AI Fuse: Intrusion Detection with Real & Synthetic Data

Final Senior Project Report Submitted to

The Department of Network Engineering and Security

Faculty of Computer and Information Technology

Jordan University of Science and Technology

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in
Network Engineering and Security

By

Asmae Baderja (142315)

Marwa El Ghali (142313)

MohammedNour Ebdah (142639)

Al-Muntuser Azzam (126618)

Spring 2024

DEDICATIONS

This project is sincerely dedicated to the Almighty, the core of our existence. His strength, wisdom, and consistent support have been the solid foundation of this venture. Likewise, we express heartfelt thanks to our parents for being the strong pillars of encouragement and inspiration in this Graduation Project. Their continuous presence and unwavering support fueled us with the strength and determination to overcome challenges. Without their ongoing love and help, bringing this project to fruition would have seemed like an impossible task.

نموذج حقوق الملكية الفكرية لمشاريع التخرج في قسم هندسة و أمن شبكات الحاسوب

يتم قراءة وتوقيع هذا النموذج من قبل الطلاب المسجلين لمشاريع التخرج في قسم هندسة و أمن شبكات الحاسوب

تعود حقوق الملكية الفكرية لمشاريع التخرج ونتائجها (مثل براءات الاختراع أو أي منتج قابل للتسويق) إلى جامعة العلوم والتكنولوجيا الأردنية، وتخضع هذه الحقوق إلى قوانين وأنظمة و تعليمات الجامعة المتعلقة بالملكية الفكرية وبراءات الاختراع.

بناءا على ما سبق أوافق على ما يلي:

1) أن أحفظ كافة حقوق الملكية الفكرية لجامعة العلوم والتكنولوجيا الأردنية في مشروع التخرج.
2) أن ألتزم بوضع اسم جامعة العلوم والتكنولوجيا الأردنية و أسماء جميع الباحثين المشاركين في المشروع على أي نشرة علمية للمشروع كاملا أو لنتائجه. و يشمل ذلك النشر في المجلات و المؤتمرات العلمية عامة أو النشر على المواقع الإلكترونية أو براءات الاختراع أو المسابقات العلمية.
3) أن ألتزم بأسس حقوق التأليف المعتمدة في جامعة العلوم والتكنولوجيا الأردنية.
4) أن أقوم بإعلام الجهة المختصة في الجامعة عن أي اختراع أو اكتشاف قد ينتج عن هذا المشروع و أن ألتزم السرية التامة في ذلك و أن أعمل من خلال الجامعة على الحصول على براءة الاختراع التي قد تنتج عن هذا المشروع.
5) أن تكون جامعة العلوم والتكنولوجيا الأردنية هي المالك لأي براءة اختراع قد تنتج عن هذا المشروع و تشمل هذه الملكية حق الجامعة في إعطاء التراخيص و التسويق و البيع كمؤسسة راعية و داعمة لكافة الأنشطة البحثية. و يكون حق للطالب شمول اسمه على براءة الاختراع كأحد المخترعين، و في حال تم إعطاء تراخيص أو تسويق و بيع لأي من منتجات المشروع يمنح المخترعون بما فيهم الطالب نسبة من الإيرادات حسب تعليمات البحث العلمي في جامعة العلوم والتكنولوجيا الأردنية.

إسم الطالب	*التوقيع*
إسم الطالب	*التوقيع*
إسم الطالب	*التوقيع*
إسم الطالب	*التوقيع*
إسم المشرف	***التوقيع***

تاريخ ....

ACKNOWLEDGEMENTS

This project would not have been possible without the unwavering support of our supervisor, Dr. Baha' Alsaify. We are extremely grateful for his guidance and assistance, which were crucial in shaping the project.
We also extend our heartfelt thanks to Jordan University for Science and Technology (JUST). The university has provided us with essential knowledge and skills, enabling us to complete this project. We wish for JUST continued success and growth as a leading institution of science and knowledge.
Our sincere appreciation goes to the Faculty of Computer Information Technology at JUST. Their expertise was vital in overcoming the challenges we faced during the project.
Finally, we thank the former graduates who generously shared their knowledge. Their insights greatly enriched our project.

ABSTRACT

This research project introduces "AI Fuse: Intrusion Detection with Real & Virtual Data", a pioneering advancement in cyber security. Our system compares real-time datasets with synthetic datasets to detect normal and abnormal activities using both deep learning and classical AI algorithms.

