- **Chapter 4: Results & Analysis** - Presents a thorough analysis and examination of results obtained through experimentation. It details the performance of the models when applied to different datasets.
- **Chapter 5: Discussion** - Discusses the results in the context of the initial research objectives and related work within the domain. This chapter explores the implications of the findings, potential limitations of the study, and the applicability of the results to real-world scenarios.
- **Chapter 6: Conclusion and Future Work** - Summarizes the research findings, contributions to the field of cybersecurity, and the potential for future research.

II. Background

A. Terminology

- **0-day Attack** - An attack that exploits a vulnerability which the software vendor is not aware of.
- **Anomaly-Based Detection (AD)** - Intrusion detection based on identifying behavior that differs from what is considered normal behavior.
- **Deep Learning** - A branch of machine learning that performs automatic feature engineering through deeper models.
- **Feature Engineering** - The process of creating, extracting, transforming and selecting features in a dataset in order to make the ML algorithm focus on the most relevant parts of the dataset.
- **Internet Protocol (IP) Address** - An identifying address assigned to a device connected to a network.
- **Intrusion Detection System (IDS)** - A system that monitors a network or a group of systems for unwanted access or malicious behavior in order to block it.
- **Log Aggregation** - The combination of messages from multiple logs from different sources in order to make them usable as one for analysis.
- **Machine Learning (ML)** - The use of data and algorithms to train a model that is able solve problems it wasn't explicitly programmed to solve.
- **Network Packet** - A single instance of communication between two devices in a network. May contain Source and Destination IP addresses, Source and Destination Ports, Network Protocol ID, and a Payload.
- **Payload** - The part of the network packet that contains the data that is being transmitted.
- **Signature-Based Detection (SD)** - Intrusion detection based on matching behavior against already known malicious behavior.
- **Supervised Learning** - Training of machine learning models that uses labeled data to learn patterns.
- **TCP Protocol** - A protocol for network communication that requires the acknowledgement of the arrival of the packet at its destination.
- **UDP Protocol** - A protocol for network communication that does not require the acknowledgement of the arrival of the packet at its destination.
- **Unsupervised Learning** - Training of machine learning models that uses unlabeled data to learn patterns.

B. Theoretical Framework

The Internet is an ubiquitous part of our modern infrastructure and the potential ramifications of cyberattacks are devastating [3]. Not only are attacks more frequent and costly, they are also more sophisticated and increasingly commodified [4], [5], [3], [17].

The purpose of cybersecurity is to ensure the confidentiality, availability, and integrity (CIA) of computers, servers, systems, and data [18]. The word itself can be defined as "a series of measures taken to protect a computer or computer system against unauthorized access or attack" [19], or "any technology, measure or practice with the purpose of preventing or mitigating the impact of cyberattacks" [20]. Intrusions can be described as an attempt to compromise the CIA or bypass the security mechanisms of a computer or network [18], [6]. Intrusion Detection (ID) is 'The process of monitoring, events occurring in a computer system or network and analyzing them for signs of possible incidents' [6]. An Intrusion Detection System (IDS) refers to the practical implementation of ID software.

C. Technical Background

A **network** is a series of connected devices that allows communication between them. If the network is connected to the Internet to allow for communication on a global level, all traffic is directed through a firewall to shield it from the most common attacks. An illustration of a typical network configuration with an intrusion detection system can be seen in Figure 1. The switch is what allows communication between devices in the internal network and the Internet through the firewall. If an Intrusion Detection System exists, it can be connected at any point of the network to monitor it. In the case of monitoring network traffic, it is often done at the switch level between the firewall and any servers or client computers. This allows it to provide an extra layer of protection in order to detect any attacks before they have reached their intended target server or client computer.

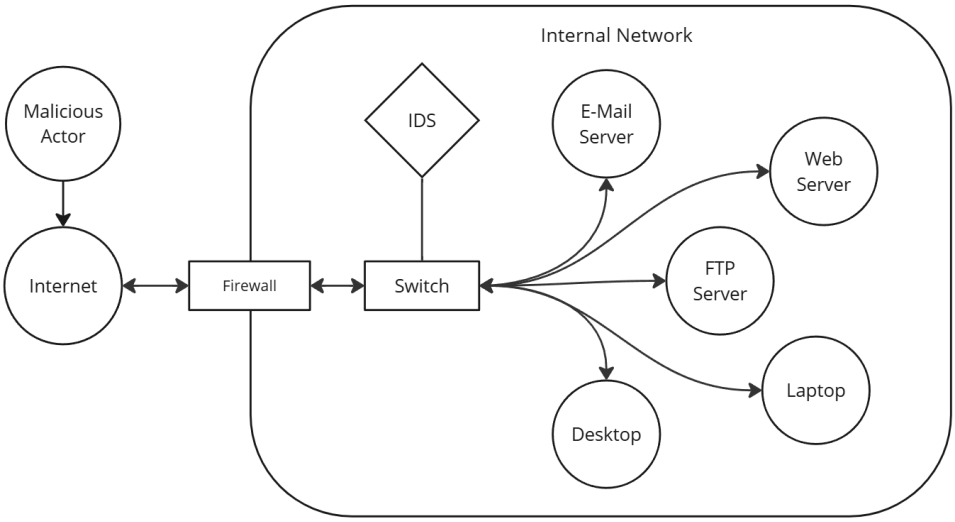

Fig. 1. A typical network configuration with a Intrusion Detection System

Communication between devices on a network is transmitted through network packets. A packet may contain different levels depending on the type of packet and between which types of devices it is being transmitted. An illustration of a packet with all levels in order for it to contain a payload is illustrated in Figure 2, where at the top level it contains a MAC address which is a hardware identifier, IP addresses that are assigned addresses to connected to the network, a protocol identifier containing ports, and finally the payload which is the data being transmitted.

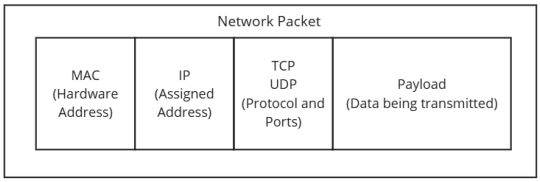

Fig. 2. An example of a packet with all the levels to contain a payload.

Intrusion Detection Systems (IDS) are an integral part of modern security systems and are critical to safeguarding networks by identifying and preventing unauthorized access or malicious activity [6]. An illustration of a Network Intrusion Detection System can be seen in Figure 3.

Intrusion detection systems (IDS) are categorized primarily into three types depending on where they are located; **Network Intrusion Detection Systems (NIDS)**, which are set up at strategic points within a network to observe traffic passing through the network. **Host-Based Intrusion Detection Systems (HIDS)**, which run on independent hosts or devices and monitor incoming and outgoing packets from that device. **Protocol-based Intrusion Detection Systems (PIDS)** are located at the front end of a server, monitoring and interpreting the protocol between a device and the server. There are also hybrid approaches that combine multiple complementary methods [18], [21].

The detection component of intrusion detection systems (IDS) can be broadly classified into **signature-based detection (SD)** and **anomaly-based detection (AD)**. Signature-based systems identify intrusions by matching network activity with known attack patterns stored in a signature database [7]. They are generally very effective against well-documented attacks and widely used hacking tools, but are unable to accurately identify novel attacks. In contrast, anomaly-based detection systems (AD) profile the normal network behavior and identify intrusions by detecting deviations from the established profile. IDS can use either forms of detection or a hybrid of both as the one illustrated in Figure 3.

If an IDS detects a potential intrusion, it has the option to alert an administrator to allow them to appropriately handle the intrusion, such as manually taking defensive actions. Some IDS can also automatically take defined preventive measures such as blocking the IP address behind the intrusion or shutting down servers or services being targeted. In these cases, IDS can also be known as an Intrusion Prevention System (IPS) to signal their more active role.

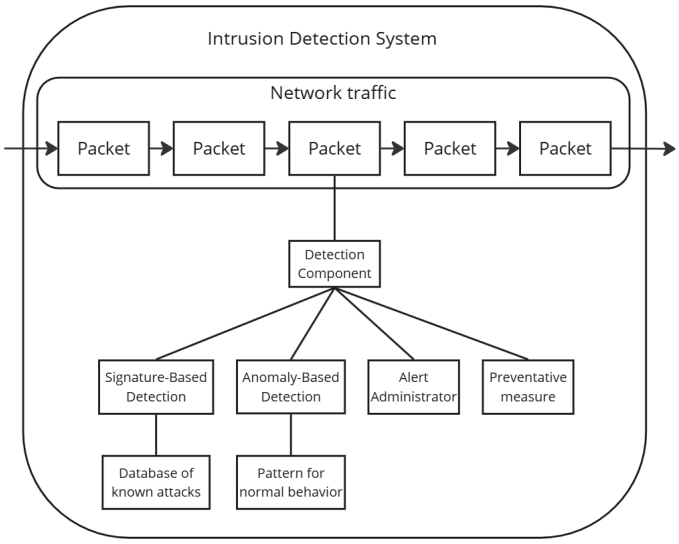

Fig. 3. Network Intrusion Detection System Components.

The anomaly-based detection component can be implemented in several different ways, such as manually defining rules for what is considered normal, outlier detection through statistics [22], or through more advanced learning methods using Machine Learning (ML) techniques. Machine learning with its ability to learn complex patterns in large amounts of data has emerged as a powerful tool for developing anomaly-based IDS [7]. ML also allows for regular retraining of models in order for the anomaly-based detection component to be more adaptive as the nature of the monitored traffic changes.

III. Related Work

The related work reviewed within the area of data science, specifically machine learning, in cybersecurity shows a high level of activity in the creation and use of different datasets designed to mimic real-world scenarios. There is also research on data preprocessing and aggregation in how to best utilize datasets for intrusion detection. Finally, a lot of attention is paid to training and testing different ML models to achieve the best possible performance metrics in the detection of intrusions.

A. Datasets

Several publicly available datasets are available for research in intrusion detection. A survey has reviewed 13 of these and categorized them based on real-life or benchmark. The benchmark datasets include DEFCON2000/2002, UNM, CAIDA2002/2016, LBNL, CDX 2009, ISX2012, UNSW-NB15, with DARPA 98, KDD Cup99 and NSL-KDD being the most popular for evaluating IDS implementations [23], For real-life the following three datasets have been surveyed, KYOTO, UNIBS and ISCX UNB, while these attempt to simulate real-life they have been generated using commonly used algorithms for attacks and are artificial in nature [23]. These datasets also have the limitation of being old, by 10 years or more, and do not reflect modern attacks or recent improvements within the intrusion detection area.

1) CIC-IDS-2017: The Intrusion Detection Evaluation Dataset (CIC-IDS-2017), published by the University of New Brunswick, Canada, is a well-known cybersecurity dataset designed to mimic real-world network traffic and cyberattacks [11]. This dataset covers a span of five days, Monday to Friday, with Monday being a normal day with only benign traffic and each subsequent day containing a different subset of attacks in addition to the benign background traffic. Attacks include variations of Brute Force FTP, Brute Force SSH, Denial of Service, Heartbleed, Web Attack, Infiltration, Botnet, and Distributed Denial of Service, for a total of 13 different labels.

CIC-IDS-2017 also includes a network traffic analysis and parsing tool named CICFlowMeter, which can extract $\tilde{8}0$ features from a given Packet Capture file (PCAP), the most important of which are the source IP, the destination IP, the source port, the destination port, and the protocol. This can be used to organize the traffic into bidirectional sessions or the flow of payload data. The output also includes statistical measures of sessions at a higher level, as well as statistical measures of the packet bytes sent and received within a session. The documentation for all features can be found on GitHub [24].

Researchers at the University of New Brunswick created CIC-IDS-2017 in response to limitations in existing datasets, such as the lack of modern attack representations and insufficient metadata. The dataset was carefully developed to include previously missing features and is based on the most prevalent attacks reported in the 2016 McAfee report. It has been validated using common machine learning algorithms, with Random Forest showing high accuracy and efficiency [11].

Despite its extensive use in research for benchmarking intrusion detection systems, CIC-IDS-2017 has faced criticisms [9], particularly concerning the deterministic nature of its simulated attacks and potential error in attacks and data [25], [13], Since attacks are generated by the use of hacking tools, resulting attack signatures are well-known and already effectively countered by traditional security software with near perfect accuracy and minimal false positives [14]. Critics also suggest that the dataset's summarized traffic data, that is, the output from CICFlowMeter concentrates the most valuable signals in just a few features, which limit the effectiveness of derived models [14], [12]. As such, using PCAP files is recommended for deeper analysis and model training. Despite these criticisms, CIC-IDS-2017 and its more comprehensive successor CSE-CIC-IDS2018 are widely used in machine learning research for cybersecurity, as they remain a valuable resource for developing and testing intrusion detection systems.

2) Limitations of intrusion datasets: Despite advances in anomaly-based IDS, there are themes of recurring limitations in the reviewed literature. A survey published in 2020 presents an overview of data science within cybersecurity, identifies multiple limitations of current research, and suggests directions for future research. Such as the creation of higher quality cybersecurity datasets that better reflect recent attacks, the investigation of better ways to handle quality issues in cybersecurity datasets, the need for more adaptive generation of security policy rules, the prediction and securing of most valuable security information, the use of context-aware solutions to leverage additional data sources outside of the security domain, the research into anomaly-based detection to overcome the limitations of signature-based systems that comprise most systems [7].

Machine learning research is often burdened by the collection of data. Network traffic, although abundant, can be very difficult to access for research purposes. The reason being the sheer volume of data, an irregularity of attacks, and a difficulty in labeling attacks. If security data are available to researchers through organizations, it is often difficult to share publicly due to security concerns [11]. Therefore, efforts have been made to create artificial datasets that are publicly available to researchers, one popular source of data in contemporary research is the Canadian Institute of Cybersecurity (CIC) in collaboration with the University of New Brunswick (UNB), Canada [11], [26].

However, the difficulty in finding usable data can lead researchers to use the same data selection repeatedly. This can be good for comparing and benchmarking models against a common metric. However, it also raises questions about the validity of research, since artificial datasets are not representative of real world attack scenarios [14].

Moreover, commonly used research datasets are often labeled, and applied methods are supervised, which generally perform well given labeled data. As an example, one paper applies a transformer model to perform classification on the CIC-IDS-2017 and CIC-DDos-2019 datasets, with an F1 score of 99.17% and 98.48% respectively [27]. In contrast, unsupervised models are less common, while simultaneously being more likely to be suitable for handling real-world attacks due to their ability to handle day-zero attacks [11], [14], [10].

B. Data Preprocessing & Aggregation

A common practice in machine learning is to use existing features in the dataset and either augment them directly or create other new features using mathematical calculations. This is a labor-intensive process that is performed because the performance of most machine learning models depends on the quality and types of features used for input when training [28]. Feature engineering may involve the creation of completely new features, the transformation of existing features to make them more suitable as input to the ML algorithm, the extraction of features to make them provide more relevant information for training, and the reduction of features through feature selection. The application of feature selection can greatly affect the performance of the machine learning algorithm, as it allows it to focus on what is most relevant while reducing the dimensionality of the dataset [29]. Therefore, feature engineering performed on data sets for training models can be a focus area in itself and has been suggested as a future direction of research for data sets for cybersecurity [7].

There are a variety of feature selection techniques that are applied to intrusion detection data. In one example, feature selection for both the UNSW-NB 15 dataset and the Network TON_IoT dataset is done through a Gini Impurity-based Weighted Random Forest (GIWRF) model. The features were reduced from 42 to 20 and from 39 to 10, respectively. Thereafter, binary classification was applied to both the full and reduced feature sets using the following algorithms; Decision Tree (DT), Gradient Boosting Tree, Multilayer Percepton, AdaBoost, LSTM and GRU. Of the algorithms compared, the DT improved its overall accuracy the most. Reporting an increase of 2.5% in accuracy, 5.24% in precision, 1.87% in F1 score, and 7.7% in false positive rate, after reduction of UNSW-NB 15 [29]. This study shows the benefits of feature selection when working with certain algorithms, allowing improvements in important metrics for intrusion detection such as accuracy, F1 score, and false positive rate. Another benefit of the feature section is the reduction in the time complexity of the algorithms and the space complexity of the dataset when training models.

Real-time performance, or the time from observation to classification, is also an important metric for intrusion detection systems, and therefore, when using certain machine learning algorithms, it can be beneficial to reduce the time and space complexity of the dataset to reduce the classification time [30]. Doing so produces an Intrusion Detection System that is quicker to react and is able to keep up with the large amount of traffic that a network generally produces.

The application of data science in cybersecurity also allows the collection and integration of tangentially related data to improve the context awareness of intrusion detection systems. One such example is the use of SVM and LSTM in order to predict Distributed Denial of Service (DDoS) attacks based on social media posts. Although not perfect, it still shows promise by identifying a relationship between negative posts on social media about a specific target and DDoS attacks against them, which can be used to predict attacks days in advance [31]. The accuracy of this approach has been further improved by including an improved Convolutional Neural Network for better sentiment analysis of social media posts and an improved LSTM model to achieve greater accuracy in predictions [32].

C. Machine Learning in Intrusion Detection

The ability of machine learning models to learn nonlinear relationships in data shows promise over traditional statistical methods when applied to cybersecurity problems. According to the reviewed literature, learning-based methods tend to perform better on average [33]. However, statistical methods can perform well when there are clear linear relationships [22]. The suitability of a given model generally depends on the context of the security problem, as models of different complexity can perform equally well [7]. Learning models also seem to have improved fault tolerance [18]. The well-performing methods for anomaly detection range from less complex models, such as Support Vector Machines (SVM), Decision Trees, k-Nearest Neighbor, Random Forest to more advanced methods, e.g., Artificial Neural Networks (ANNs), Recurrent Neural Networks (RNNs), Long-Short-Term Memory (LSTM), Deep Convolutional Networks (CNNs), and Reinforcement Learning (RL) [7], [18].

There are also hybrid approaches that combine multiple complementary methods [18], [21]. Signature-based detection works by comparing known intrusion patterns, that is, signatures, to recorded events. The detection references a database of known malicious patterns to determine whether the observed pattern is a threat [18], [7]. Anomaly-based detection works by profiling the normal behavior of a subject within a system. A profile conveys the expected behavior and can be derived from monitoring activities, connections, hosts, users, etc. over a period of time. These profiles can be compared with observations to identify abnormalities of behavior in relation to an established normal. Any outliers are then indicative of security threats or other malicious behavior [18], [7].

Anomaly-based IDS and Signature-based IDS can utilize both supervised and unsupervised learning methods. Anomaly-based models are often unsupervised, but can use supervised learning if labeled data is available, or limited supervised learning if data is incomplete. Signature-based models typically use supervised learning as they rely on known attack vectors stored in signature databases. These systems use labeled data. According to research, unsupervised models are less commonly researched, while simultaneously being more suitable for handling real-world cyberattack scenarios. Part in fact to their ability to learn from large amounts of data what is considered normal behavior, rather than being explicitly told what patterns to look for to identify attacks as in supervised. This form of learning allows unsupervised models to react as soon as abnormal behavior occurs, such as in the case of 0-day attacks [11], [14], [10]. A probable reason for this underrepresentation of unsupervised models is the fact that popular benchmark datasets, such as CIC-IDS-2017, are generally labeled [11], [14].

Generalizability and overfitting are other aspects to consider when transferring models between datasets, especially if the same selection of research datasets is recurring. A study used a cross-validation approach to estimate model generalization and found that models trained on CIC-IDS-2017 do not transfer well to the CSE-CIC-IDS-2018 dataset, showing a drop in average accuracy of 25.63%, indicating high overfitting to the initial dataset, despite the relative similarity between the two. The authors suggest the investigation of high-level feature engineering in order to capture more common patterns between datasets, and the study of fine-tuning the hyper parameters of the trained models using a very small part of the unseen dataset through transfer learning [10].

1) Unsupervised models: While popular datasets used for benchmarking intrusion detection models, such as CIC-IDS-2017 [11], are labeled, which means that the majority of applied learning methods are supervised, the application of unsupervised learning methods is not completely uncommon. One example of research into the unsupervised models applied to the CIC-IDS-2017 dataset is the comparison of One-class SVM, Isolation Forest, Autoencoders and PCA, where the motivation for the use of unsupervised learning is its potential in detecting zero-day attacks, or in other words: unknown attacks as soon as they happen. In contrast to supervised learning, which is recognized in its ability to excel in accurate identification of attacks similar to known attacks. In evaluation of the models tested on the CIC-IDS-2017 dataset, it is found that the Autoencoder is the best with an AUC-ROC of 0.978, followed by One-class SVM, Isolation Forest, and PCA in order. In addition, a One-class SVM outperforms the Autoencoder when reducing the false positive rate is preferred over catching every potential attack [34].

Another approach for unsupervised learning is the combination of Autoencoder with Isolation Forest for binary classification, as normal or attack, of incoming packets to the infrastructure devices supporting the Internet of Things. Although Autoencoder alone provided an already high accuracy, together they resulted in an overall accuracy of 95.4% on the NSL-KDD dataset, higher than other state-of-the-art intrusion detection systems. One main benefit of using Autoencoder for training learning models for intrusion detection is its ability to overcome potential class imbalance in the dataset. In this case, it is only trained on one class, normal traffic, to reconstruct it as well as possible. Then it will have issues reconstructing traffic that contains attacks, the underrepresented class, and therefore identifying them [35].

Another suggested approach is the use of a Deep Sparse Autoencoder (D-SAE) to compress the features in the CIC-IDS-2017 intrusion dataset before using Random Forest for classification. It showed higher accuracy, shorter training time, and shorter testing time than without compression. It also outperformed other feature selection methods such as Pearson's method and a SAE combined with SVM [30]. However, one of the limitations of the suggested approach was the low accuracy in the classification of underrepresented classes. Highlighting the necessity to handle imbalanced datasets when training models on intrusion datasets.

2) Supervised models: A hierarchical packet-based CNN (PBCNN) is proposed under the motivation that network traffic has a hierarchical structure of byte-packet-flow, and instead of using the feature-engineered CSV files found in the CSE-CIC-IDS2018 dataset, the raw PCAP files and their intact bytes-packet-flow representation can be used to leverage information that otherwise would be lost during feature engineering. The first level of the network reads the bytes from PCAP-files and creates a bytes-to-packet hierarchical representation, and the second level creates a packet-to-flow hierarchical representation

for classification. The proposed PBCNN network achieves 98.2% accuracy, 98.3% recall and 98.3% F1 score on the CSE-CIC-IDS2018 dataset, slightly outperforming the second best compared method of ID3 [36].

In a study comparing the performance of models on NSL-KDD, a network intrusion dataset, Extreme Gradient Boosting (XGBoost) was found to be highly efficient and effective. Demonstrating an accuracy of 98.70% in the classification of intrusions. The models XGBoost were compared with were; Support Vector Machines (SVMs), Radial-Basis Functions (RBFs), Neural Networks (NNs) and Decision Trees (DTs). The study concludes that XGBoost, compared to the models examined, has an advantage in speed, flexibility, efficiency, and the ability to handle sparse input [37].

3) Deep Learning: A survey published in 2021, looking at recent developments and challenges of machine learning and deep learning methods for intrusion detection, came to the following conclusions. Machine learning methods report good performance in many cases where the data do not present very complex relationships. Machine learning requires extensive feature engineering and expert knowledge of the domain area. To alleviate these issues, the research into Deep Learning (DL) methods in IDS has gained interest in recent years, due to its ability to learn features and allow for more flexibility in handling newer and more complex problems. The challenges still present for DL models include their adaptability to the nature of attacks that change day by day, the generalization of DL models that rely on extensive labeled datasets, and the black-box nature of DL models, which does not allow insight into how they come to a certain conclusion [23].

As more devices and different types of devices, for example, Internet of Things, are connected in large networks, the volume and velocity of traffic data generated by networks are increasing, which poses a problem for traditional intrusion detection systems. Deep learning methods, because of their ability to handle large amounts of complex data, represent a current research effort into anomaly-based detection systems [38].

With the rise of popularity of the Internet of Things and its many physical devices connected in large networks, the amount and level of sophistication of attacks against these have been increasing. In relation to this, the volume and velocity of traffic data generated by these have also grown, which poses a problem for traditional intrusion detection systems. To address it, deep learning methods and their ability to handle large quantity of complex data is leveraged through a proposed approach combining transfer learning from ResNet to capture spatial features, with Transformer and Bidirectional LSTM to capture temporal features in network traffic data for detection and classification of attack-type. On the CIC-IDS-2017 dataset the proposed approach achieves an overall accuracy of 99.15%, an 0.2% to 10% increase to compared methods [38].

Another paper applies a transformer model to perform classification on the CIC-IDS-2017 and CIC-DDos-2019 datasets with an F1 score of 99.17% and 98.48% respectively. This corresponds to an 4.56% improvement over RNN, and 1.67% improvement over LSTM. Another advantage of the proposed solution is the ability to perform classification in real time. In future work, the authors are also considering the use of meta-learning to address the few-shot classification problem [27].

D. Summary

In the reviewed literature, there exists an abundance of machine learning models for anomaly-based detection that report promising results [35], [38], [27], [36], [37]. However, proposed models are applied almost exclusively to established datasets that, according to critics, are limited in their real-world applications and are of little to no use for other purposes than benchmarking model performance [14], [11], [9], [8].

It is important to understand that the datasets used in research are regularly developed and limitations pointed out by critics are often addressed in subsequent iterations, e.g. CIC-IDS-2017 and CIC-IDS-2018. In other words, there are ongoing efforts to produce new data for research that is relevant to the security concerns of today [23], and for models that are trained using these datasets, the efforts are most often focused on novel architectures [38], [36], [27] to maximize performance metrics.

With these efforts in mind, there is still a lack of real-world data to aid in the investigation of the practical implications of model results in intrusion detection [14]. That is, application of models to real data or real testing scenarios, to understand how well the model performs in circumstances that are more similar to real life.

Therefore, multiple gaps are revealed in recent research. The validity and real-world applicability of models trained on artificial research datasets [11], [14], the comparison of performance metrics in terms of generalizability between artificial and real-world data [10], as well as the repeated use of supervised models on outdated and solved attack vectors [14], the limited comparison between supervised and unsupervised performance on the same data, as well as limited research on labeling real network data.

These gaps can be addressed by the using real-world datasets for intrusion detection, the training and testing of supervised and unsupervised machine learning models on the same datasets for a comparative analysis, the measuring of generalizability

of these models between the real-world dataset and a commonly used dataset in intrusion detection research, and developing a method of labeling real network data.

In summary, the reviewed literature highlights the potential of machine learning for anomaly-based intrusion detection. Various machine learning algorithms, both supervised and unsupervised, have shown promising results when applied to commonly used datasets. However, more research is needed to address the limitations of using common benchmark datasets, which may not fully represent real-world network traffic. In addition, a comprehensive evaluation is warranted that compares the effectiveness of different types of machine learning algorithms for anomaly-based detection in intrusion detection systems.

IV. Methodology

The methodology of this work follows the main steps and principles of Design Science. It is a problem solving approach that aims to contribute to research through the creation of artifacts [39]. It consists of six steps generally performed in sequential order, with room for multiple iterations of steps, allowing for improvements after feedback has been collected. The Design Science steps are visualized in Figure 4, beginning with Problem Identification and Motivation and ending with the Communication of the research findings.

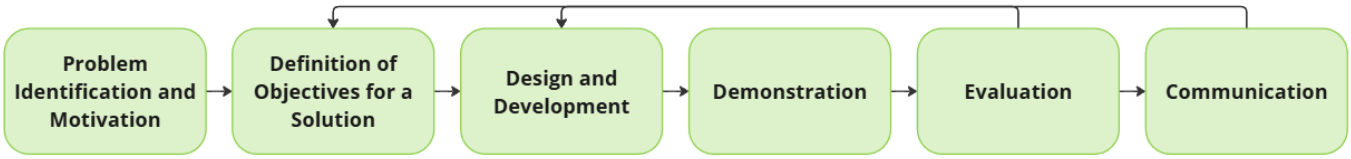

Fig. 4. The steps of Design Science applied in this work.

The Design Science steps and how they are applied in this work:

1) **Problem Identification and Motivation:** A literature review is performed on current research within the domain and data is collected in order to establish and motivate the research problem.
2) **Definition of Objectives for a Solution:** Using the reviewed literature and the research problem to define the purpose, goal, objectives and research questions. In a way that clearly indicates what will be achieved with the research in this work.
3) **Design and Development:** The design and development of two artifacts, a labeling method and ML models, in order to achieve the objectives and to answer the research questions.
4) **Demonstration:** Applying the labeling method and training the ML models in order to demonstrate the use of the artifacts.
5) **Evaluation:** Testing the ML models and evaluating their performance on the data labeled using the new labeling method.
6) **Communication:** Writing and presenting the findings from the evaluation of the artifacts.

The selection of Design Science as the research methodology for this work is motivated by its focus on problem solving, where it provides a structured method to solve problems [39] such as the labeling method and ML models for anomaly-based intrusion detection in this work. It also provides a structured framework for creating innovative artifacts through iterative development [39]. This is similar to the core principle of the experimental nature of the creation of machine learning models. This means that several iterations and attempts at different solutions are often necessary to produce the best possible performing models. Design science also places an emphasis on research that is relevant in practice [39], which is part of the purpose of this work, where the contribution of knowledge aims to be directly relevant to the external stakeholder.

Other research methodologies such as Experimental Research was not chosen because of its rigidity in controlled experiments do not provide the flexibility needed in the development of machine learning models, where adjustments may be needed on-the-fly. Case Study Research was not chosen because it would be too in-depth and focus on a single case which would not capture the breath of the different solutions needed in the ever-changing field of machine learning development. Survey Research was not chosen for the opposite reason, where collecting data from multiple sources was not necessary because of the partnership with the external stakeholder, and it would lead to a less focused thesis.

A. Problem Identification and Motivation

Identifying and motivating the problem is achieved through discussion with the external stakeholder ESS-DMSC and through a review of the literature where the main objective is to explore and understand the current state of research in the domain area. The contact person at ESS-DMSC is experienced and knowledgeable in the area of cybersecurity and has a good understanding of the systems from which the data used in the work is collected. Therefore, the discussion with them established the general problem and the relevance of it on a practical level. Through this discussion and the following review of the literature, a purpose, a goal, several objectives, and two research questions are defined.

The selection of a literature review as a research method is motivated by its efficiency in synthesizing knowledge by reviewing articles with different perspectives in the same domain area, identifying potential approaches to assist in solving the problem and to ensure that the approach remains relevant and novel to the research field. Data collection is also performed as part of this step in order to assist in formulating a relevant problem that can be solved using the available data.

1) Literature Review: The literature review in this work consists of the following steps, the definition of search keywords, the definition of inclusion and exclusion criteria, the selection of journal databases to search and the selection of articles based on their relevance to the problem [40].

The search keywords are defined on the basis of the research questions to find the most relevant articles. In this work, three main search concepts have been established, and their keywords can be seen in Table I. The first concept is the field of data science and its sub-areas, the second concept is the domain of the problem statement, and the third concept is different approaches or resources for answering the research questions.

Concept 1	Concept 2	Concept 3
Machine learning	Intrusion detection	Prediction
Data science	Hacking attacks	Classification
Statistics	Cybersecurity	Anomaly detection
Deep learning	Information security	Analysis
		Datasets

TABLE I
SEARCH KEYWORDS FOR LITERATURE REVIEW

The following inclusion and exclusion criteria have been defined in order to ensure that the selected articles are relevant to the problem statement, aid the research in this thesis, and meet a certain level of quality.

- Articles that are not relevant to Data Science and Cybersecurity are excluded.
- Articles that do not follow a scientific writing structure are excluded.
- Articles with poor or insufficient references are excluded.
- Articles that are not peer reviewed are included but are considered appropriately.

The following journal databases are used to search for articles in order to further ensure that the articles found meet a certain level of quality.

- IEEE Explore
- ACM
- Springer Link
- Elsevier

The sections of the articles found are then read in the following order: title, abstract, introduction, conclusion, results, and methodology. If at any point the article isn't considered relevant anymore, it is discarded. If the article is considered relevant, the relevant key points are summarized and critically reviewed in the Related Work section.

2) Data Collection: There are two primary sources of data; the artificially generated Intrusion Detection Evaluation Dataset (CIC-IDS-2017) published by the University of New Brunswick, Canada [11], and the real world data provided by the Data Management and Software Center at ESS [15] (ESS-DMSC). These two sources of data differ significantly. The artificial dataset (CIC-IDs-2017) is static, labeled, and documented as it is preconstructed and available as a standardized dataset [11]. In comparison, the data provided by ESS-DMSC is not predefined as it consists of network traffic and system information captured in real-time. It is therefore without labels and documentation before any processing is applied.

In order to compare machine learning models trained on the two datasets, the number of classes in each is reduced to binary, that is, malicious or benign. The reason being is that the ESS-DMSC dataset is dynamic and therefore contains an undefined number of attacks that are not known in advance. Thus, any direct comparison between the ESS-DMSC data and CIC-IDS-2017 requires either the same subset of cyberattacks present in both datasets or a reduction of the problem from multiclass classification to binary classification. Reducing the number of classes can also be justified by multiclass classification making data more difficult to work with, especially if the data is unbalanced [41].

The variations of cyberattacks present in CIC-IDS-2017 can be found potentially in the ESS-DMSC data. However, finding matching types of attacks between the two sources of data is not suitable for the scope of this work, as it would require continuous monitoring of network traffic, in-house domain expertise, and cybersecurity domain expertise. Thus, the types of cyberattacks in each dataset is reduced to the common denominator, which is malicious and benign traffic. This reduction is also motivated by the hypothesis that binary and malicious packet payloads are of two different distributions that are separable.

3) ESS-DMSC Data: The data provided by ESS-DMSC is composed of two components; raw network traffic captured external to the primary firewall (PCAP) and corresponding system log data (CSV), which are produced as a result of the aforementioned network activity interacting with the systems behind the firewall. The log data are aggregated by Graylog [42], an aggregation tool designed to consolidate system log data produced by various components within a cohesive system. Due to the diverse nature of these components, the contents of log messages vary greatly. Among these messages, those from host-based intrusion detection systems (HIDS), in this case OSSEC [43], are particularly informative. These messages can

sometimes be used directly to pinpoint malicious activity. In contrast, other log data may not be directly indicative of security issues. Instead, they reflect routine behavior, changes in the system, notifications, warnings, etc. The utility of information varies between messages, as some are more descriptive, e.g., referencing a particular user, or IP address, in its description. While other messages offer less utility as they only contain a timestamp and a generic statement, e.g., ”Connection blocked by TCP wrappers”, without referencing any identifiers, such as user or IP address. The CSV files produced by Graylog [42] are subject to inconsistencies and a different database schema. These inconsistencies arise from various factors such as message originator (e.g., system component), the range of system information available at the time, and the type of information that ESS-DMSC is permitted and prepared to disclose.

Both network traffic and log messages depict current network traffic and system activity rather than a predefined set of intrusion attempts. Therefore, the data is not organized with the intent to categorize its contents by default, it is simply a rendition of the current security landscape at ESS-DMSC. As such, there are no pre-defined set of labels, and available documentation is limited to secondary sources pertaining to components present in the network, such as Graylog and OSSEC [42], [43], rather than the traffic itself, which is available for CIC-IDS-2017 [11]. The data is captured in different time-windows and therefore not as comprehensive, information-dense, and organized, as a typical research dataset might be. For instance, CIC-IDS-2017 displays a variety of attacks over the span of a week, whilst providing a separate day containing only benign traffic used for profiling the system [11], [7]. In the ESS-DMSC data no such distinction exists as there is no pre-defined benign traffic, nor a guarantee of any particular cyberattack. It is therefore important to understand that the ESS-DMSC data is limited by the contents of the given time-window, and that the work concerns the contents of the data, not the contents of the networking landscape and systems from which it is captured.

B. Definition of Objectives for a Solution

Based on the literature review and the established problem and purpose, objectives are defined in such a way that they aid in or directly answer the research questions. As part of design science, the objectives for a solution are adjusted or redefined multiple times as the demonstration and evaluation steps produce results. Thus, changing the objectives to be more appropriate for solving the problem of the work.

In this work, the final objectives involve the creation of a new labeling method using aggregated log data, the creation of a realistic dataset for intrusion detection using the new labeling method, the implementation and training and of machine learning models using six different algorithms, and finally the testing of the models on the dataset on which they are trained, but also the unseen dataset in order to measure generalizability.

C. Design and Development

The design and development in this work involve data pre-processing to ensure it is ready for ML model training, data labeling to allow training using supervised learning methods and evaluation of model performance, and finally ML model implementation.

1) Data Processing: The data processing is divided into four sections; network traffic processing, log message processing, labeling and data cleaning. An overview of the processing sequence is depicted in figure 5. The network traffic processing, labeling and data cleaning, are applied to both datasets. While the log message processing is only relevant to the ESS-DMSC dataset. The labeling primarily relates to the ESS-DMSC dataset since it involves the creation of new labels, while CIC-IDS-2017 only requires matching the labels present in the original pre-labeled CSV files to the output generated from the network processing section. Data cleaning is applied to both datasets follows data cleaning principles established in the book ”Data Processing in Data Mining” by Luengo et al [44].

2) Network Traffic Processing (PCAP): The raw network data captured in Packet Capture (PCAP) format requires initial parsing to be transformed into CSV files. The extracted information includes; packet identifiers such as; Source IP, Destination IP, Source Port, Destination Port and Protocol, and binary payload data if the packet contains such information. Identifiers refer to any information that can be used to identify a specific transaction between the sender and receiver. The packet payload refers to the binary contents of the transaction.