The core of the system comprises sophisticated software algorithms that monitor network activities and identify potential security breaches. By integrating deep learning techniques with classical AI methods, AI Fuse effectively distinguishes between normal and suspicious activities.

Preliminary testing has demonstrated high levels of accuracy in intrusion detection. The system's dual-dataset approach significantly improves its ability to detect anomalies.

This technology holds great potential for enhancing network security, providing a solid foundation for future innovations in this critical area. The findings and technologies developed through this project can substantially reduce the impact of cyber threats, thereby enhancing overall cyber security.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

1.1.1 The Importance of Intrusion Detection Systems

Intrusion detection systems (IDS) are vital components in modern cyber security frameworks. They serve as an essential line of defense, providing early warnings of potential attacks and unauthorized access [1, p. 5]. With the increasing sophistication of cyber threats, traditional IDS methods often reliant on static rule-based systems are becoming less effective. These conventional methods struggle to keep pace with the dynamic nature of modern cyber-attacks, which frequently employ novel techniques to evade detection. As a result, there is an urgent need to develop more advanced IDS capable of adapting to and identifying these evolving threats [2, p. 12].

1.1.2 Advancements in AI and Machine Learning for IDS

Recent advancements in artificial intelligence (AI) and machine learning (ML) offer promising avenues for enhancing IDS capabilities. AI-based systems can analyze vast amounts of data in real-time, identify patterns, and adapt to new threats more effectively than traditional methods [3, p. 24]. These systems leverage sophisticated algorithms to detect anomalies and predict potential security breaches, significantly improving the accuracy and reliability of IDS [4, p. 34]. This evolution in technology underscores the importance of integrating AI into IDS to bolster their effectiveness and ensure robust cyber defense mechanisms in the face of ever-evolving threats[5, p. 45].

1.2 Research Problem

1.2.1 Data Acquisition Challenges

Creating an effective AI-based intrusion detection system involves overcoming several significant challenges, with one of the primary difficulties being the acquisition of up-to-date and comprehensive datasets. Real-time datasets offer authentic insights but may not cover all potential threats, while synthetic datasets provide controlled environments but may lack real-world variability [6, p. 56]. The balance between authenticity and comprehensiveness in data is crucial for training and testing AI models effectively. Ensuring that these datasets are representative of the myriads of potential cyber threats is essential for the system's overall reliability and robustness [7, p. 67].

1.2.2 Algorithm Selection Challenges

Another major challenge lies in identifying new types of attacks, as cyber threats continually evolve, rendering many existing detection methods obsolete. Choosing the most effective algorithms is critical for addressing this issue.
Deep learning algorithms, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have shown great promise in detecting complex attack patterns due to their ability to learn from vast amounts of data [8, p. 78]. However, these models can be computationally intensive and may require large datasets for training. On the other hand, classical AI algorithms, such as decision trees and support vector machines (SVMs) offer simplicity and interpretability but may not be as effective in handling complex or subtle attack patterns [9, p. 89].

Addressing these challenges is crucial for enhancing the performance and reliability of IDS. Our project aims to tackle these issues by integrating real-time and synthetic datasets and employing both deep learning and classical AI algorithms to create a robust and efficient intrusion detection system [9, p. 89].

1.3 Objectives of the Study

The main goal of this study is to assess the accuracy and effectiveness of using both real and synthetic datasets in an AI-based intrusion detection system. This research aims to provide valuable insights and tools for organizations across various sectors that rely on robust IDS for protecting their networks and data.

1.3.1 Primary Objectives:

a) **To evaluate the effectiveness of various AI algorithms in detecting intrusions:** This involves testing different AI models, including deep learning and classical algorithms to determine which methods provide the highest accuracy and reliability in identifying abnormal activities [3, p. 24].

b) **To compare the performance of AI-based IDS on real and synthetic datasets**: By analyzing the detection capabilities of the IDS using both types of datasets, this study aims to understand the strengths and limitations of each approach. This comparison will help in developing a more comprehensive and effective intrusion detection strategy [6, p. 56].

1.3.2 Secondary Objectives:

a) **To identify key features and data patterns that contribute to successful intrusion detection:** Understanding these elements will aid in refining the algorithms and improving the system's overall performance [8, p. 78].

b) **To provide recommendations for integrating AI-based IDS into existing cyber security frameworks:** These guidelines will help organizations implement advanced IDS technologies effectively, enhancing their security posture against evolving cyber threats.