The packet processing is a combination of two parts, illustrated in Figures 6 and 7. The first part of the sequence is the extraction of identifiers and payload data from the packet, which is done using the Python library Scapy [45]. It provides a toolkit for reading and extracting information from PCAP files. Since the files are very large, it is not possible to read the files into memory at once and thereafter process the contents in one go. Instead, PCAP files are read one packet at a time, while simultaneously writing the extracted contents of the packet to CSV. The extraction process filters the packets depending on the network protocol and writes each protocol to a separate CSV file. Only two networking protocols are considered, these are the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). The reason being that the majority of traffic

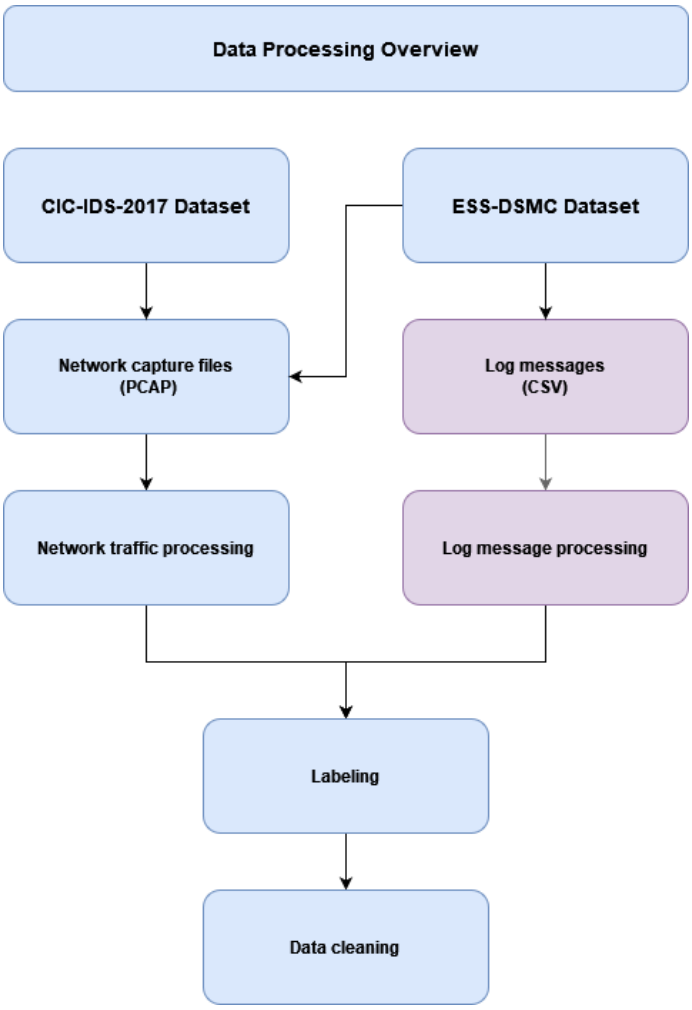

Fig. 5. An overview of the data processing steps for the two datasets.

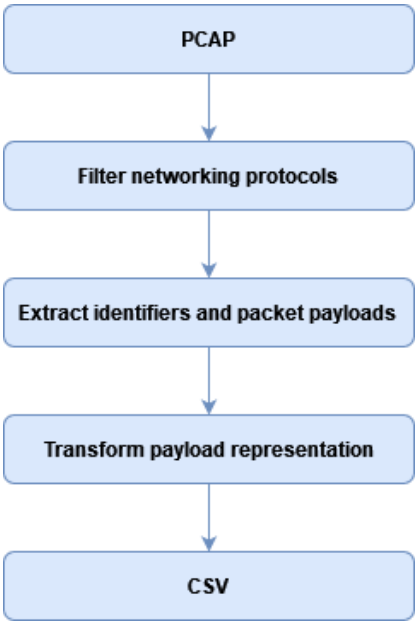

Fig. 6. Overview of network traffic processing part one (PCAP)

in both datasets uses these two protocols. In addition, both protocols use IP addresses that are required to match individual network packets with associated labels.

When the payload data is extracted it is also converted from a binary representation to a numerical representation. This is done to match the expected input of the machine learning models. Since the payload is stored as a series of bytes, each a group of 8 bits, can be represented as a number between 0 and 255. This conversion is done according to equation 1. The product of each bit and its corresponding power of 2 are summed up to a numerical value. These series of numerical values are stored as vectors of variable length. The length of each vector correlates directly with the size of the payload in the packet.

$$v = \sum_{i=0}^{7} b_i \cdot 2^i \tag{1}$$

The resulting numerical values are then normalized using standard Min-Max normalization (Equation 2) and thereby converted to values between 0 and 1. Where v_{min} is 0, v_{max} is 255, v is the input value, and v_{norm} is the resulting normalized value. This is done to ensure that every input value influences the machine learning algorithms equally.

$$v_{\text{norm}} = \frac{v - v_{\text{min}}}{v_{\text{max}} - v_{\text{min}}} \tag{2}$$

After the extraction process is completed, the resulting CSV files undergo additional processing, illustrated in Figure 7. Here, data types of different columns are standardized to one single data type. The timestamps are converted from Unix format to Pandas datetime format, normalized to milliseconds, and adjusted for time zone differences. The CSV files are then stored in chunks using Parquet format. This is done to make file sizes manageable and to preserve data types when reading and writing data to memory.

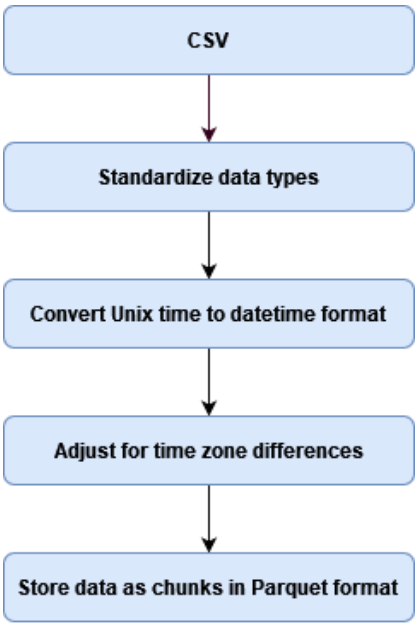

Fig. 7. Overview of network traffic processing part two (PCAP)

In summary, the processing involves stream reading the PCAP files and extracting identifiers and payload data. The payload data is transformed to fit the expected input of the machine learning models. The data is then filtered on TCP and UDP protocols and then written in CSV format. The resulting CSV files then undergo additional processing to standardize the data types within the files and are then stored as chunks to conserve memory. An overview of the entire process is illustrated in Figures 6 and 7. The resulting data are now ready to be matched with the associated labels for each dataset. For CIC-IDS-2017 the labels from the original pre-labeled CSV files are fitted to the packet payloads based on identifiers. For the ESS-DMSC data, matching packet payloads to labels requires additional processing and the creation of new labels since none exists. This is done in the following section.

3) Log Message Processing: As seen in Figure 5 depicting an overview of the entire data processing pipeline. Log message processing is only relevant for the ESS-DMSC data. In this section, aggregated log messages from separate components in the network are transformed and organized into distinct events. The purpose of organizing the log messages into events is to allow for the classification of data points as either malicious or benign based on the context in which they occur, e.g., how each event relates to another event for a given identifier, such as IP address. Once the log messages are labeled, they are matched with the processed network traffic (CSV) from the previous section, see in Figure 5. Thereby labeling the network packet payloads from the processed network traffic using the new labels created from the log messages. The output of this section forms the basis for the creation of new labels in the labeling section.

The first step of the log message processing is the reuse of the second part of the network traffic processing; see Figure 7. The Graylog CSV files [42] are ingested and processed so that data types are consistent between columns, any timezone differences are removed, and the timestamp formatting is changed from ISO format to Pandas datetime. The only difference between the two applications of the function is the native timestamp format in each data source. The network traffic files use Unix time, and the CSV files produced by Graylog use ISO time.

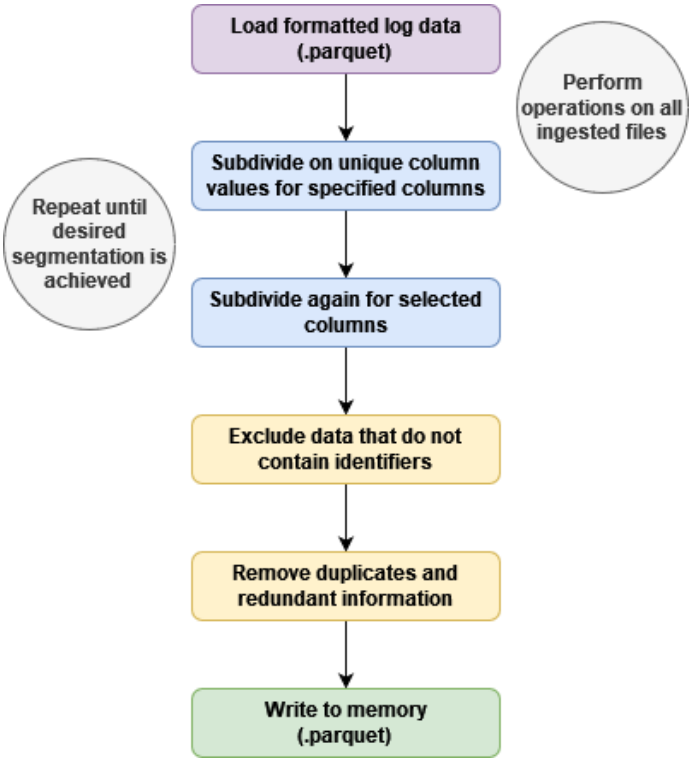

Fig. 8. Log message processing overview

After formatting the data, it undergoes an additional subdivision in combination with aggregation. The purpose of this is to collect similar data points scattered across multiple Graylog excerpts with varying schema, and to organize the collected data points into separate files. Each output file can be thought of as a type of event describing a security-related happening in the system. Moreover, each event has the additional property that the contents of log messages are similar within the event and dissimilar to other events. An overview of the process is illustrated in Figure 8.

The subdivision works by splitting the data into separate files with one file per each unique value in a column for a group of selected columns. This subdivision can be thought of as a form of recursion with a qualitative stopping criterion. When dividing the data no longer increases the similarity of log-message contents within a given split, the recursion stops. In Figure 9 this process is visualized as a tree structure with nodes, in purple, representing splitting points, and leaves, in green, representing events. This process is applied to all ingested Graylog excerpts, such that all rows with identical splitting sequences are written to the same output file. The events, i.e., output files, are named after the splitting sequences, that is; the path traversed through the tree to get from the root to a given leaf. When a node is traversed, the name of that node is added to the event name.

An example of this subdivision and naming convection can be seen in Figure 10, which depicts some of the events that can be generated from splitting on unique values in the column ”severity” and then splitting again for each of the unique values in the column ”name”.

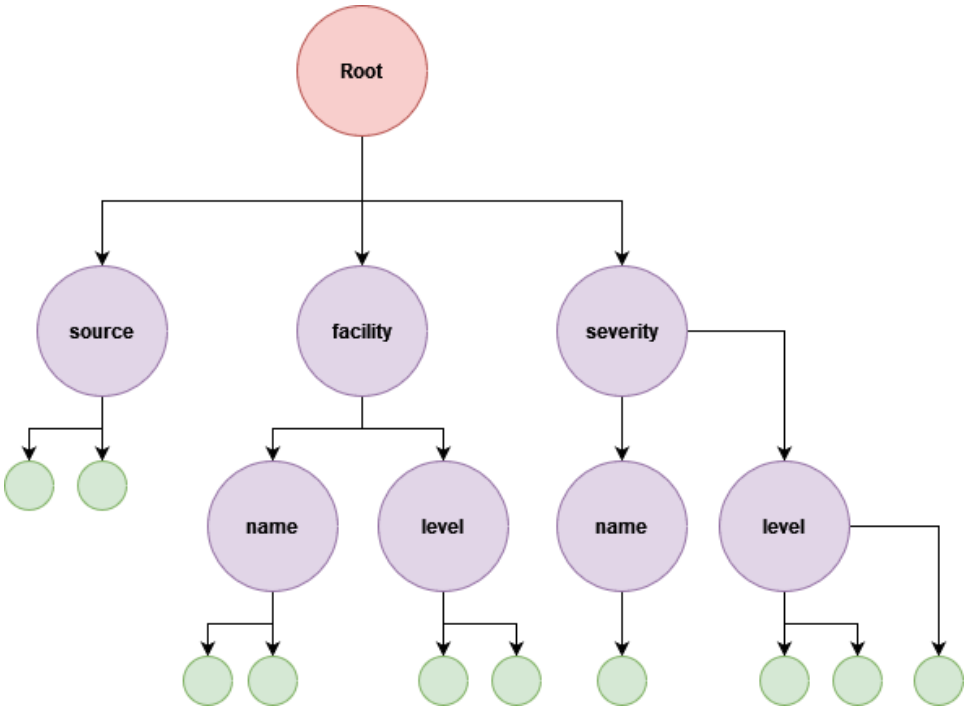

Fig. 9. The subdivision in the creation of events can be visualized as a tree-structure

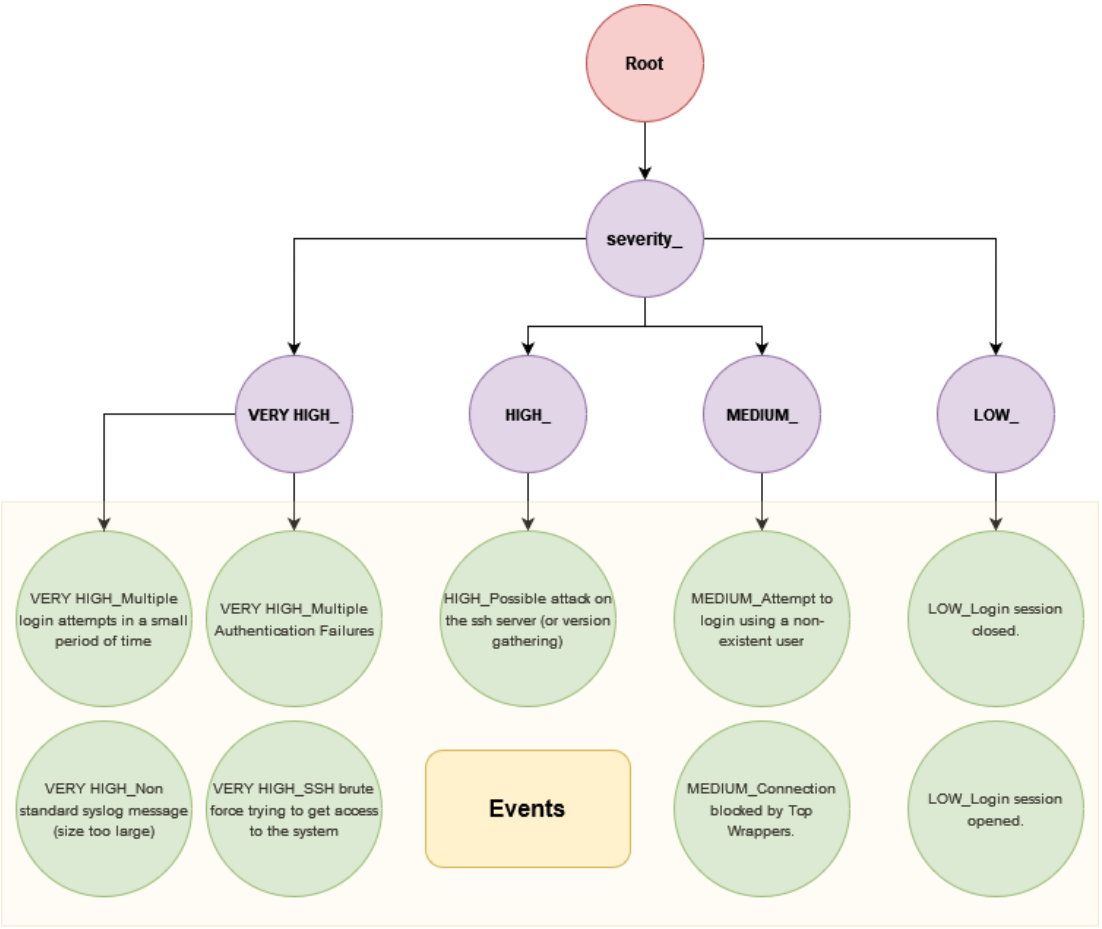

Fig. 10. Example of naming convention during creation of events

After dividing the data into separable events, the log-message contents in each event are parsed for identifiers, that is, IP, Port, etc. Any event that does not have parsable identifiers is excluded from the dataset. The reason being that the labeling process uses these identifiers to associate combinations of events to a given IP address, thereby determining the entry as either malicious or benign. Thus, events without identifiers do not offer any utility for the labeling process at this stage. Lastly, any duplicate entries or entries with redundant information are removed. In total, the log-message processing results in 46 different events that are distinguishable from one another.

4) Labeling: In this section, packet payloads from the processed network traffic are labeled. The process follows the steps described in Figure 11. The first part is unique to the ESS-DMSC dataset and involves the creation of new labels using the generated set of events obtained from processing the Graylog excerpts. The second part involves matching the labels to the processed network traffic and, thereby, assigning labels to the associated network packet payloads. This step applies to both datasets but is more comprehensive for the ESS-DMSC dataset due to the fact that labels do not have a predetermined fit. In comparison, each unique combination of Source IP, Destination IP, Source Port, Destination Port and timestamp in the CIC-IDS-2017 dataset will have predetermined fit between payload and label. Thus, the labeling process for CIC-IDS-2017 involves copying the labels from the CSV files to the processed network traffic files based on the unique combinations of the following identifiers; Source IP, Destination IP, Source Port, Destination Port, and timestamp.

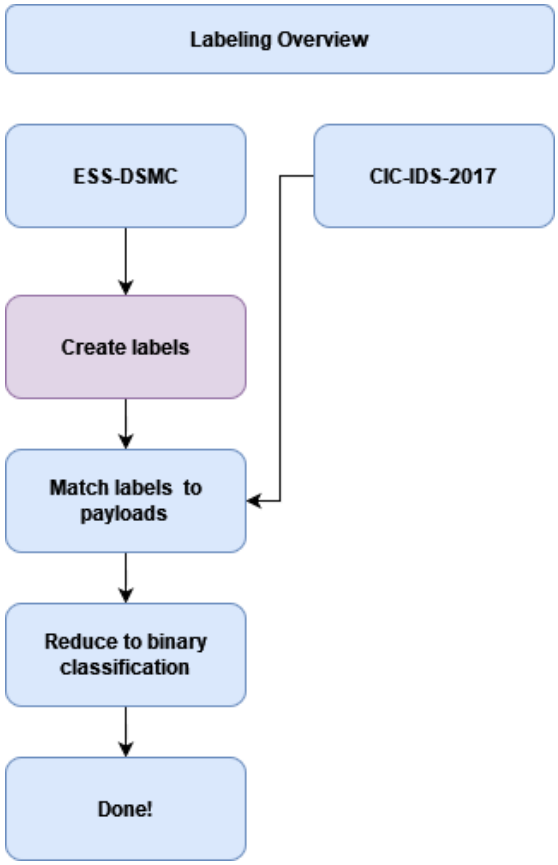

Fig. 11. Overview of the entire labeling process

For the ESS-DMSC dataset labels have to be created as there are none by default. This process follows the steps in Figure 12. First, the Graylog excerpts are processed to create events. This is done in the log-message processing section. In total, 46 unique events are created. Each event corresponds to a security-related happening in the system.

The events are then organized into different groups based on the commonalities between events. For example, Figure 13 shows an event grouping related to SSH activity, e.g., login attempts and failed login clusters, etc. The primary reason for grouping similar events is easier analysis as groups provide context to, and initial segmentation of security-related activity patterns. These groups form the basis for the qualitative assignment of the labels later on. In total, three group events are established. These are SSH activity, depicted in Figure 13, Unusual activity, Authorization & access. An overview of the groups and the types of events in each group can be seen in 14. These groupings are not definitive and can be expanded if needed. The motivation for using these particular groupings is that they are easy to contextualize from a qualitative point of view.

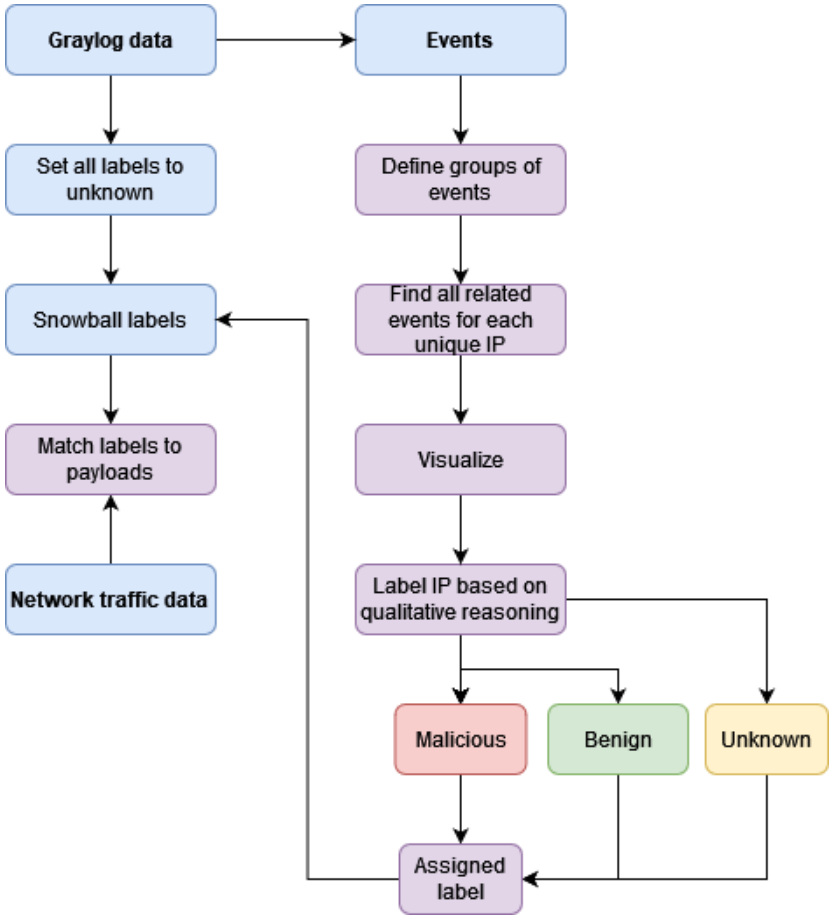

Fig. 12. Overview of the labeling process for ESS-DSMC dataset

In the next step, see Figure 12, unique IP address values are extracted from the subset of events in each group. Thus, subsets of IP addresses corresponding to each of the groups are created. For each unique IP in each of these subsets, a graph representation is built. Events are represented as nodes, and edges are created between events and the given IP address if the given IP occurs in the particular event. The weight of the edge is equal to the number of times the given IP address occurs in a particular event. An example of this graph representation can be seen in Figure 15. Using this graph representation, labels can be assigned to the given IP address by qualitative assessment. For example, the IP address depicted in Figure 15 has been spotted once in a warning message and three times in informational messages, as well as twice in events related to failed login attempts. Thus, this IP address is labeled malicious due to a combination of failed login attempts and a security-related warning. This assessment is based on qualitative reasoning guided by the domain knowledge of the collaborating stakeholder. The entire process is manual, using examples verified by a stakeholder to establish an outline of how entries are to be evaluated. The number of manually labeled entries is roughly 100. This approach is assumptive, but it serves to establish a plausible set of labels where there otherwise are none.

The assigned labels are either Malicious, Benign or Unknown. After assigning a label to an IP address, the label is copied to all other entries that contain the same IP address. Hence, the label is propagated, or snowballed, across the dataset. The reason for this approach is that the number of entries is too large for each occurrence to be labeled manually. It also relies on the assumption that if an IP address is determined to be malicious, then all other entries with the same IP address are also likely to be malicious. Moreover, due to the scarcity of malicious data points, there is a bias towards labeling data points as malicious even if only potentially malicious.

After the labels have been assigned to the Graylog excerpts, see Figure 11, they are matched with the processed network traffic using a fuzzy time window approach. This is due to the fact that ”Source” and ”Destination” for IP and Port are unspecified in the Graylog excerpts. Meaning extracted values for IP and Port can fit any permutation of the identifiers, i.e., Source IP, Source Port, Destination IP, Destination Port and timestamp, present in the processed network traffic files. This means that a definitive match between payload and label is not guaranteed. Therefore, the matching process uses a fuzzy approach that performs the matching iteratively by trying different permutations of IP and Port with varying time windows

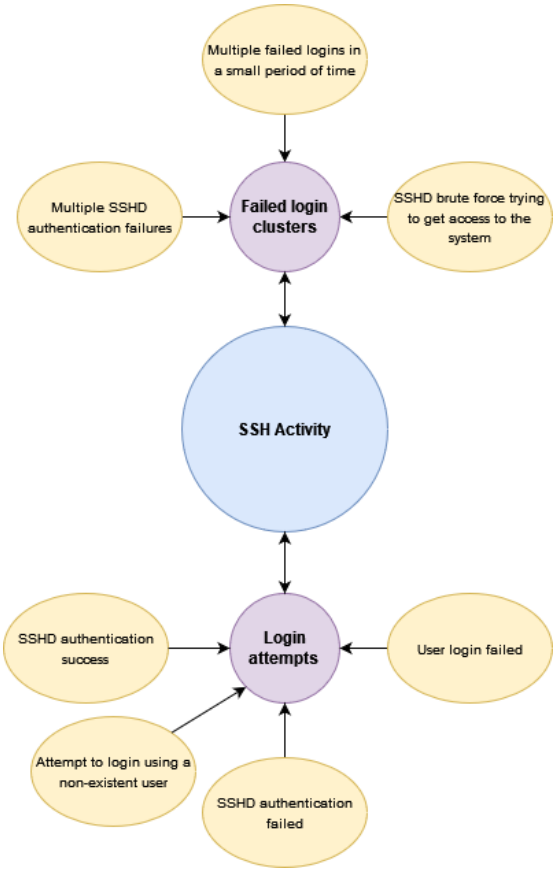

Fig. 13. Example of an event grouping related to SSH activity.

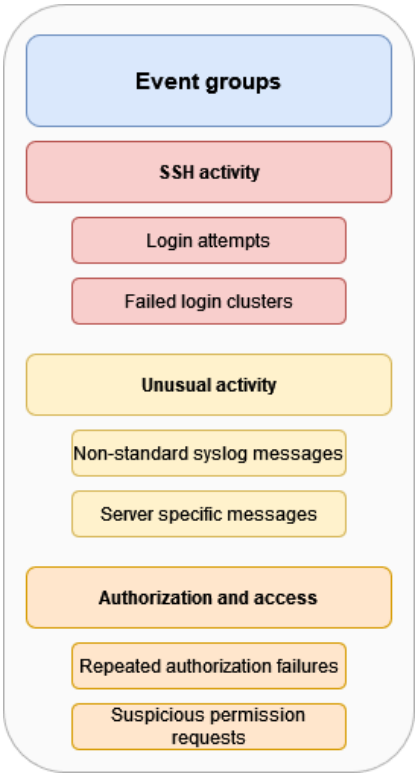

Fig. 14. Overview of event groups and the types of events in each group

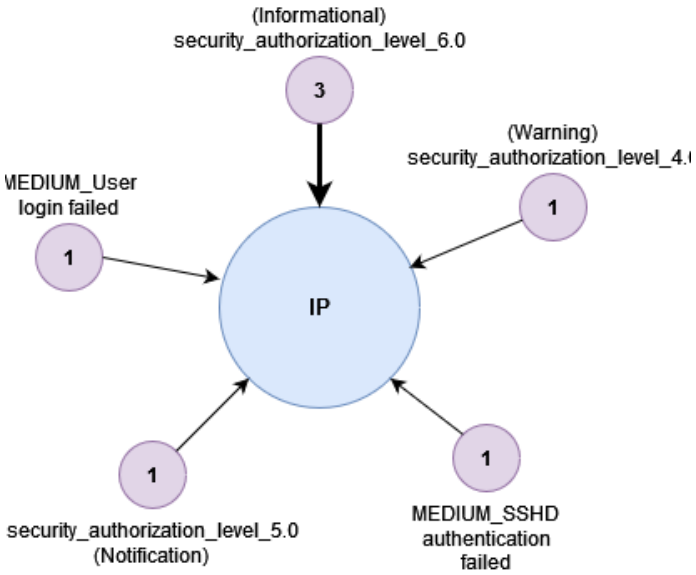

Fig. 15. Example of graph representation of IP address activity

depending on the number of matches. The process can be thought of as search for a likely match within the defined time window. Once the labels are matched with the network traffic, the labeling process is complete.

5) Data Cleaning: The last part of the data processing involves data cleaning [44], see 5. This section involves handling missing values, removal of duplicate values, and balancing of observations. For the ESS-DMSC dataset all rows with the label ”Unknown” are removed from the dataset. These labels can be thought of as a buffer that can be included in the benign category for additional testing purposes. These labels are ultimately not included in the final training, as they tend to reduce the separability between the two classes. Continuing, all duplicate rows are removed and any columns missing over 90% of the values are also removed. Thereafter, all labeled entries without any associated packet payloads are removed from both datasets. Lastly, observations are balanced using majority class undersampling, e.g., samples are drawn from the majority class such that the number of observations become equal to the minority class. The reason for balancing the observations is to prevent bias when training the machine learning models. Although majority class undersampling is efficient in achieving its goal, it has the additional drawback of reducing the total number of observations.

6) Model Selection and Implementation: The selection and implementation of the models is based on the literature review and their suitability for anomaly-based intrusion detection. Using the CIC-IDS-2017 dataset as one of the two datasets for the implementation and training of the models, it allows direct comparison to results found in existing research.

The following four unsupervised machine learning models are implemented: An Autoencoder for its capability to handle complex, high-dimensional, large-scale datasets [46] and imbalanced datasets often found in network traffic, a Convolutional Autoencoder with its ability to capture spatial patterns in the data through convolutional operations, Isolation Forest and a One-Class Support Vector Machine for being commonly used for anomaly-based intrusion detection and showing consistently good results [47] [10].

Two supervised machine learning methods are also implemented to take advantage of the already labeled CIC dataset and to more directly evaluate the proposed labeling approach for the ESS-DMSC dataset. The first being a Convolutional Neural Network for its feature extraction through convolutional operations and overall strong classification performance, inspired by the good performance of the Packet-Based CNN [36]. In addition, the more recent, highly efficient, and well-performing classification model XGBoost was found to perform well when used for intrusion detection [37]. By covering both unsupervised and supervised machine learning methods, it allows comparisons to be made in their suitability for intrusion detection on the two datasets. Such as ease of implementation, their necessity of data preprocessing, and their resulting performance metrics in anomaly-based detection.

The steps of how the machine learning model is applied to the network packet are visualized in Figure 16. The payload is extracted from the packet, converted into normalized integers, and then fed to the model. The output of the model is a prediction of benign or malicious for the packet.

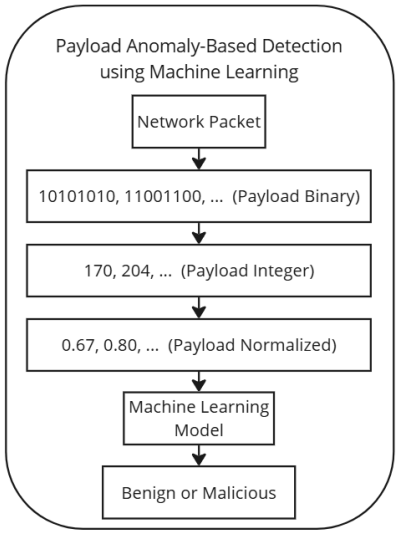

Fig. 16. Steps for how the machine learning models are applied to a network packet's payload.

7) Autoencoder: The Autoencoder is implemented as a reconstruction model where the intention is to reconstruct the input payload data with as low error as possible on the data it is trained on. Training the model to reconstruct benign payloads with as low error as possible will make the model reconstruct payloads that are not similar to benign, such as malicious payloads, with higher error. This results in a difference in error between benign and malicious payloads, which can be used to set a threshold value where the error for every payload above a certain value is predicted as anomaly or malicious. The calculation of the error threshold for the Autoencoder is described in Section IV-E1 as part of the model evaluation. An example of a reconstruction model applied to the payload data of three packets is illustrated in Figure 17, where packet #2 has been predicted as malicious because its reconstruction error is above the set error threshold.

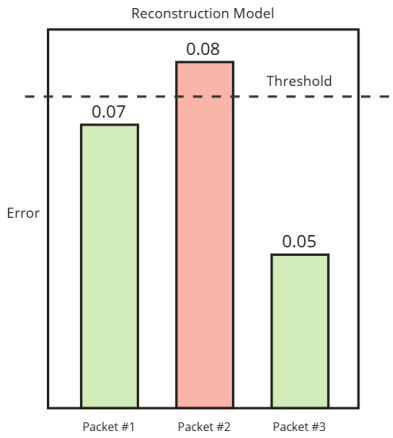

Fig. 17. Reconstruction model with a error threshold for prediction of class

The architecture of an Autoencoder consists of two parts, an encoder and a decoder. The encoding part can be composed of multiple layers, where the first layer has the same number of neurons as the number of values of the input data and every subsequent layer has fewer neurons. By feeding data through the encoding part, the data are compressed to the smallest encoding dimension. The decoder then works in the opposite direction, where it decodes or decompresses the encoder output back to the original size. Therefore, the Autoencoder is trained by setting the input and output to the same instance to minimize the error in reconstruction. An Autoencoder with fewer layers and a smaller latent dimension size generally results in a higher error in reconstruction.

The optimizer used for the Autoencoder is Adam (Adaptive Moment Estimation), which is an extension of the common Stochastic Gradient Descent used for adjusting the weights of the network. Adam offers the benefit of an adaptive computation

of the learning rate informed by other parameters of the model. For the Autoencoder, Adam is initialized with the following parameters:

- A learning rate of 0.001, 0.01 and 0.1 are tried. This rate sets the size of the adjustments to the weights are during computation of gradient descent.
- The betas of (0.9, 0.999), which are used to compute the averages of the square of the gradient. The two numbers represent the exponential decay rates for the estimates.
- An epsilon of 1e-08, which is a small number added to prevent division by zero when calculating the gradients.
- A weight decay of variable number, which is a regularization technique by subtraction of a multiple of a value from the weights in order to combat model overfitting.
- Amsgrad is disabled, which is a technique to speed up convergence of the training by using the maximum of past calculated gradients instead of average to update weights.

Mean Squared Error (Equation 3) and L1Loss (Equation 4) are tested as loss functions, where y represents the actual value, $\hat{y}$ represents the predicted value and n is the total number of samples.

$$MSELoss(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3}$$

$$L1Loss(y, \hat{y}) = \sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4}$$

The activation function for all layers except the last one of the decoder uses ReLu, LeakyReLU or ELU, and the last layer use Sigmoid.

ReLU (Equation 5) returns x or 0, whichever is larger, where x is the input value.

$$ReLu(x) = \max(0, x) \tag{5}$$

LeakyReLU (Equation 6) is a variant of ReLU that introduces a small slope to keep the updates of the weights alive. x is the input to the function and α is a slope coefficient that can be adjusted depending on the application.

$$LeakyReLU(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{otherwise} \end{cases} \tag{6}$$

ELU (Equation 7) is a loss function that can bring the mean activations closer to zero, which speeds up the learning. x is the input to the function, α is the ELU coefficient which is usually set to 1, and e is the base of the natural logarithm.

$$ELU(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases} \tag{7}$$

Sigmoid (Equation 8) maps the input value x to a value between 0 and 1.

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

L1 or Lasso regularization encourages sparsity in the model by shrinking some coefficients to zero. This allows the model to do feature selection and ignore some features altogether. This technique works particularly well when the feature count is very high.

L2 or Ridge regularization encourages coefficients to be small but not zero, meaning all features are included, but their effects are minimized, and this technique combats overfitting.

All hyper parameters and their tested values for the Autoencoder can be seen in Table II. Here, num_layers is the number of layers in the encoder and decoder. The number of neurons in each layer is calculated with Equation 9.

$$\frac{Sequence\_Length * Feature\_Count}{2^{(Layer\_Index)}} \tag{9}$$

Where in the encoder, the first layer is the same input size and each subsequent layer is half the size until 0 or the number of layer count is reached. For the decoder, the layer sizes are mirrored of the encoder, where it starts with the smallest size and doubles in size until the input size is reached again.

The dropout_rate sets the number of neurons that will have their neurons zeroed when updating the weights and mitigates overfitting. The activation function, loss function, and learning rate have been explained above. L1_lambda sets the value for L1 regularization, a higher value leads to more features being ignored and a simpler model, however, too high of a value may cause underfitting. L2_lambda sets the value for L2 regularization, higher values tend to reduce overfitting, but too high value will lead to underfitting.

Parameter	Values
num_layers	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
dropout_rate	0, 0.1, 0.2, 0.3, 0.4, 0.5
activation_function	ReLU, LeakyReLU, ELU
loss_function	MSELoss, L1Loss
learning_rate	0.001, 0.01, 0.1
l1_lambda	0, 0.01, 0.1, 1, 10
l2_lambda	0, 0.01, 0.1, 1, 10

TABLE II
AUTOENCODER HYPERPARAMETERS

8) Convolutional Neural Network: A Convolution Neural Network (CNN) uses convolutional operations instead of multiplications in some or all layers when feeding forward the input data. The convolutions takes patches of input data and applies filters to them to compute the output; these filters are moved across the input data to create the new set of values [48]. Filter values are adjusted during the training process, which allows the network to automatically extract the features that are most important to the problem. Depending on the representation of the data and the dimensions of the filters, it can capture spatial and temporal patterns in the data. The CNN architecture consists of a variable number of convolutional layers, each with a batch normalization layer, a pooling layer, and a dropout layer. The final output of these layers are fed into a fully connected layer, followed by another fully connected layer with a single output neuron. This neuron is processed by Sigmoid (Equation 8) to map the value between 0 and 1 for binary classification, where values above 0.5 are considered 1.0 and values below are considered 0.

Adam is also used as the optimizer for the CNN, with its parameters initialized to the same values as for the Autoencoder.

The loss functions used for training the CNN are either Cross Entropy Loss or Binary Cross Entropy Loss, both which are commonly used for classification problems. Cross Entropy Loss (Equation 10) combines the Logarithmic Softmax and Negative Likelihood Loss function into a single loss function for multiclass problems. Where y_{ij} is the true value, $\hat{y}_{ij}$ is the predicted value, N is the number of samples in the dataset, C are the classes in the dataset, the logarithm is the natural logarithm.

$$CrossEntropyLoss(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \cdot log(\hat{y}_{ij}) \tag{10}$$

Binary Cross Entropy Loss (Equation 11) is a loss function for binary classification problems where y_i is the target, $\hat{y}_i$ is the predicted value, n is the number of samples in the dataset and the logarithm is the natural logarithm.

$$BCELoss(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)] \tag{11}$$

The same activation function is used on all but the last layer in the CNN, either ReLu (Equation 5), Sigmoid (Equation 8), or Tanh (Equation 12). The last layer uses Sigmoid to map the value between 0 and 1 for binary classification through a threshold of 0.5.

$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{12}$$

All hyper parameters and their tested values for CNN can be seen in Table III. Where num_conv_layers is the number of convolutional layers and num_filters is their number of channels, their kernel_size, their stride and their padding value, num_neurons_fc is the number of neurons in the first fully connected layer. Latent_dim is the the number of neurons of the latent dimension between the convolutional layers and fully connected layers. The activation function, dropout rate, loss function, learning rate, l1 and l2 lambas are the same as for the Autoencoder.