CHAPTER 2

REVIEW OF RELATED LITERATURE

This chapter presents the essential preparations needed to initiate the "AI Fuse: Intrusion Detection with Real & synthetic Data" project. It includes a review of related literature on intrusion detection systems, the use of real-time and synthetic datasets, and the application of deep learning and classical AI algorithms in cyber security. The discussion will cover previous related work and how our project builds upon and differentiates from these works.

Intrusion detection systems (IDS) have been a critical component in cyber security, providing the necessary tools to detect and prevent unauthorized access to network systems. The evolving landscape of cyber threats necessitates continuous advancements in IDS technology. This chapter explores the foundational literature in the field of intrusion detection, highlighting the significant studies and methodologies that have informed the development of our project. By examining prior research, we contextualize our approach and demonstrate the innovative aspects of "AI Fuse."

Intrusion Detection Systems

The development of IDS has undergone significant transformations since its inception. Traditional IDS, as discussed by Denning (1987) relied on rule-based detection methods to identify potential threats. These systems, while effective for known threats, struggled with novel or evolving attack vectors. To address these limitations, researchers began exploring anomaly-based detection methods, as highlighted by Lunt et al. (1990) which focused on identifying deviations from normal behavior.

Real-time vs. Synthetic Datasets

The use of real-time and Synthetic datasets in IDS research has been widely studied. Real-time datasets, such as those provided by KDD Cup 1999 (Tavallaee et al., 2009), offer authentic insights into network activities but often lack comprehensive coverage of all potential threats. Synthetic datasets, on the other hand, allow for controlled experimentation and the inclusion of diverse attack scenarios, as demonstrated by Shiravi et al. (2012) [8, p. 78].
Our project aims to integrate these two approaches, leveraging the strengths of both real-time and synthetic data to enhance detection accuracy.

Deep Learning in Intrusion Detection

The application of deep learning techniques in IDS has garnered significant attention in recent years. Works by Kim et al. (2016) and Yang et al. (2019) illustrate the effectiveness of convolutional neural networks (CNN) and recurrent neural networks (RNN) in detecting complex attack patterns [8, p. 78]. These studies underscore the potential of deep learning to improve detection rates and reduce false positives. Our project builds upon these findings by incorporating advanced deep learning models tailored for both real and simulated datasets [9, p. 89].

Classical AI Algorithms

ClassicalAI algorithms, such as decision trees, support vector machines (SVM), and k-nearest neighbors (KNN), have also been extensively used in IDS research. Studies by Auld et al. (2007) and Canedo et al. (2016) demonstrate the robustness of these methods in various detection

scenarios [4, p. 34]. While classical algorithms may not match the complexity handling capabilities of deep learning, their simplicity and interpretability offer valuable insights.
"AI Fuse" integrates these classical techniques with deep learning models to create a hybrid system that maximizes detection performance [5, p. 45].

Project Relation to Previous Work

"AI Fuse" builds on the extensive body of research in IDS by utilizing both real-time and synthetic datasets to compare the performance of various deep learning and classical AI algorithms. Unlike previous studies that typically concentrate on a single dataset type or detection method, our project systematically evaluates different datasets to determine which methods yield the best results. This approach aims to enhance detection accuracy and reliability by identifying the most effective algorithms, thereby addressing the limitations observed in earlier research [3, p. 24].

Conclusion

The review of related literature highlights the evolution of IDS and the significant advancements in using real-time and synthetic datasets, as well as deep learning and classical AI algorithms. "AI Fuse" seeks to advance this field by integrating these elements into a cohesive system, offering a novel approach to intrusion detection. This chapter sets the stage for the subsequent sections, where the methodology and implementation of our project will be discussed in detail.

CHAPTER 3

ANALYSIS AND DESIGN

This chapter outlines the analysis and design process for developing an enhanced Intrusion Detection System (IDS) aimed at effectively countering modern cyber threats. The project leverages both classical algorithms (Random Forest, k-NN, SVM) and Convolutional Neural Networks (CNN) to improve the accuracy and adaptability of IDS. It utilizes up-to-date datasets, namely the CSE-CIC-IDS2018 [6] and CIC Modbus 2023 datasets [7] and focuses on comparing the performance of the different classification techniques when applied to real and synthetic datasets. The sections below detail the design requirements, engineering standards, realistic constraints, alternative design approaches, and the developed design.