9) Convolutional Autoencoder: The Convolutional Autoencoder is implemented like the regular Autoencoder as a reconstruction model of the packet payloads, but the fully connected layers in the encoder segment of the Autoencoder are replaced with convolutional layers. The reason being that convolutional operations allow for the extraction of potential spatial patterns in the input and potentially improving the accuracy of the model. The pooling layers are also used instead of the fully connected layers to reduce the dimensionality of the data as it is fed through the network. The architecture of the Convolutional Autoencoder consists of a variable number of convolutional layers in the encoder part, where the number of neurons in the first layer is the number of features in the dataset, and the output is a variable filter size. All subsequent convolutional layers have the same filter, input and output size, ending with a fully connected layer that reduces the output to a variable latent dimension size. All convolutional layers have the same kernel size, stride distance, and padding value. The decoder part consists of a fully connected layer that decompresses the data back to its original input size. Every layer in the encoder and decoder is

Parameter	Values
num_conv_layers	1, 2, 3, 4, 5, 6, 7, 8
num_filters	32, 64, 128
kernel_size	1
stride	1, 2, 3
padding	0
num_neurons_fc_layer	32, 64, 128
latent_dim	8, 16, 32, 64, 128
activation_function	Relu, Sigmoid, Tanh
dropout_rate	0, 0.1, 0.2, 0.3, 0.4, 0.5
loss_function	BCELoss, CrossEntropyLoss
learning_rate	0.001, 0.01, 0.1
l1_lambda	0, 0.01, 0.1, 1, 10
l2_lambda	0, 0.01, 0.1, 1, 10

TABLE III
CONVOLUTIONAL NEURAL NETWORK HYPERPARAMETERS

also followed by a dropout layer to reduce overfitting. The dropout value for every layer is the same and is exposed as a hyperparameter for testing.

Adam is used as the optimizer with its parameters initialized to the same values as the regular Autoencoder and the CNN. The Convolutional Autoencoder is tested with the same loss functions, MSELoss and L1Loss, as the regular Autoencoder. Based on their common ultimate goal of reconstructing the input with as low error as possible.

Regarding the activation functions, the Convolutional Autoencoder is tested with one function in common with the regular Autoencoder, ReLU, as this is a common and well-performing activation function for reconstruction models. In addition, two other tested functions are tested, Sigmoid and Tanh. These are in common with the CNN as these are more suitable for the output of convolutional layers.

All hyper parameters and their tested values for the Convolutional Autoencoder can be seen in Table IV. All hyperparameters have the same description as for the CNN.

Parameter	Values
num_conv_layers	1, 2, 3
num_filters	16, 32, 64, 128
kernel_size	1
stride	1, 2
padding	0
latent_dim	8, 16, 32, 64, 128
activation_function	Relu, Sigmoid, Tanh
dropout_rate	0, 0.1, 0.2, 0.3, 0.4, 0.5
loss_function	MSELoss, L1Loss
learning_rate	0.001, 0.01, 0.1
l1_lambda	0, 0.01, 0.1, 1, 10
l2_lambda	0, 0.01, 0.1, 1, 10

TABLE IV
CONVOLUTIONAL AUTOENCODER HYPERPARAMETERS

10) Isolation Forest: The purpose of Isolation Forest is to isolate outliers, which is why it often performs well in anomaly detection tasks such as intrusion detection. It does this by creating an ensemble of isolation trees by recursive random partitioning. Anomalies are points that are easier to separate from the rest of the sample, which means that they are few and clearly different from what is considered normal.

All hyperparameters and their tested values for the Isolation Forest can be seen in Table V. Where n_estimators is the number of base estimations in the ensemble. Max_samples is the proportion of samples to draw from the training set to train each base estimator. Contamination is the proportion of outliers that should be included in training. Max_features is the number of features to use when training. Finally, bootstrap, when set to true, allows individual trees to fit to random subsets of the training data.

11) One-Class Support Vector Machine: As an variant of Support Vector Machine (SVM) primarily designed for outlier or anomaly detection, its goal is to identify points for which deviate significantly from the norm. It does this by creating a boundary around the majority class in the feature space and then maximizes the margin around this class, leading to a strong separation for any potential anomalous data points. Since it only uses the majority class during training, it has an inherent handling of imbalanced datasets.

All hyperparameters and their tested values for the One-Class SVM can be seen in Table VI. Where nu is the upper bound on the fraction of training errors and lower bound of the fraction on support vectors. This hyperparameter controls the model's

Parameter	Values
n_estimators	100, 200, 500
max_samples	'auto', 0.5, 0.7, 1.0
contamination	'auto', 0.1, 0.2, 0.3
max_features	1, 2, 3, 4
bootstrap	True, False

TABLE V
ISOLATION FOREST HYPERPARAMETERS

sensitivity to outliers. The "kernel" specifies which function to be used in the in algorithm. Gamma is the coefficient for the kernel function. Finally, "shrinking" when set to true is a heuristic technique that speeds up the optimization problem by removing certain constraints (support vectors).

Parameter	Values
nu	0.1, 0.5, 0.7
kernel	'linear', 'poly', 'rbf', 'sigmoid'
gamma	'scale', 'auto', 0.1, 1, 10
shrinking	True, False

TABLE VI
SUPPORT VECTOR MACHINE HYPERPARAMETERS

12) XGBoost: eXtreme Gradient Boosting, or XGBoost, is a Gradient Boosting framework that utilizes Regularization to prevent overfitting. It constructs an ensemble of decision tree models in which each model tries to correct the mistakes of the previous model based on the gradient of the loss function [37]. The output is then combined as a weighted sum to generate predictions. The framework is both robust and efficient, as it can handle a large number of features and missing values in a computationally efficient way. It is also scalable, as it can be integrated on top of distributed environments.

All hyperparameters and their tested values for XGBoost can be seen in Table VII. Here, max_depth determines how deep a tree can be, and increasing this value will make the model more complex and more prone to overfitting. Learning_rate sets the step size of shrinking in update. n_estimators is the number of gradient boosted trees or, in other words, the number of boosting rounds. Gamma is the minimum loss reduction needed to make a further split of a leaf node in the tree. A larger gamma leads to a more conservative model. Min_child_weight is the minimum sum of instance weight, meaning if a tree partition steps in a leaf node with a sum less than this value, the partition will stop. Subsample is the ratio of training instances, 0.5 means that half of all data will be sampled before starting to build the tree. Which will prevent overfitting. Finally, colsample_bytree is the subsample ratio of columns when constructing each tree.

Parameter	Values
max_depth	3, 5, 7, 10
learning_rate	0.01, 0.1, 0.2
n_estimators	100, 200, 500
gamma	0, 0.1, 0.2
min_child_weight	1, 5, 10
subsample	0.5, 0.7, 1.0
colsample_bytree	0.5, 0.7, 1.0

TABLE VII
XGBOOST HYPERPARAMETERS

D. Demonstration

The demonstration of the artifacts is done by applying the new label method to the ESS-DMSC data and using the then labeled dataset and also the CIC-IDS-2017 dataset to train models for each of the six machine learning algorithms.

Model training is done experimentally in order to answer the research question about model performance and to confirm or deny the related hypotheses. As part of the experiments, different model architectures and hyperparameters are tested to find the best-performing combination to produce the best possible results.

1) Batching: To manage memory constraints, reduce computational requirements when training, and for the model to converge faster, the CIC-IDS-2017 dataset is divided into batches of 256 instances, and the ESS-DMSC dataset is divided into batches of 8 instances because it is smaller. Each model is trained for enough epochs on all batches for the training and validation loss to stabilize and see no further improvement. The model at which the validation loss is at its lowest is saved for testing as this is the model that performs best to the validation set and has the lowest amount of overfitting to the training set.

2) Model Input format: The data are structured in 3-D tensors of (batch size, sequence length, feature count). The sequence length is set to 1 for packet-based anomaly detection, but allows for longer sequences for session-based models. The feature count represents the number of converted bytes included from the packet payload, and different feature counts are tested to find the best performing. This is done out of an effectiveness point, where fewer features is more effective computationally, but also to gauge if the amount of payload used has an effect on the generalizability of the model. As the payloads are of variable size, the features are either truncated or padded with 0 values to be of the same length when fed to the model as input.

3) Majority class undersampling: In order for supervised learning methods to correctly classify every class, the dataset must have a balanced class representation. Otherwise, the model is often trained to accurately classify the majority class and perform poorly on any minority classes. Which, in the case of intrusion detection, is particularly detrimental, as the minority class is the most important class. As both the CIC-IDS-2017 and ESS-DMSC datasets are imbalanced to a certain degree, undersampling of the majority class is performed in order to achieve an evenly balanced dataset for model training and testing.

4) Train, Val, Test split: The dataset is split into three sets of 70% training, 15% validation, and 15% testing, to mitigate the overfitting of the model and to reliably evaluate the performance of the model's ability to generalize to unseen data from the same dataset.

5) K-Fold Cross Validation: In order to find the best combination of hyperparameters, K-Fold Cross Validation with 5 folds is used on each combination. K-Fold Cross Validation involves the splitting of the dataset into K folds, where K-1 folds are used to train the model and the remaining fold is used for validation. The permutation of folds is then rotated so that each fold is used for validation once. This allows for the validation of the models to take all parts of the data into account and avoid situations where a model may overfit or underfit to certain parts of the data. For the same reasons, it also ensures a more robust selection of the optimal hyperparameters.

6) Hyperparameter Search Methods: Grid search is when every possible combination of hyperparameters is trained and tested for a model, which can be computationally expensive when the number of hyperparameters is many. Therefore, random search is an alternative where a random subsample of hyperparameter combinations is tested in the hope of finding a well-performing combination. It is then possible to use the explored combinations in order to further inform which other combinations should be explored further. Meaning that if certain values for hyperparameters are producing worse results, they can be removed in future random search combinations.

E. Evaluation

To evaluate the artifacts the models are tested on the unseen test set and performance metrics are calculated based on the models predictions compared the ground truth.

The following evaluation metrics are selected so that they can explain how effective the models are in anomaly-based detection. Since the problem is binary classification or prediction, the predictions of the models can be divided into the following four categories:

- True Positive (TP): Correctly predicted positive values.
- True Negative (TN): Correctly predicted negative values.
- False Positive (FP): The true class is negative, but the predicted class is positive.
- False Negative (FN): The true class is positive, but the predicted class is positive.

With these categorizations of predictions, the following metrics can be calculated to evaluate the performance of the models on the unseen testing set. A value of 1.0 is considered ideal:

- Accuracy (Equation 13) which is the ratio of correct predictions to the total number of predictions.
- Precision (Equation 14) which is the ratio of correctly predicted positive observation to the total predicted positive observation. It gives a measure of False Positive Rate.
- Recall (Equation 15) which is the ratio of correctly predicted positive observations to all observations in actual classes. It gives a measure of False Negative Rate.
- F1 Score (Equation 16) which is a weighted average of Precision and Recall.
- AUC-ROC: The area under the receiving operating characteristic curve is a measurement of the performance of classification problems at various threshold settings. It shows how well the model can separate positive and negative instances.
- AUC-PRC: The area under the Precision-Recall curve is a measurement of the performance of the model's trade-off between precision and recall of the true positive class.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{13}$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{14}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{15}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{16}$$

Accuracy is a way to get an general idea of the performance of a model, but it has downsides when working with unbalanced datasets. This is often the case of network traffic, where benign or normal traffic often consists of over 90% of the data. In these cases, the accuracy of the model can be very high, while still not accurately identifying the minority class. While the datasets are rebalanced in this work, metrics such as Recall and Precision are of greater interest, since they are more descriptive of performance in relation to the different classes. For intrusion detection, the goal is usually to achieve high Recall or, in other words, to reduce the amount of false negatives, since the potential harmful effects of missing intrusions may be severe. However, achieving high precision or reducing the number of false positives is still important, since falsely predicting benign behavior as malicious may be disruptive if the predictions are used to lock users out of the system. Therefore, ideally, a high F1 score is the best, where the false negatives and false positives are both minimized, but high Recall is preferred over high Precision if both cannot be achieved.

1) Autoencoder & Convolutional Autoencoder: In order for the Autoencoder and the Convolutional Autoencoder to predict packet payloads as anomalies, a threshold for reconstruction error must be calculated. Any packet error above the threshold would be predicted to be malicious. Multiple thresholds are calculated on the basis of the observed reconstruction error on benign data in the training set in order to find which one performs the best in terms of accuracy, precision, recall and F1 score.

Four different thresholds are defined based on the 90th, 95th, 99th and 99.9th percentile in reconstruction error on benign data. Ideally, the separation in reconstruction error when testing both benign and malicious packets is that distinct the 99.9th percentile can be used for near perfect anomaly detection. However, in cases where there is too high a similarity in the reconstruction error between the classes, the lower percentiles may still predict all malicious packets as anomalies, while sacrificing overall accuracy and introducing false positives. Therefore, it is worth experimenting with different thresholds in order to find the one with the best accuracy, precision, recall, and f1 score.

The Autoencoder and Conv. Autoencoder are trained on strictly benign data, but tested on both benign and malicious data in order to accurately determine performance of the model for both classes. The Autoencoder is trained using random search on the hyperparameters found in Table II, and Conv. Autoencoder on hyperparameters in Table IV.

2) Convolutional Neural Network & XGBoost: As CNN and XGBoost are supervised learning methods, they are trained and tested on the balanced version of the dataset containing equal amounts of benign and malicious instances. The input when training the model is the payload and the target is the label for the payload. The validation set is used as an early stopping criterion, where if the validation loss does not improve over 10 epochs, the training is terminated and the model is saved. The hyperparameters explored for CNN can be seen in Table III, and for XGBoost in Table VII.

3) Isolation Forest & One-Class Support Vector Machine: Isolation Forest and One-Class SVM are trained on strictly benign data and then tested on both benign and malicious data. They are unsupervised learning methods, and unlike the Autoencoder, they do not have a target when training. The stopping criterion for the training is when no further improvement in convergence is observed. The hyperparameters explored for the isolation forest can be see in Table V, and for One-Class SVM in Table VI.

4) Inter-dataset generalizabilty: First the trained models are tested on the test set from the same data set on which they were trained, to verify the implementation of the model and to find the best performing model. Then the best performing models as determined by the highest F1 Score, are tested on the dataset they were not trained in order to measure the inter-dataset generalizability. A visualization of the four model train and test combinations can be seen in Figure 18.

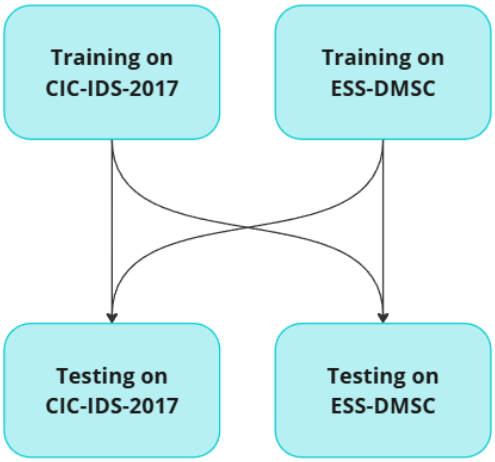

Fig. 18. Model Training and Testing Combinations

F. Communication

The communication of the knowledge gained from the research performed in this work is realized through the writing of this thesis, the oral presentation of the results, the discussion with the external stakeholder, and the submission of a condensed version of the thesis to workshops and journals.

The thesis is mainly written for the target audience of computer scientists or data scientists with knowledge in the field of machine learning, since a large part of the thesis consists of technical descriptions of machine learning model implementations. However, the discussion section of the thesis is aimed at a higher level, which means that it can also be of benefit to cybersecurity professionals or computer scientists without experience in machine learning.

This thesis is also rewritten and condensed into a 5-page version to be submitted to the following Internet of Things or Cybersecurity conferences and journals. In this version of the thesis, the content has been adjusted to be less of a technical description of machine learning implementations and instead focuses on the analysis of the results, the discussion, and more actionable insights.

- The 14th International Conference on the Internet of Things (iot-conference.org)
- Journal Home:: Internet of Things and Cloud Computing:: Science Publishing Group (jiotcc.com)
- Journal of Cybersecurity — Oxford Academic (academic.oup.com/cybersecurity)

G. Ethical Considerations

Working with network traffic data and system log messages raises questions about privacy and data protection. To ensure, privacy of users and compliance with data protection regulations, sensitive information is removed by ESS-DMSC before sharing the data for research purposes. The exact steps taken by ESS-DMSC are not explicitly disclosed, but the collaborating stakeholder has confirmed that the particular subset of data used in this work does not raise privacy concerns. Whilst the data contains intact log message strings, the exact contents of log messages, nor any information extracted from log messages, e.g., IP addresses, Ports, Domains, Hosts, Users, etc, are not disclosed in this work. Only high-level descriptions, for example, names, of log-message contents are provided. Should there be a risk for specifics, the values are anonymized before inclusion. Thus, this work should, for all intents and purposes, comply with any security and privacy related concerns.

H. Limitations

This research is subject to multiple limitations that pose potential threats to the validity of the results presented. Despite these limitations, this research offers valuable insight into the potential of machine learning for anomaly-based intrusion detection. By acknowledging these limitations, this work strives to be transparent and scientifically rigorous.

The effectiveness of machine learning models can be impacted by the size and quality of the data. Although this research uses two data sets, one from a public repository (CIC-IDS-2017) and one from an external stakeholder (ESS-DMSC), the generalizability of findings may be limited partly due to the size of the data, but primarily due to both datasets being interpretations of real-world network traffic. The ESS-DMSC dataset consists of real-world traffic, but is limited by the specific segment shared by the collaborator and the necessary interpretations needed to transform the raw data to a format that is usable as input for machine learning models. Moreover, the problem is reduced to binary classification, which is a simplification of the real world, such that the two datasets can be made comparable. This reduction is also justified by the

complexity of multi-class classification in cybersecurity, especially when working with unbalanced data [41]. However, due to the simplification the models presented in this thesis may not perform as well against attack vectors present in production systems, due to perpetrators employing a wide variety of attacks and techniques [3].

Hyperparameters are a set of values that govern the behavior of training algorithms in machine learning models. Therefore, the models are sensitive to the chosen set of hyperparameters. This research uses Random Search to find the optimal set of values for the parameter as Grid Search is this case is considered too computationally expensive, but there is always a possibility that better parameters exist. Finding the best set of hyperparameters is a search problem, and thus the perceived optimal can, in fact, be an approximation of the optimal and not the actual optimal. The model evaluation is based on metrics such as accuracy, precision, recall, F1 score, AUC-ROC, AUC-PRC. These metrics provide valuable performance insights and are used by researchers as a point of reference when interpreting the performance of models. However, these metrics are not definitive and can become inflated or biased in complex models. Thus, there is a risk that performance is interpreted incorrectly from a theoretical point of view.

Additionally, these metrics may not fully capture the effectiveness of the proposed models should they be applied to real-world production networks, because the metrics require contextual interpretation. For instance, false positives and false negatives are from a machine learning point of view different aspects of the same statistic. However, in a real-world setting false negatives, that is, mistaking intrusion attempts as regular traffic, are far more problematic than false positives, that is, mistaking regular traffic as intrusion attempts. Meaning real-world application and interpretation requires in-depth knowledge about the domain and system to which it is applied, and may also require a bespoke solution to the particular system. Thus, the interpretation of metrics in this work is limited by the fact that the proposed models are not used in any production network, as deployment is outside the scope of this research. However, from a theoretical point of view, efforts are made to reduce the number of false negatives as this is in agreement with the desired behavior models when deployed in real-world network systems.

Continuing, the labeling is intended to provide a starting point for comparing the performance of models when trained on artificial research data and data constructed from real-world traffic, since the real-world data is supplied by an external stakeholder. Labeling is limited by restricted insight and access to the networking systems at ESS-DMSC, as well as to the particular segments and number of data points ESS-DMSC is willing to share. Thus, the correctness of labels is limited by the qualitative assumption and reasoning applied when assigning labels. Moreover, the correctness of labels is also affected by necessary strategies such as the propagation of labels, which inadvertently can mislabel entries. However, efforts are made, that is, a discussion with the stakeholder and research into the documentation of the systems, to ensure that the set labels are as correct as possible given the posed limitations.

V. Results & Analysis

This section details and analyzes the results produced by the applied methodology. Statistics and performance metrics are introduced and interpreted. The first part of this section details and analyzes the results of the data processing method described in the methodology, i.e., the statistics of the ESS-DMSC dataset are described and compared with the statistics of the CIC-IDS-2017 dataset. The second part of this section details and analyzes the performance metrics of the selected machine learning models when trained and tested on both datasets for the purpose of detecting anomalous payloads.

A. A Novel Labeling Method

The applied methodology results in the creation of a dataset (ESS-DMSC) and the transformation of an existing dataset (CIC-IDS-2017). After processing the two datasets they are structured as CSV files containing labeled key-value pairs. Key refers to the unique combination of identifiers typically present in network capture files, e.g., Source IP, Destination IP, Source Port, Destination Port and timestamp. Value refers to the transformed numerical representation of the packet payloads. Thus, the two datasets contain the corresponding headers and column values. For a visual comparison between the formatting of the two datasets; see Tables VIII and IX. Labels are assigned by matching unique combinations of key values with corresponding entries in the label files. For CIC-IDS-2017 labels are copied over from the original CSV files as each unique key combination already has a predefined label. This is not the case for the ESS-DMSC dataset where labels are created through a qualitative process, and thereafter matched with the processed network traffic by searching for likely matches. An overview of the ESS-DMSC dataset labeling process is illustrated in Figure 12. In both datasets, the number of classes is reduced to binary, e.g., malicious and benign. For the CIC-IDS-2017 dataset this involves combining the classes FTP-Patator and SSH-Patator into a single malicious class. In the ESS-DMSC dataset malicious entries are not explicitly separated into different categories but can be thought of as a combination of the following groups; SSH Activity, Unusual Activity and Authorization & Access, see Figure 14 for an overview of the groups and Figure 13 for a more detailed depiction of events associated with the SSH Activity group.

Label	Timestamp	Src IP	Src Port	Dst IP	Dst Port	Protocol	Feature_0	Feature_1	Feature_2	Feature N-1	Feature N
BENIGN	2017-04-07 12:32:00 UTC	23.15.4.16	80	192.168.10.14	50518	6	0.5608	0.5373	0.7765	0.2118	0.3725
BENIGN	2017-04-07 12:32:00 UTC	23.15.4.16	80	192.168.10.14	50518	6	0.8784	0.6078	0.0549	0.2	0.6706
BENIGN	2017-04-07 12:32:00 UTC	23.15.4.16	80	192.168.10.14	50518	6	0.4980	0.1804	0.7373	0.4902	0.6314

TABLE VIII
Excerpt from the CIC-IDS-2017 after processing

Label	Timestamp	Src IP	Src Port	Dst IP	Dst Port	Protocol	Feature_0	Feature_1	Feature_2	Feature N-1	Feature N
Malicious	2024-03-18 13:05:20 UTC	XXX.XX.XXX.XXX	XXXX	XXX.XXX.XXX.XXX	XXX	6	0.0824	0.0118	0.0118	0.0	0.1020
Malicious	2024-03-18 13:06:40 UTC	XXX.XX.XXX.XXX	XXXXX	XXX.XXX.XXX.XXX	XXX	6	0.0902	0.0118	0.0118	0.0314	0.3216
Benign	2024-03-18 13:06:50 UTC	XXX.XXX.XXX.XXX	XXXXX	XXX.XXX.XXX.XXX	XXX	6	0.0902	0.0118	0.0118	0.0118	0.9412

TABLE IX
Excerpt from the ESS-DMSC dataset after processing

Figure 19 depicts a statistical overview of the labeling process for the ESS-DMSC dataset. The initial number of data points in the network traffic captures (PCAP) are approximately 628 thousand, with 43% of entries containing a payload, and remaining 53% of entries containing no payload. Thus, less than half of the captured network traffic is usable in the construction of the ESS-DMSC dataset.

In comparison, processing the CIC-IDS-2017 PCAP fields yields approximately 11.5 million packets in total with approximately 5.7 million packets carrying a payload, corresponding to approximately 50% of packets, as illustrated in Figure 20. Thus, the ratio between packets with a payload and packets without a payload is similar between the two datasets. The total number of payloads in the ESS-DMSC dataset is approximately one-tenth of the total number of payloads in the CIC-IDS-2017 dataset.

The number of data points in the aggregated log messages (Graylog excerpts) amounts to roughly 22 thousand. Each of these data points were labeled by first setting the default label to ”unknown” for all entries, and thereafter propagating the assigned label, that is, malicious or benign, to all other entries with same IP address. Starting with the propagation of benign labels and thereafter malicious labels to allow for benign entries to be relabeled as malicious entries, and not the other way around. The reason for giving precedence to malicious entries over benign entries is because of their relative scarcity. Mislabeling benign entries as malicious is acceptable, within reason, if the number of malicious entries can be increased to eliminate the scarcity. Moreover, the assumption is; if the IP address of a given row is labeled as malicious, then all other instances where that IP occurs must also be malicious. This process results in roughly 41% of log messages labeled malicious and 27% benign, the remaining 32% are unknown. Unknown entries are removed before training the models but are kept in the dataset as a potential buffer. Thus, approximately 68% of log messages are used to find a potential match in processed network traffic.

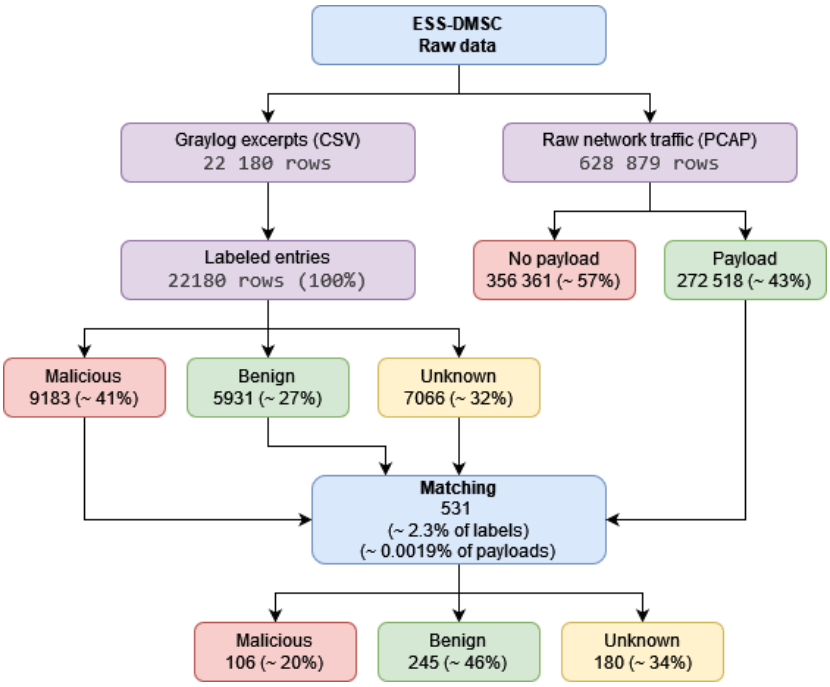

Fig. 19. Overview of how the total number of labels changes during the labeling process

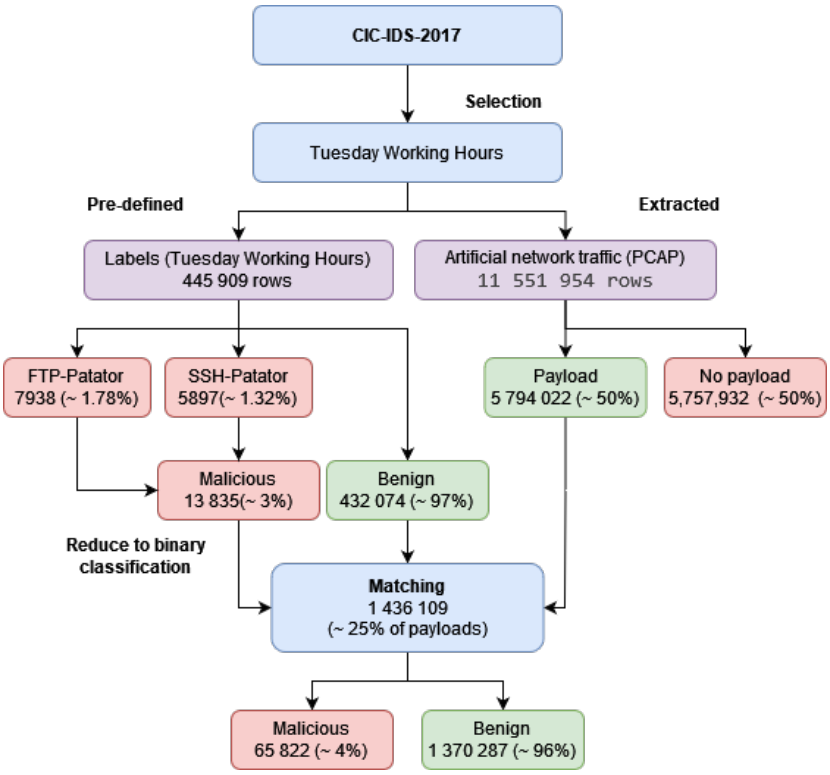

Fig. 20. Overview of the total number of labels in the CIC-IDS-2017 dataset

Matching the assigned labels with the processed network traffic results in 531 labeled packet payloads, 20% of these are labeled malicious, and 46% are labeled benign, the remaining 34% are unknown. Thus, only 2.3% of labeled log entries have a corresponding match in the processed network traffic, despite searching for the closest match within a tolerance of 0.5 seconds in either direction of a given time stamp. This corresponds to approximately 0.0019% of the total number of packets containing payloads. Showing both how rapidly the total number or data points decline at each step of the processing and how the majority of raw network traffic leaves no apparent trace when considering system log messages as potential source of labels.

Compared to the ESS-DMSC dataset, the CIC-IDS-2017 dataset (Tuesday), see Figure VIII, contains about 445 thousand labeled entries, 97% of which are labeled benign and the remaining 3% as malicious. Matching the labeled entries to the artificially generated network traffic results in approximately 1.4 million matches, which corresponds to a match for approximately 25% of payloads. Of the labeled payloads, 96% are labeled as benign and 4 % as malicious. Showing a relatively large drop in quantity despite having pre-defined labels for unique combinations of key-values. These results indicate two things; that large portions of network traffic, even if artificially generated and labeled, consist of payloads that have no apparent label, and that malicious network traffic also consist of malicious payloads that are, according to the labels set in CIC-IDS-2017, separable from benign payloads.

In conclusion, both datasets contain identifiers, payload data represented as numerical features, and binary classification labels after processing. Thus, the formatting, as illustrated in Tables IX and VIII, is comparable. Moreover, the classification categories in the CIC-IDS-2017 dataset before reduction are FTP-Patator and SSH-Patator, see 20. Since the ESS-DMSC dataset uses similar types of groupings, in particular SSH 13, the labeled packet payloads themselves are also corresponding. Thus, the two datasets are comparable from a broader perspective, despite the apparent difference in the number of data points, which is limited by the size of the collected data rather than its contents.

B. Payload Anomaly-Based Detection

The results and analysis of the model performance for payload anomaly detection are presented in the following order: First, the models trained and tested on the CIC-IDS-2017 dataset, followed by the models trained and tested on the newly created ESS-DMSC dataset. The best-performing models from these two tests are then further tested on the dataset they weren't trained on, and the results for these tests are presented in the following order: First, the models trained on CIC-IDS-2017 and tested on ESS-DMSC, then the models trained on ESS-DMSC and tested on CIC-IDS-2017.

The results are sorted according to the highest performing metrics in the following order: F1 score, Recall, Precision, Accuracy, AUC-ROC, AUC-PRC. The F1 score is chosen as the primary metric for evaluation, as it provides a balance of precision and recall, and effectively the model minimizes both false positives and false negatives. In the case of a tie in F1 score, the model is then sorted on recall, as minimizing false negatives is considered more important than reducing false positives in intrusion detection.

For the Autoencoder and the Convolutional Autoencoder, the 90th percentile error in reconstruction is used as the threshold to predict malicious instances, as it was found when testing to be the best performing threshold in terms of F1 score across both datasets.

1) CIC-IDS-2017 Train-Test: The best-performing models, one from each of the six machine learning algorithms, trained and tested on the CIC-IDS-2017 dataset, can be seen in Table X. The performance metrics are presented with the hyperparameters for each of the best-performing models. The order of the hyperparameters are in the same order as their description in the tables found in the respective sub-sections of Section IV-C6.

model	accuracy	precision	recall	f1_score	auc_roc	auc_prc	features	hyper parameters
AE	0.9054	0.9641	0.9203	**0.9417**	0.8700	0.9558	200	[[200], 0.4, 'LeakyReLU', 'MSELoss', 0.1, 0, 0.01]
CNN	0.9775	0.9556	0.9995	**0.9771**	0.9990	0.9989	200	[2, 32, 1, 3, 0, 64, 16, 'relu', 0, 'BCELoss', 0.001, 0, 0]
CAE	0.9042	0.9623	0.9207	**0.9410**	0.8637	0.9277	200	[2, 64, 1, 2, 0, 8, 'relu', 0, 'L1Loss', 0.1, 0, 0]
IF	0.8664	0.8618	0.9995	**0.9256**	0.5338	0.7967	10	[500, 0.7, 'auto', 1, True]
OCSVM	0.9218	0.9326	0.9764	**0.9540**	0.1235	0.6653	200	[0.3, 'sigmoid', 0.1, True]
XGB	0.9998	0.9996	1.0	**0.9998**	0.9999	0.9999	160	[10, 0.1, 500, 0.2, 1, 1.0, 0.5]

TABLE X
BEST MODELS TRAINED AND TESTED ON CIC-IDS-2017.

The **Autoencoder (AE)** model presented in Table X we can see that the model achieved an accuracy of 0.9054, which is above the baseline of 0.5 for a balanced datasets with binary classes. Indicating that the model is able to accurately predict both classes well. This is reemphasized by its precision of 0.9641, showing that for the predictions of instances belonging to the true positive class, malicious, it is correct in roughly 96% of its predictions. The model therefore displays a low false positive rate and a potential effective intrusion detection solution. The model also produces a recall of 0.9203, which means that of all instances belonging to the malicious class, it has correctly predicted roughly 92% of them. High recall is good

and a preferred metric for intrusion detection systems, because missing potential attacks can be devastating. Together, these numbers make the model produce an F1 score of roughly 0.94, indicating a good balance between precision and recall for an overall well-performing model. The model's AUC-ROC value of 0.87 indicates that the model has good measure of separability between the classes at various thresholds, meaning it is better at ranking a random positive instance higher than a random negative one. When compared to existing research, it is lower than the AUC-ROC 0.978 produced by Verkerken et al. on the same dataset [34]. The AUC-PRC value of 0.9558 tells us that the model is highly capable of distinguishing between the two classes, or that it is confident with the classes it has predicted.

The **Convolutional Neural Network (CNN)** model in Table X, the model shows an accuracy of 0.9775, a F1 score of 0.9771, a precision of 0.9556, with recall at 0.9995, AUC-ROC at 0.9990 and AUC-PRC at 0.9989. Not much can be said here other than that it has outperformed the Autoencoder in every metric aside from the precision, where it displays a slightly more proneness to false positives. Instead, looking at the CNN model's hyper parameters, the model has a latent dimension size of 16 neurons, out of possible 8, 16, 32 and 64, suggesting an effective model that is able to represent the features in a smaller dimension, and it may be reducing the overfitting to the dataset. However, dropout, L1 and L2 regularization are all zero, suggesting that the model complexity is right for the problem and that reducing overfitting is not necessary.

The **Convolutional Autoencoder (CAE)** model in Table X, the model achieves an accuracy of 0.9042, a precision of 0.9623, a recall of 0.9207, an F1 score of 0.9410, and AUC-ROC at 0.8737 and AUC-PRC at 0.9277. That is, the model is able to reduce both false positives and negatives, while being confident in its predictions at different thresholds and providing high separability between classes.

When comparing the number of features used between all three deep learning models of AE, CNN and CAE, it shows that they all used the maximum available features of 200 for their best performing models. Indicating a need for maximum features in order to capture the intricacies in the payload data for highest possible performance.

When comparing CAE with the regular AE, the replacement of fully connected layers with convolutional layers in the encoder part results in similar performance in all metrics, especially CAE's F1 Score of 0.9410 to AE's 0.9417. Suggesting that the difference in performance between the AE and CNN may not be down to the convolutional layers but instead the supervised contra unsupervised learning methods.

The **Isolation Forest (IF)** model seen in Table X the model achieves an F1 score of 0.9256, the lowest accuracy of 0.8664, the lowest precision of 0.8618, and a high recall of 0.9995. That is, it correctly identifies almost all malicious instances while achieving a comparably higher false-positive rate to that of the other models. Unlike the other machine learning algorithms, this model achieved its best results with only 10 features, which may explain its higher false-positive rate but may also produce lower overfitting and therefor better generalizability.

The **One-Class SVM (OCSVM)** model in Table X, shows an F1 score of 0.9540, an accuracy of 0.9218, precision of 0.9326, and recall of 0.9995. Menaing, it is able to detect almost all malicious instances and maintain comparably low false positive rate. Compared to results produced by Verkerken et al [34] on the same data, where the One-Class SVM achieved the lowest false positive of all the models, in this case the OC-SVM is the fifth best beaten by Autoencoder, CNN, CAE, and XGB. However, by its low AUC-ROC of 0.1236 and AUC-PRC of 0.6653, it is not confident in its predictions at different thresholds and cannot achieve as good separability between classes as the other models.

The **XGBoost (XGB)** model in Table X shows the best results across all metrics on the CIC-IDS-2017 dataset, with values close to perfect 1.0. Indicating that either the model has perfectly learned the patterns present in the data to predict the classes or there is overfitting present towards dataset.

Comparing the number of features used by the three shallow learning methods of Isolation Forest, One-Class SVM, and XGBoost, shows that IF used the lowest amount of 10, while OCSVM and XGBoost either used 200 or 160, which both are closer to those of the compared deep learning methods. Seemingly, the amount of features used has an impact on the model's precision or false-positive rate, not on its recall.

2) ESS-DMSC Train-Test: The best-performing models, one from each of the six machine learning algorithms, trained and tested on the ESS-DMSC dataset, can be seen in Table XI. The performance metrics are presented with the hyperparameters for each of the best-performing models. The order of the hyperparameters are in the same order as their description in the tables found in the respective sub-sections of Section IV-C6.