3.1 Design Requirements

This project addresses the challenge of enhancing Intrusion Detection Systems (IDS) to effectively counter modern cyber threats. The primary goal is to improve the detection accuracy and adaptability of IDS by integrating both classical algorithms and deep learning algorithm, and by using comprehensive and up-to-date datasets.

Comparative Analysis: The IDS must be capable of comparing real-time and synthetic datasets, specifically normal and abnormal communications.

Dataset Integration: The system must efficiently handle CSE-CIC-IDS2018 and CIC-Modbus 2023 datasets.

Algorithm Efficiency: Implement and optimize classical algorithms (Random Forest, k-NN, SVM) and CNN for IDS.

Storage and Processing: Ensure sufficient storage capacity and processing power for handling large datasets and running complex algorithms.

Scalability: Design the IDS to be scalable for future enhancements and integration with other datasets and algorithms.

3.2 Engineering Standards

The project adheres to the following engineering standards to ensure comprehensive, secure, and reliable intrusion detection capabilities:

a) CSE-CIC-IDS2018 Standards:
This dataset includes protocols such as HTTP, SMTP, FTP, and SSH, which the IDS must support. The CSE-CIC-IDS2018 standards provide guidelines for the development of intrusion detection systems (IDSs) that can effectively identify and respond to various types of network attacks [8]. By adhering to these standards, the project ensures that its IDS can detect and handle a wide range of network protocols and attacks.

b) CIC Modbus 2023 Standards:

This dataset incorporates the MODBUS, ACL, and ELK communication protocols, which the IDS must also support. The CIC Modbus 2023 standards provide specifications for the implementation of MODBUS, ACL, and ELK protocols in the IDS. This ensures that the IDS can effectively communicate with devices and systems that use these protocols, enhancing its ability to detect and respond to potential security threats.

c) Security Standards:

Follow best practices and standards for secure data handling and processing, including encryption and secure communication protocols.

3.3 Realistic Constrains

3.3.1 Economic Constraints:

a) **High storage requirements due to large dataset sizes:**

The project is working with large datasets, which require significant storage capacity. The sheer volume of data collected and processed by the intrusion detection system (IDS) places a high demand on the storage infrastructure. This can lead to increased costs for acquiring and maintaining the necessary storage solutions, whether on-premises or in the cloud.

b) **Budget limitations for acquiring and maintaining high-performance computing resources:**

In addition to the storage requirements, the project also needs access to high-performance computing resources to effectively analyze the large datasets and run the IDS algorithms. However, the project may face budget constraints that limit the ability to acquire and maintain the required computing power, such as powerful servers, GPUs, or cloud-based computing services. This can impact the performance and scalability of the IDS, potentially compromising its effectiveness in detecting and responding to security threats.

3.3.2 Manufacturability and Sustainability:

a) **Use of sensors in devices must be cost-effective and sustainable:**

The project aims to ensure that the use of sensors in devices is both cost-effective and sustainable. This involves considering the total cost of ownership, including the initial purchase price, maintenance, and upgrade costs.

b) **Consideration for the long-term maintenance and upgradability of the IDS**:

The project also considers the long-term maintenance and upgradability of the IDS. This includes ensuring that the system can be easily updated with new features, protocols, and threat signatures as needed.

3.3.3 Environmental Constraints:

Requirements for stable and secure environments to house sensors and other devices used in real-time data collection, while also considering bandwidth and power consumption constraints to ensure efficient and reliable operation. Additionally, factors such as physical security to prevent tampering, environmental conditions (temperature, humidity, etc.) to ensure device longevity, data privacy regulations to protect sensitive information, and scalability to accommodate future growth and increased data loads are crucial for maintaining a robust and effective system.

3.3.4 Other Constraints:

a) Ethical Considerations in data handling:
The project must ensure that it handles data in an ethical and responsible manner, respecting the privacy and security of the data.

b) Time constraints for project completion and deployment

3.4 Alternative/Different Designs Approaches/Choices

Single Dataset IDS:
Advantages: Simpler design and implementation, lower computational requirements.
Disadvantages: Limited applicability, less robust performance analysis.

Hybrid IDS using Real-time and Synthetic Data:
Advantages: Comprehensive analysis, better performance metrics, adaptable to different data types.
Disadvantages: Higher complexity, greater computational and storage requirements.

Chosen Approach:
The project opts for the hybrid IDS approach, which integrates real-time and synthetic datasets to provide a robust and comprehensive analysis of IDS performance.

3.5 Developed Design

The developed design incorporates both classical algorithms and DL for IDS. It leverages the CSE-CIC-IDS2018 and CIC Modbus 2023 datasets, focusing on a comparative analysis between real-time and synthetic datasets.

3.5.1 CNN Design

After acquiring the dataset, preprocessing is a must (deleting unnecessary columns, rows with negative values, and rows with (-∞: ∞) values), then we proceed to data transformation phase.

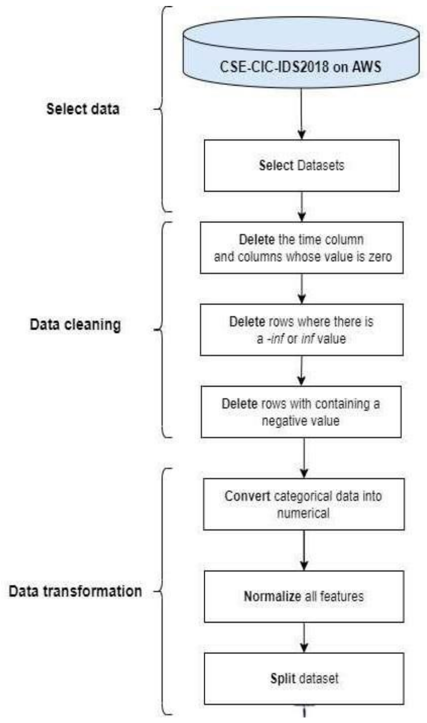

CSE-CIC-IDS2018 on AWS
Select data
Select Datasets
Data cleaning
Delete the time column and columns whose value is zero
Delete rows where there is a -inf or inf value
Delete rows with containing a negative value
Data transformation
Convert categorical data into numerical
Normalize all features
Split dataset

Figure 1: visualization the sequence of steps in our CNN model

The CNN architecture used in this model comprises three Conv1D layers, each followed by a Batch Normalization layer and a MaxPooling1D layer, to effectively capture and refine temporal patterns in the input time series data. After these convolutional and pooling operations, a Flatten layer converts the output into a 1D vector, which is then processed by two Dense layers with ReLU activation to learn high-level features. Finally, an output Dense layer with a softmax activation function classifies the data into one of the two target classes. This structured approach ensures efficient feature extraction and accurate classification

Layer (type)	Output Shape
conv1d (Conv1D)	(None, 72, 64)
Batch normalization	(None, 72, 64)
max_pooling1d	(None, 36, 64)
conv1d_1 (Conv1D)	(None, 36, 64)
batch_normalization_1	(None, 36, 64)
max_pooling1d_1	(None, 18, 64)
conv1d_2 (Conv1D)	(None, 18, 64)
batch_normalization_2	(None, 18, 64)
max_pooling1d_2	(None, 9, 64)
flatten (Flatten)	(None, 576)
dense (Dense)	(None, 64)
dense_1 (Dense)	(None, 64)
dense_2 (Dense)	(None, 3)

Table 1: illustrating the model architecture of our CNN model

3.5.2 Random Forest, SVM and KNN Design

The flowchart outlines the process of preparing and training an XGBoost classifier. It begins by defining folder paths and loading data from each folder, removing unnecessary columns, and sampling 50,000 rows per file. Categorical columns are then converted to dummy variables, and rows with missing values are dropped. The data is split into training (80%) and testing (20%) sets.