Starting with the **Autoencoder (AE)** model in Table XI. With an accuracy of 0.8333, it is not as accurate as the Autoencoder trained and tested on the CIC-IDS-2017 dataset, but still above the 0.5 baseline for a balanced binary dataset. The model has

model	accuracy	precision	recall	f1_score	auc_roc	auc_prc	features	hyper parameters
AE	0.8333	0.8559	0.8962	**0.8756**	0.8450	0.8758	20	[[20], 0, 'ReLU', 'L1Loss', 0.01, 0, 0]
CNN	0.8444	0.75	1.0	**0.8571**	0.9514	0.9431	200	[1, 32, 1, 2, 0, 32, 64, 'relu', 0.2, 'BCELoss', 0.001, 0, 0]
CAE	0.7469	0.7928	0.8302	**0.8111**	0.7460	0.7907	40	[3, 64, 1, 1, 0, 8, 'relu', 0, 'MSELoss', 0.01, 0, 0]
IF	0.8827	0.8537	0.9906	**0.9170**	0.1617	0.4755	20	[50, 1.0, 'auto', 1, True]
OCSVM	0.8889	0.8667	0.9811	**0.9204**	0.1530	0.4735	10	[0.3, 'rbf', 10, False]
XGB	0.7111	0.6250	0.9524	**0.7547**	0.6677	0.5914	100	[5, 0.01, 500, 0, 1, 1.0, 0.5]

TABLE XI
BEST MODELS TRAINED AND TESTED ON ESS-DMSC.

a precision of 0.8559, indicating a fairly low false positive rate, but again not as good as the CIC-IDS-2017 model. The recall of 0.8962 is closer to the CIC-IDS-2017 model's 0.9203, show good performance in predicting malicious instances in the ESS-DMSC dataset. The model's F1 Score of 0.8756 indicates that it is also able to balance precision and recall well. With an AUC-ROC of 0.84 it has a good measure of separability, and with an AUC-PRC of 0.87 it is able to reliably predict true positives and negatives. In summary, the Autoencoder trained and tested on the ESS-DMSC dataset performs well in all metrics, but not at the same level as the CIC-IDS-2017 Autoencoder model. This may be explained by the complexity in the patterns in the ESS-DMSC dataset's payloads or by the limitation in available data volume for training. The ESS-DMSC Autoencoder model also achieved its best results with only 20 features compared to the CIC model's 200, possibly also explained by the CIC's larger dataset and the available variance in benign instances used for training.

The **Convolutional Neural Network (CNN)** model in Table XI, it achieves an F1 score of 0.8571, an accuracy of 0.8444, a precision of 0.75 and recall of 1.0. While not as good as the best performing CIC-IDS-2017 model, the ESS-DMSC model is still able to predict all malicious attacks as shown by its perfect recall. Its high AUC-ROC and AUC-PRC shows good predictions across different thresholds and good separability between the classes. Similarly to the CIC model, the ESS-DMSC model also uses the maximum number of features of 200 from the payload, and this is also unlike the other models trained on the ESS-DMSC dataset, which all use 100 or fewer features. This difference may be explained by the limited amount of training data, in particular the number of malicious instances, as it is the other supervised learning method of XGBoost that uses the second highest number of features.

The **Convolutional Autoencoder (CAE)** model in Table XI. The model has an F1 score of 0.8111, an accuracy of 0.7469, a precision of 0.7928 and recall of 0.8302. Showing that there are still improvements to be found, both in terms of reducing false positives and increasing the prediction of malicious instances. Similarly to the regular Autoencoder, the Convolutional Autoencoder trained and tested on the ESS-DMSC dataset achieved its best results with fewer features than the CIC-IDs-2017 model, 40 features instead of 200. Once again, possibly explained by the different in dataset volume and the patterns found in the data used for training.

The **Isolation Forest (IF)** model in Table XI, produces an F1 score of 0.9170, an accuracy of 0.8827, a precision of 0.8537 and a recall of 0.9906. Showing an ability to predict almost all malicious instances in the dataset, while producing a relatively low false-positive rate, only reduced by the considerably lower AUC-ROC of 0.1617 and AUC-RPC of 0.4755. This indicates that it has predictive abilities at different thresholds that are not as consistent, and the separability between random true positive and true negative instances is not as clear as other models. Similarly to the CIC-IDS-2017 model, this model achieves its best results with only 20 features, once again promoting the possible efficiency by Isolation Forest.

The **One-Class SVM (OCSVM)** model in Table XI, achieves the highest F1 score of 0.9204 on the ESS-DMSC dataset, an accuracy of 0.8889, a precision of 0.8667, and a recall of 0.9811. Indicating a model capable of detecting close to all malicious instances while maintaining a fairly low false-positive rate. Its AUC-ROC and AUC-PRC values are nearly identical to the isolation forest model, suggesting that it has been able to learn the same patterns for predicting malicious instances.

The last model trained on ESS-DMSC dataset, **XGBoost (XGB)** in Table XI performed the worst of the six. This is opposite to the XGBoost model on the CIC-IDS-2017 dataset where it performed the best. However, is able to achieve a near perfect recall of 0.95, meaning it is able to detect almost all malicious instances, precision of 0.62 which show that the model is not reliable in its predictions and has a high false positive rate. This results in the lowest F1 score of 0.7547 for the dataset. Here, it is hypothesized that the limited amount of data, in particular malicious instances, is the factor for poorer performance. What is of note is that it still has a relatively high AUC-ROC and AUC-PRC, indicating that it is able to achieve high separability of the two classes. and high confidence in its predictions of the true positive and true negative instances.

3) CIC-IDS-2017 Train & ESS-DMSC Test: The best performing models, one from each of the six machine learning algorithms, trained on the CIC-IDS-2017 dataset, then tested on the ESS-DMSC dataset, can be seen in Table XII. The performance metrics are presented with the hyperparameters for each of the best-performing models. The order of the hyperparameters are in the same order as their description in the tables found in the respective sub-sections of Section IV-C6.

model	accuracy	precision	recall	f1 score	auc_roc	auc_prc	hyper parameters
AE	0.4773	0.4737	0.4091	0.4390	0.3863	0.5373	[[200], 0.4, 'LeakyReLU', 'MSELoss', 0.1, 0, 0.01]
CNN	0.4667	0.4667	1.0	0.6364	0.5278	0.4791	[2, 32, 1, 3, 0, 64, 16, 'relu', 0, 'BCELoss', 0.001, 0, 0]
CAE	0.3636	0.4000	0.5455	0.4616	0.3512	0.4589	[2, 64, 1, 2, 0, 8, 'relu', 0, 'L1Loss', 0.1, 0, 0]
IF	0.8863	0.8148	1.0	0.8980	0.0991	0.3228	[500, 0.7, 'auto', 1, True]
OCSVM	0.4667	0.4571	0.7619	0.5714	0.5099	0.4609	[0.3, 'sigmoid', 0.1, True]
XGB	0.5556	0.6667	0.0952	0.1667	0.5992	0.5806	[10, 0.1, 500, 0.2, 1, 1.0, 0.5]

TABLE XII
BEST MODELS TRAINED ON CIC-IDS-2017, TESTED ON ESS-DMSC.

Taking the best-performing models trained on the CIC-IDS-2017 dataset (Table X) and testing them on the ESS-DMSC dataset, establishes how well the trained model generalizes from an artificial dataset to a realistic unseen dataset. The results of this test can be seen in Table XII.

The **Autoencoder (AE)** model in Table XII has its accuracy reduced by 0.43, precision by 0.49, recall by 0.51, f1 score by 0.5, AUC-ROC by 0.48, AUC-PRC by 0.42. This shows strong overfitting to the CIC-IDS-2017 dataset and poor generalizability across all metrics to the ESS-DMSC dataset.

The **Convolutional Neural Network (CNN)** in Table XII also shows a reduction in the range of 0.35 to 0.5 units for all metrics but recall. Indicating that while performing well on the dataset it has been trained on as seen in Table X), the model is unable to generalize to the ESS-DMSC dataset. However, with it maintaining its perfect recall, it shows that the detection of malicious instances is transferable between datasets, but with the trade-off of a higher false positive rate of 0.5.

The **Convolutional Autoencoder (CAE)** in Table XII, displays the same drop in performance metrics as the CNN model when testing on the unseen ESS-DMSC dataset. However, unlike the CNN, the Convolutional Autoencoder also experiences the a 0.36 reduction in Recall, similarly to the regular Autoencoder. Showing poor generalizability for all metrics and an lowered ability to detect malicious instances, and a common disadvantage displayed by both of the two reconstruction models.

The **Isolation Forest (IF)** model in Table XII shows a decrease of less than 0.03 in F1 score, a slight increase in accuracy, a reduction in precision, and a perfect recall of 1.0. Suggesting that this model is able to generalize to the ESS-DMSC dataset. However, as evident by the decrease in AUC-ROC and AUC-PRC, separability and predictive capabilities at different thresholds have decreased, showing that while it is capable of performing well, it is not as confident in its predictions.

The **One-Class SVM (OCSVM)** model in Table XII experienced a reduction of over 0.4 in all metrics except the recall which saw a decrease of approximately 0.25. The model is also likely overfitted to the CIC-IDS-2017 dataset and is unable to generalize to the ESS-DMSC dataset well.

The **XGBoost** model in Table XII has its accuracy and precision drop by approximately 0.4, and recall decrease to just 0.09, showing an almost complete lack of ability of the model to detect the malicious instances in the ESS-DMSC dataset. In complete contrast to CNN, Isolation Forest, and One-Class SVM which maintained their perfect recall. The poor F1 score of 0.17 shows a model that is unable to generalize well and has probably been overfitted to the trained dataset.

4) ESS-DMSC Train & CIC-IDS-2017 Test: The best-performing models, one from each of the six machine learning algorithms, trained on the ESS-DMSC dataset, but tested on the CIC-IDS-2017 dataset can be seen in Table XIII. The performance metrics are presented with the hyperparameters for each of the best-performing models. The order of the hyperparameters are in the same order as their description in the tables found in the respective sub-sections of Section IV-C6.

Taking the best-performing models trained on the ESS-DMSC dataset (Table XI) and testing them on the CIC-IDS-2017 dataset, may not be as relevant when the opposite is tested, but by testing how well models trained on the realistic generalizes to the artificial dataset, it may surface patterns or potential model selection or architectures that can inform further training of models. The results of this test are presented in Table XIII.

Starting with the **Autoencoder (AE)** model in Table XIII which is similar to those when the opposite test, CIC-IDS-2017 to ESS-DMSC, was performed (Table XII). Accuracy, precision, AUC-ROC, and AUC-PRC are all within tenths of the corresponding metric of the opposite test. Recall is almost twice as high, 0.80 compared to 0.40, when testing the ESS-DMSC

model	accuracy	precision	recall	f1 score	auc_roc	auc_prc	hyper parameters
AE	0.4719	0.4832	0.8052	0.6040	0.4434	0.4468	[[20], 0, 'ReLU', 'L1Loss', 0.01, 0, 0]
CNN	0.5068	0.4933	0.9848	0.6573	0.3683	0.4864	[1, 32, 1, 2, 0, 32, 64, 'relu', 0.2, 'BCELoss', 0.001, 0, 0]
CAE	0.4307	0.7510	0.4306	0.5787	0.2989	0.7478	[3, 64, 1, 1, 0, 8, 'relu', 0, 'MSELoss', 0.01, 0, 0]
IF	0.5132	0.5067	1.0	0.6726	0.3230	0.3907	[50, 1.0, 'auto', 1, True]
OCSVM	0.5050	0.5025	1.0	0.6689	0.6640	0.6110	[0.3, 'rbf', 10, False]
XGB	0.5851	0.5369	0.9891	0.6960	0.4829	0.4490	[10, 0.1, 500, 0.2, 1, 1.0, 0.5]

TABLE XIII
BEST MODELS TRAINED ON ESS-DMSC, TESTED ON CIC-IDS-2017.

model on the CIC-IDS-2017 dataset, and the F1 score is 0.60 instead of 0.44. This shows that a model trained on the ESS-DMSC dataset is slightly more able to predict malicious instances in the CIC-IDS-2017 dataset, than that of the opposite test. This may be due to the smaller number of features used, 20 rather than 200, for this model, which allows the model to generalize more effectively.

The **Convolutional Neural Network (CNN)** model in Table XIII shows roughly the same drop in metrics compared to when the opposite test was performed. The model maintains a perfect recall, confirming the model's ability to predict malicious instances. However, as in the opposite test, the false positive rate is increased, as shown by the 0.4933 precision down from 0.75.

The **Convolutional Autoencoder (CAE)** model in Table XIII also sees roughly the same drop in metrics as when the opposite test was performed. The main difference being the precision maintaining a relatively higher value of 0.75 compared to 0.4. Showing that is more precise with its predictions, but it's ability to detect malicious instances is lower. Therefore, the Convolutional Autoencoder model does not generalize well to the ESS-DMSC dataset either.

The **Isolation Forest (IF)** model in Table XIII sees a drop in accuracy and precision to approximately 0.5 and a recall of 1.0 indicates that it is only able to accurately predict malicious instances reliably. The model is therefore unable to generalize to the CIC-DMSC dataset without an increase in false-positive rates.

The **One-Class SVM (OCSVM)** model in Table XIII experiences a loss in accuracy, precision, and F1 score of roughly 0.3, but maintains its perfect recall of 1.0. Similarly to Isolation Forest, CNN, and XBGoost, only the ability to detect malicious instances is completely generalizable, with the trade-off of an increase in false positive rate.

Finally, the **XGBoost** model in Table XIII also maintains its near perfect level of recall, while the other metrics are around the baseline of 0.5-0.6. Indicating that the model is not able to generalize to the dataset, due to an increase in false-positive rate.

5) Summary: Based on the models' performance in Tables X, XII, XI and XIII, the following can be said.

- All six models trained and tested on the CIC-IDS-2017 dataset show F1 scores and Recall above 0.92. All models except Isolation Forest are able to achieve a precision greater than 0.93.
- When the models trained on the CIC-IDS-2017 dataset are tested on the ESS-DMSC, a drop in F1 score of over 0.3 is experienced by all models but Isolation Forest. Isolation Forest sees only a drop of 0.02, indicating that it is able to generalize with low false positive and negative rates to the ESS-DMSC data.
- Isolation Forest's ability to perform well with only 10 features may be a benefiting factor to its ability to generalize.
- The six models trained and tested on the ESS-DMSC dataset are able to achieve F1 scores above 0.81. With Isolation Forest and One-Class SVM achieving 0.92, putting them ahead.
- When the models trained on ESS-DMSC dataset are tested on CIC-IDS-2017, all models see a drop in F1 score to 0.70 or lower, while CNN, IF, OCSVM, and XBG maintain near perfect recall. Indicating the ability to still identify the malicious instances present in the dataset. However, this trade-off leads to an increase in false positives, as shown by the relatively low precision of around 0.5.

VI. Discussion

The problem of detecting intrusions is very complex as perpetrators knowingly adapt dynamic strategies to avoid detection and employ increasingly sophisticated techniques [3]. To address this problem, the suitability of machine learning, and other data science techniques, are actively investigated and evaluated [7]. However, these techniques require large amounts of data, and the acquisition, labeling, and subsequent validation, of cybersecurity data are seen as a difficult, time-consuming, and error-prone process [13], especially if the dataset contains multiple attack vectors [41]. Although there are many publicly available research datasets, they are often outdated and do not reflect modern security concerns [11], [14]. Since acquiring data for machine learning is difficult, researchers often rely on using pre-existing datasets instead [11], [12], [14]. However, critics argue that there is an over-reliance on research datasets and that these datasets do not aid cybersecurity professionals in their work, and that models trained on these datasets offer limited insight into the applicability of machine learning models in real-world security contexts [14], [9]. Thus, the investigation of alternative data sources for training machine learning models in Intrusion Detection Systems (IDS), are of interest to cybersecurity professionals and researchers alike [7], [14]. The use of alternative sources of data, such as IDS logs, firewall logs, network traffic data, packet data, etc, for training models remains a relatively unexplored area of research [7].

1) A Novel Labeling Method: This research uses a combination of a popular research dataset (CIC-IDS-2017) [11] and real-world data supplied by a collaborating stakeholder (ESS-DMSC) [15] to train unsupervised and supervised machine learning models to investigate whether the two sources of data can be made comparable and if models are capable of generalizing between the two. In this pursuit, it is discovered that it is possible to leverage system log message data to label IP addresses present the raw network traffic captures, and that the two sources of data can be made to be comparable. Moreover, leveraging alternative sources of data, such as system log messages, can theoretically offer the possibility of utilizing the predictive power of tools commonly incorporated in large networks, for instance; Host Intrusion Detection Systems (HIDS) such as Open Source HIDS SECurity (OSSEC) [43], or Graylog [42].

Thus, the power of these systems can possibly be leveraged in the creation of new algorithms capable of evaluating the connection between system log messages and actors within the system. The method applied in this work describes how log messages can be represented as security related happenings, illustrated by Figure 9, 10 and 13. Figure 15 depicts an example of how different security related happenings, i.e., events, can be connected and structured as a graph. Hence, offering context at a glance for the qualitative labeling of IP addresses. In this example, the number of times that the given IP address appears in one of the connected events constitutes the weight of that particular edge. The logic of how weights are calculated can potentially be expanded to include other dimensions such as the absolute time differences between entries, etc. Interestingly, representing the connection between entity and happening as a graph can potentially allow for the automation of labeling. By assigning a severity score and a severity threshold to Figure 15, demonstrated in Figure 21. It is possible to demonstrate how the qualitative evaluation can become a quantitative process. Here, the severity scores for each node are multiplied by the respective weights and summed, resulting in a total score of 14, which is higher than the tolerated threshold of 12. Hence, labeling this specific entry as malicious. This approach would require assigning severity values to each event and specifying a tolerance threshold. However, doing so would potentially allow for automated evaluation of entries. Moreover, any evaluation that is applied, automated, or manual, can from a theoretical point of view be interpreted as the practical application of a security-policy rule-set [7]. Interpreting the label evaluations as such could theoretically allow researchers to label smaller segments of data manually and thereafter train a model to label the rest of the data by learning the implicit security-policy rule-set established by the researchers in the smaller segment. This could potentially enable easier training of real-world applicable anomaly-detection IDS by providing researchers with more dynamic and up to date datasets specific to the particular system [14], [11], [13], [7], effectively integrating both alternative sources of data and security related contexts, which are areas that have been highlighted as possible future work by researchers [7].

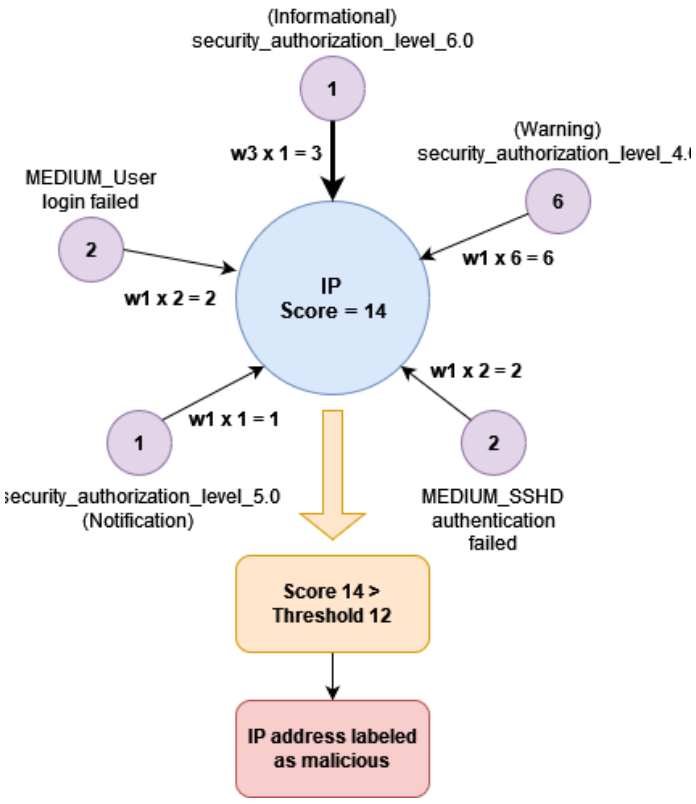

Fig. 21. Theoretical application of severity threshold for automating labeling

A. *Payload Anomaly-Based Detection*

Using the payloads from network packets rather than manually created statistics through feature engineering allows the model to find patterns in the data that may otherwise be lost when performing feature engineering. However, by using the payloads, it also increases the potential volume of the dataset, since payloads can be any size depending on the type of network traffic being captured. Therefore, it is necessary to reduce the number of features that are used in the payload by using models that can perform well with fewer features, mirroring the feature selection steps commonly applied to datasets with many manually created features [30]. The results in this thesis show that Isolation Forest was able to perform well with just 10 features in the CIC-IDS-2017 dataset, while the other models needed 160 or 200 features to only perform slightly better. This is relevant from an effectiveness perspective, as network traffic data is high volume and high velocity, meaning there is a lot of it over a short span of time. The ability of a model to use as little of the payload as possible for detection may prevent an intrusion detection system that would otherwise be too slow to react or not keep with the detection workload.

1) In relation to contemporary research: From the review of the literature on contemporary research applied to the CIC-IDS-2017 dataset, there is an apparent focus on the application of supervised learning methods. The search show impressive results for intrusion detection on an artificial dataset with accuracies and F1 scores near the perfect 1.0, and these results have been replicated by the implemented models in this thesis, through the implementation of a Convolutional Neural Network and XGBoost.

However, a question has also been raised in existing research about supervised learning and the ability of models trained with it to generalize to other datasets. It is asked how relevant are these model's perfect results in practicality when they need to be able to detect intrusions that are unlike those found in the training dataset and changing in nature. Instead, the branch of unsupervised learning has been raised as a potential solution to detecting unknown attack as they rely on outlier or anomaly detection when trained, rather than being trained more directly for what patterns to look for. Here, the results in this thesis have shown that the only model that can generalize with good performance metrics to the unseen realistic dataset is the unsupervised learning method of Isolation Forest. Showing an improvement over the results presented in the CIC-IDS-2017 to CIC-IDS-2018 generalization test [10], where the unsupervised models showed an average drop of 25% accuracy when tested on the unseen dataset.

In the results presented in this thesis, the two supervised learning models, CNN and XGBoost, were able to maintain high

recall in three out of the four cases when generalizing to an unseen dataset, showing that they could still detect malicious instances, but their false positive rate increased by a large factor. This may be an acceptable trade-off, depending on which type of attacks are being levied against the network or system. In the fourth case, XGBoost, which saw near-perfect performance metrics when trained and tested on the CIC-IDS-2017 dataset, saw the biggest drop in F1 score and Recall when tested on the ESS-DMSC dataset. Putting into question its ability to learn the more general patterns found in the dataset and severely hampering its ability to generalize. Furthermore, by testing the generalizability of supervised models in an approach similar to that presented by Verkerken et al. [10], an area of proposed future work is covered.

2) The broader implications: Anomaly-based detection through the use of data science, more specifically machine learning, is an active field of research. The results in this thesis directly contribute to the area, but also the broader implications of the things investigated can have an impact as an exploratory study. Finding that Isolation Forest being the only model that was able to generalize well to the completely unseen dataset without losing much in F1 score opens up an area for further research. What is also of note is that models generalized slightly better when trained on ESS-DMSC and tested on CIC-IDS-2017 rather the opposite. Hinting at the possibly that leveraging realistic datasets for training models and instead of artificial may be a way of increasing generalizability. As the ESS-DMSC dataset is smaller in volume than the CIC-IDS-2017 dataset, it might be a contributing factor to combat overfitting, since it allows the trained model to focus on learning the simpler patterns found in the data.

In discussion with the external stakeholder, when asked about the preference regarding the trade-off between high false positive rate and high false negative rate, it was raised that this depends on the nature of the attack. Although ideally a high F1 score is preferred, and this has been shown to be possible when taking the Isolation Forest model trained on the CIC-IDS-2017 dataset and testing it on the ESS-DMSC dataset. There is still a consideration to be made when evaluating the remaining model's results. For instance, for external stakeholder having a low false negative rate is important when trying to detect completely unknown, or so-called 0-day attacks, as missing these may have severe consequences. For more common or well-known attacks that have low severity, instead a low false positive rate is preferred, as it may impact the day-to-day operations if systems are temporarily shut down in order to prevent a low impact attack.

The results show that there is an identified commonality in the true positive class, malicious, between the two different datasets. As six out of the twelve models were able to maintain 0.98 or higher Recall when tested on the unseen dataset. Showing that unlike what related research states about the shortcomings of these commonly used datasets in intrusion detection research and their relevance in the real world [14], there is an additional potential bridge between theoretical and practical intrusion detection. The ability of some of the models to detect malicious instances with the trade-off of higher false positive rate warrants future research into how to reduce false positive rate after transferring the model to a new dataset.

When reflecting on the limitations of the payload anomaly detection, two things warrant discussion: The first is the limited amount of realistic data compared to the size of the artificial dataset. Because of this, the results of the models trained and tested on the realistic ESS-DMSC dataset may be questioned when judged on their own. However, also training and testing models of the same ML algorithms on the known CIC-IDS-2017 dataset, the validity of their implementation and ultimate performance increases as results by comparison to similar models in related research. Another limiting factor is the hyperparameter search for the models trained on the payloads. As the number of possible hyperparameter combinations can be in the millions, there is a strong case that with Random Search, the ideal combination has not yet been found. This means that better performance metrics may be possible for each of the models presented, and this is an area that can be explored in future work.

3) Improving intrusion detection research: With the results showing how the models generalize to a completely unseen dataset in mind, an area of future research is investigating how to leverage transfer learning by taking an existing model trained on an artificial dataset like CIC-IDS-2017 and then further training it with a sample of the realistic dataset. To see how quickly the different performance metrics improve, and how the model is more reliably being able to identify the malicious instances. Transfer learning through pre-trained models is an area that has seen great success in computer vision problems such as with the YOLO network. However, as of the time of writing, there are no such pre-trained models for intrusion detection problems. Therefore, if transfer learning shows great promise, a further area of future work would be to offer pre-trained intrusion detection models for everyone to use.

Ultimately, between the new labeling method using aggregated log data and the evaluation of models generalizability between datasets, can be used as an approach for creating new intrusion detection datasets that are more realistic or relevant to real-world instances. Therefore, a final suggestion for future work would be to use this approach to create new intrusion detection datasets and make them available for research by anyone. This direction of work would be motivated by requests in related intrusion detection research for more relevant and up-to-date datasets [7], [14].

VII. Conclusion

This thesis presents a novel labeling method for integrating system log messages with raw network traffic to establish a baseline of labels. These labels are combined with network traffic captures to effectively label individual payloads. Showing comparable formatting and contents to established research data. Multiple machine learning models are both trained and tested on both datasets to evaluate the generalizability of data and models used in machine learning research in Cybersecurity, showing how well different models are capable of generalizing and how false positives, contra false negatives, are affected by unseen data.

Revisiting RQ1, ”How can system log messages be integrated with network traffic in order to create a dataset capable of training machine learning models for the task of intrusion detection?”. It is possible to collect system log data by leveraging security related services commonly present in larger networks, such as Graylog [42] and OSSEC [43]. These log messages can be interpreted, or restructured, as events, with each event representing a security related happening within the network. By extracting identifiers, e.g., IP addresses and Port, from log messages, and counting the occurrence of each unique IP address in each event, it is possible to represent the relationship between IP addresses and events as a graph, see Figure 15. Visualizing the security related context of each IP address. Thus, enabling the qualitative assignment of a label to each unique IP address by assessing the security related context, resulting in labeled system log messages. These labeled log messages can then be matched with the network traffic using fuzzy matching logic. Allowing labels to be assigned to individual packet payloads, confirming the hypothesis; it is possible to label the log messages such that the log messages can be used to label the packet payloads from the raw network traffic.

The applied methodology results in the creation of a dataset by integrating both log messages produced by the components of the network with the network traffic itself. This dataset can be used for training machine learning models for intrusion detection by the detection of anomalous packet payloads. Hence, answering RQ1; log messages can be integrated with network traffic in order to create a dataset capable of training machine learning models for intrusion detection. In addition, the dataset is also comparable in format with the existing research dataset, CIC-IDS-2017 [11], [26], used in this work, aligning with the hypothesis; the labels can be used to create a realistic dataset that is comparable in format to an existing intrusion detection dataset for training machine learning models.

Revisiting RQ2, ”How does the performance metrics of machine learning models for intrusion detection compare when trained and tested on the same dataset versus when trained on one dataset and tested on an unseen dataset?”
First, the results show that all six implemented models, Autoencoder, CNN, Conv-Autoencoder, Isolation Forest, One-Class SVM, and XGBoost, show an F1 score higher than 0.92 on the artificial CIC-IDS-2017 dataset, and above 0.81 for all models except XGBoost on the realistic dataset ESS-DMSC. Detailed performance metrics and their analysis are presented in Section V-B.

Regarding the first hypothesis to the second research question, ”The performance metrics of the models differ depending on the model’s suitability for the characteristics of the dataset”. This assumption is correct, as the two supervised learning models of CNN and XGBoost perform best on the CIC-IDS-2017 dataset, while the unsupervised learning models of OC-SVM and Isolation Forest perform best on the ESS-DMSC dataset.

Continuing to the second part of the second research question regarding testing on an unseen dataset. The results show that when models trained on the CIC-IDS-2017 are tested on the realistic created for this study, ESS-DMSC, all models but Isolation Forest lose more than 0.3 in F1 score. Isolation is found to generalize better than the other implemented models, lending an argument to the use-case for models trained on artificial data in practical settings. However, considering that some models do not generalize as well, experimentation will be necessary to find which one is most suitable for the dataset being tested on. Therefore, the second hypothesis, ”There will be a loss in performance metrics for all models when tested on the unseen dataset as they are not able to fully generalize to the unseen dataset”. is also correct, but some models such as Isolation Forest show a smaller loss in performance metrics.

A. *Future Work*

This work has identified multiple areas for future research. First, further explore unsupervised models such as Isolation Forest, in their ability to generalize to other datasets and detect potentially unknown or different attacks. Secondly, investigate the trade-offs between high false positive rate and high false negative rate in the context of detecting zero-day attacks versus more common, low-impact attacks. Third, the investigation into improving the generalization ability of supervised learning methods in order to better translate their near-excellent performance when trained and tested on the same dataset. Fourth, the creation of realistic datasets for intrusion detection research using the labeling method proposed in this thesis. Fifth, leverage transfer-learning in order to improve the adaptability of models trained on artificial datasets when applied to realistic datasets or practical situations.

Future work related to labeling includes; automating the labeling process by the inclusion of a scoring system, training models to learn the implicit security-policy rule set established when determining labels and applying this rule-set to unlabeled entries to label large datasets using smaller segments, and validating the correctness of labeling when using automated techniques, or label propagation.

REFERENCES

[1] Cisco, "What is cybersecurity? - cisco," 2024, accessed: Jun 11, 2024. [Online]. Available: https://www.cisco.com/c/en/us/products/security/what-is-cybersecurity.html

[2] L. S. Vailshery, "Number of internet of things (iot) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030," *Statista*, vol. 2030, 2023.

[3] S. Papastergiou, H. Mouratidis, and E.-M. Kalogeraki, "Cyber security incident handling, warning and response system for the european critical information infrastructures (cybersane)," in *Engineering Applications of Neural Networks*. Springer, 2019, pp. 476–487.

[4] IBM, "Data breach report," https://www.ibm.com/reports/data-breach, 2024, [Online; accessed 19-January-2024].

[5] Statista, "Expected cost of cybercrime until 2027," https://www.statista.com/chart/28878/expected-cost-of-cybercrime-until-2027/, 2024, [Online; accessed 19-January-2024].

[6] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (idps)," *National Institute of Standards and Technology*, vol. 800-94, 2007.

[7] I. H. Sarker, A. S. Kayes, S. Badsha, H. Alqahtani, P. Watters, and A. Ng, "Cybersecurity data science: an overview from machine learning perspective," *Journal of Big Data*, vol. 7, 2020.

[8] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: the cicids2017 case study," in *2021 IEEE Security and Privacy Workshops (SPW)*, May 2021, pp. 7–12.

[9] L. Liu, G. Engelen, T. Lynar, D. Essam, and W. Joosen, "Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018," in *2022 IEEE Conference on Communications and Network Security (CNS)*, Oct 2022, pp. 254–262.

[10] M. Verkerken, L. D'hooge, T. Wauters, B. Volckaert, and F. D. Turck, "Towards model generalization for intrusion detection: Unsupervised machine learning techniques," *Journal of Network and Systems Management*, vol. 30, 2022.

[11] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy*, vol. 2018-January, 2018.

[12] A. Rosay, E. Cheval, F. Carlier, and P. Leroux, "Network intrusion detection: A comprehensive analysis of cic-ids2017," *International Conference on Information Systems Security and Privacy*, 2022.

[13] M. Lanvin, P.-F. Gimenez, Y. Han, F. Majorczyk, L. Mé, and É. Totel, "Errors in the cicids2017 dataset and the significant differences in detection performances it makes," in *Risks and Security of Internet and Systems*, S. Kallel, M. Jmaiel, M. Zulkernine, A. Hadj Kacem, F. Cuppens, and N. Cuppens, Eds. Cham: Springer Nature Switzerland, 2023, pp. 18–33.

[14] R. Dube, "Faulty use of the cic-ids 2017 dataset in information security research," *Journal of Computer Virology and Hacking Techniques*, vol. 36, no. 1, pp. 1–15, 2023. [Online]. Available: https://link.springer.com/article/10.1007/s11416-023-00509-7

[15] European Spalation Source, "Data management & software centre," https://europeanspallationsource.se/data-management-software-centre, 2024, [Online; accessed 24-January-2024].

[16] ——, "European spalation source," https://europeanspallationsource.se, 2024, [Online; accessed 31-January-2024].

[17] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, 2019. [Online]. Available: https://doi.org/10.1186/s42400-019-0038-7

[18] H. J. Liao, C. H. R. Lin, Y. C. Lin, and K. Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, 2013.

[19] Merriam-Webster, "Definition of cybersecurity," https://www.merriam-webster.com/dictionary/cybersecurity, 2024, [Online; accessed 19-January-2024].

[20] IBM, "Cybersecurity," https://www.ibm.com/topics/cybersecurity, 2024, [Online; accessed 19-January-2024].

[21] Y. Rbah, M. Mahfoudi, Y. Balboul, M. Fattah, S. Mazer, M. Elbekkali, and B. Bernoussi, "Machine learning and deep learning methods for intrusion detection systems in iomt: A survey," *2022 2nd International Conference on Innovative Research in Applied Science, Engineering and Technology, IRASET 2022*, 2022.

[22] M. Xu, K. M. Schweitzer, R. M. Bateman, and S. Xu, "Modeling and predicting cyber hacking breaches," *IEEE Transactions on Information Forensics and Security*, vol. 13, 2018.

[23] G. Kocher and G. Kumar, "Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges," *Soft Computing*, vol. 25, 2021.

[24] A. Laloui and H. Jalili, "CICFlowMeter," https://github.com/ahlashkari/CICFlowMeter, Accessed 2024.

[25] L. Liu, G. Engelen, T. Lynar, D. Essam, and W. Joosen, "Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018," in *2022 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2022, pp. 254–262.

[26] "University of new brunswick — unb," https://www.unb.ca/, leading discovery and innovation since 1785. Choose from 100+ study options at the University of New Brunswick's two campuses. Discover your potential at UNB.

[27] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "Rtids: A robust transformer-based approach for intrusion detection system," *IEEE Access*, vol. 10, pp. 64 375–64 387, 2022.

[28] J. Heaton, "An empirical analysis of feature engineering for predictive modeling," *Conference Proceedings - IEEE SOUTHEASTCON*, vol. 2016-July, 2016.

[29] R. A. Disha and S. Waheed, "Performance analysis of machine learning models for intrusion detection system using gini impurity-based weighted random forest (giwrf) feature selection technique," *Cybersecurity*, vol. 5, 2022.

[30] J. Lee, J. G. Pak, and M. Lee, "Network intrusion detection system using feature extraction based on deep sparse autoencoder," *International Conference on ICT Convergence*, vol. 2020-October, 2020.

[31] Z. Wang and Y. Zhang, "Ddos event forecasting using twitter data," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 0, 2017.

[32] R. M. Alguliyev, R. M. Aliguliyev, and F. J. Abdullayeva, "Deep learning method for prediction of ddos attacks on social media," *Advances in Data Science and Adaptive Analysis*, vol. 11, 2019.

[33] C. Soundarya and S. Usha, "Analyzing and predicting cyber hacking with time series models," *International Journal of Research in Engineering, Science and Management*, vol. 3, 2020.

[34] M. Verkerken, L. D'Hooge, T. Wauters, B. Volckaert, and F. D. Turck, "Unsupervised machine learning techniques for network intrusion detection on modern data," *2020 4th Cyber Security in Networking Conference, CSNet 2020*, 2020.

[35] K. Sadaf and J. Sultana, "Intrusion detection based on autoencoder and isolation forest in fog computing," *IEEE Access*, vol. 8, 2020.

[36] L. Yu, J. Dong, L. Chen, M. Li, B. Xu, Z. Li, L. Qiao, L. Liu, B. Zhao, and C. Zhang, "Pbcnn: Packet bytes-based convolutional neural network for network intrusion detection," *Computer Networks*, vol. 194, 2021.

[37] S. S. Dhaliwal, A. A. Nahid, and R. Abbas, "Effective intrusion detection system using xgboost," *Information (Switzerland)*, vol. 9, 2018.

[38] S. Wang, W. Xu, and Y. Liu, "Res-tranbilstm: An intelligent approach for intrusion detection in the internet of things," *Computer Networks*, vol. 235, 2023.

[39] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, 2007.
[40] J. W. Knopf, "Doing a literature review," *PS - Political Science and Politics*, vol. 39, 2006.
[41] A. Abdullah ALFRHAN, R. Hamad ALHUSAIN, and R. Ulah Khan, "Smote: Class imbalance problem in intrusion detection system," in *2020 International Conference on Computing and Information Technology (ICCIT-1441)*, Sep. 2020, pp. 1–5.
[42] Graylog, Inc., "Graylog," https://graylog.org/, Accessed 2024.
[43] Trend Micro, Inc., "OSSEC," https://www.ossec.net/, Accessed 2024.
[44] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*. Springer Cham, 01 2015, vol. 72.
[45] "Scapy," https://scapy.net/, accessed: 2024-05-06.
[46] M. Catillo, A. Pecchia, and U. Villano, "Successful intrusion detection with a single deep autoencoder: theory and practice," *Software Quality Journal*, vol. 32, 2024.
[47] M. Zhang, B. Xu, and J. Gong, "An anomaly detection model based on one-class svm to detect network intrusions," *Proceedings - 11th International Conference on Mobile Ad-Hoc and Sensor Networks, MSN 2015*, 2016.
[48] A. Saxena, "An introduction to convolutional neural networks," *International Journal for Research in Applied Science and Engineering Technology*, vol. 10, 2022.