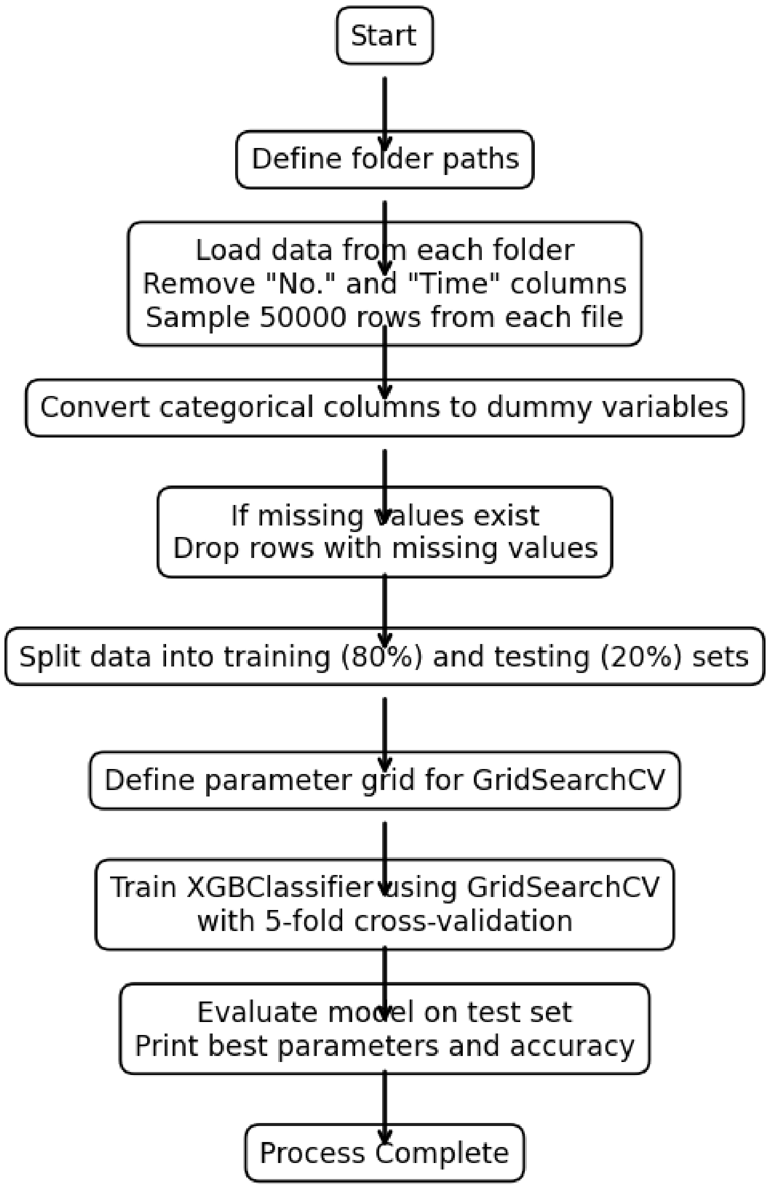

Start
Define folder paths
Load data from each folder
Remove "No." and "Time" columns
Sample 50000 rows from each file
Convert categorical columns to dummy variables
If missing values exist
Drop rows with missing values
Split data into training (80%) and testing (20%) sets
Define parameter grid for GridSearchCV
Train XGBClassifier using GridSearchCV
with 5-fold cross-validation
Evaluate model on test set
Print best parameters and accuracy
Process Complete

Figure 2: visualization the sequence of steps in our Random Forest, SVM, KNN model

CHAPTER 4

IMPLEMENTATION

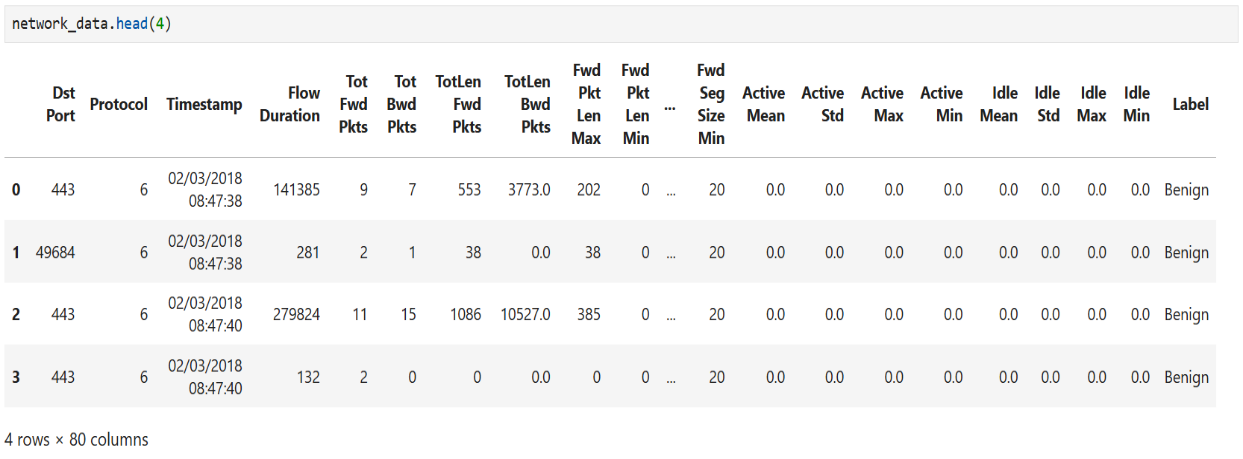
4.1 Dataset description

4.4.1 CSE-CIS-IDS2018 Dataset

The CSE-CIC-IDS2018 dataset is a real word intrusion detection dataset [8] containing about 16,233,002 instances of network traffic collected over 10 days. It is the most recent publicly available real dataset that covers a wide range of attack types. The dataset consists of 79-83 features and is distributed over 10 CSV files [9]. Each file is named according to the day the data was captured, and where each entry represents a record of a network flow. A network flow is a bidirectional sequence of packets between a specific source and destination and vice versa [6].

Classes	Description	Features
Flow duration	total duration of a flow from start to finish	fl_dur
Packet counts and sizes	Counts and sizes of packets exchanged in the forward and backward directions	tot_fw_pk, tot_bw_pk, tot_l_fw_pkt
Packet Size Metrics (Forward Direction)	Statistical measures of packet sizes in the forward direction	fw_pkt_l_max, fw_pkt_l_min, fw_pkt_l_avg, fw_pkt_l_std
Packet Size Metrics (Backward Direction)	Statistical measures of packet sizes in the backward direction	Bw_pkt_l_max, Bw_pkt_l_min, Bw_pkt_l_avg, Bw_pkt_l_std
Flow Rates	Rates of data transfer and packet exchange	fl_byt_s, fl_pkt_s, fw_pkt_s, bw_pkt_s
Flow Inter-Arrival Times	Times between consecutive flows	fl_iat_avg, fl_iat_std, fl_iat_max, fl_iat_min
Forward Direction Inter-Arrival Times	Times between packets sent in the forward direction	fw_iat_tot, fw_iat_avg, fw_iat_std, fw_iat_max, fw_iat_min
Backward Direction Inter-Arrival Times	Times between packets sent in the backward direction	bw_iat_tot, bw_iat_avg, bw_iat_std, bw_iat_max, bw_iat_min
Flags	Counts of specific TCP flags (e.g., PSH, URG) set in packets	fw_psh_flag, bw_psh_flag, fw_urg_flag, bw_urg_flag
Header Information	Total bytes used for headers in the forward and backward directions	fw_hdr_len, bw_hdr_len
Packet Length Metrics	statistical measures of packet lengths within a flow	pkt_len_min, pkt_len_max, pkt_len_avg, pkt_len_std, pkt_len_va
Control Packet Counts	Counts of specific control packets (e.g., FIN, SYN, RST) used in TCP communication	fin_cnt, syn_cnt, rst_cnt, pst_cnt, ack_cnt, urg_cnt, cwe_cnt, ece_cnt
Ratios and Averages	Derived metrics such as download/upload ratios and average packet sizes	down_up_ratio, pkt_size_avg, fw_seg_avg, bw_seg_avg
Bulk Metrics (Forward Direction)	Metrics related to bulk data transfer rates in the forward direction	fw_byt_blk_avg, fw_pkt_blk_avg, fw_blk_rate_avg
Bulk Metrics (Backward Direction)	Metrics related to bulk data transfer rates in the backward direction	bw_byt_blk_avg, bw_pkt_blk_avg, bw_blk_rate_avg
Sub-Flow Metrics	Average numbers of packets and bytes in sub-flows	subfl_fw_pk, subfl_fw_byt, subfl_bw_pkt, subfl_bw_byt
Initial Window Metrics	Bytes sent in the initial window in both directions	fw_win_byt, bw_win_byt
Active Packets	Count of packets with at least 1 byte of TCP data payload in the forward direction	Fw_act_pkt
Segment Metrics	Minimum segment size observed in the forward direction	fw_seg_min
Active and Idle Times	Times a flow remains active before becoming idle and idle before becoming active	atv_avg, atv_std, atv_max, atv_min, idl_avg, idl_std, idl_max, idl_min

Table 2: Classification and Description of Network Flow Features



```
network_data.head(4)
```

	Dst Port	Protocol	Timestamp	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	...	Fwd Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
0	443	6	02/03/2018 08:47:38	141385	9	7	553	3773.0	202	0	...	20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Benign
1	49684	6	02/03/2018 08:47:38	281	2	1	38	0.0	38	0	...	20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Benign
2	443	6	02/03/2018 08:47:40	279824	11	15	1086	10527.0	385	0	...	20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Benign
3	443	6	02/03/2018 08:47:40	132	2	0	0	0.0	0	0	...	20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Benign

4 rows × 80 columns

Table 3: Example of CSE-CIC-IDS 2018 Dataset

4.1.2 CIC Modbus 2023 Dataset

The CIC Modbus Dataset is designed to facilitate research and development in the security of substation networks using the Modbus protocol. It includes network captures and attacks logs from a synthetic substation network.
The primary purpose of this dataset is to support research, analysis, and development of intrusion detection systems, anomaly detection algorithms, and other security mechanisms for substation networks.

The dataset was generated from Wireshark captures in a simulated Docker environment, The Dataset is provided in PCAP format for network captures, chunked into 100MB files.

During the analysis of the Dataset, we converted PCAP files into CSV files with specific columns to facilitate detailed examination and processing.
The columns included in the CSV files are No., NO. Time, Source, Destination, Protocol, Length, Src port, Dst port, Ack, TSval, Seq, Syn, Fin, Reset, Push, Urg, Ece, Cwr, Win, TSecr, TCP Flags, Frame Length, and Label.

The label column consists of four different classes:

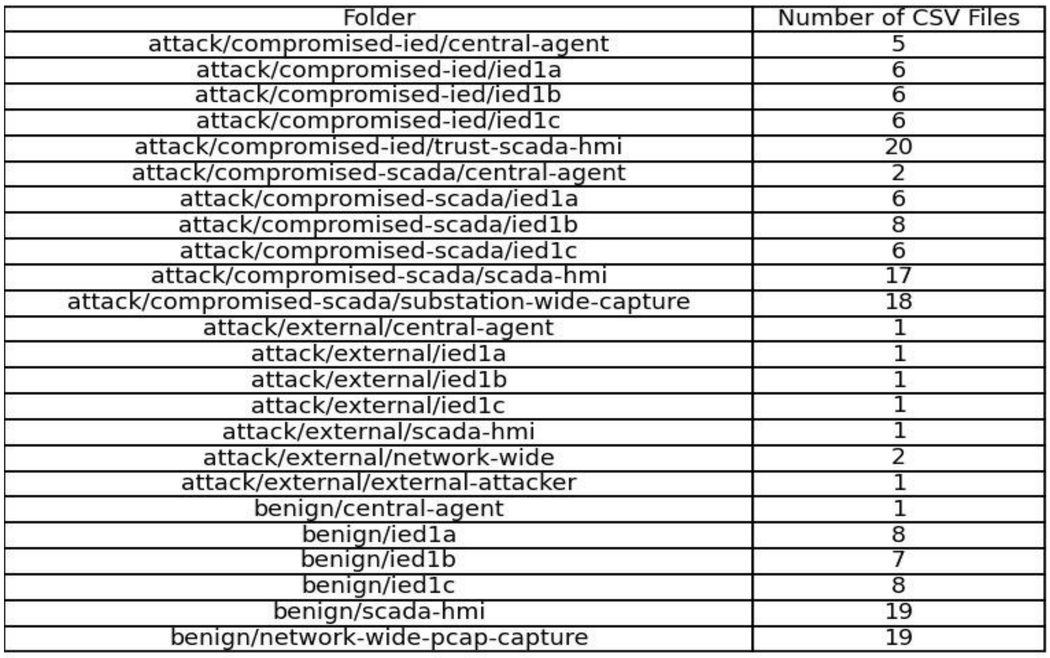
C-ied (compromised IED): This label indicates that the flow is associated with a compromised Intelligent Electronic Device (IED).

C-Scada (compromised SCADA): This label signifies that the flow is related to a compromised Supervisory Control and Data Acquisition (SCADA) system.

Benign: Normal and non-malicious.

External: This label identifies flows that originate from or are directed to external sources outside the monitored network.

To create the analyzed dataset, we selected one file from each folder within the dataset, ensuring a comprehensive representation of different types of network traffic and attack scenarios. This approach allows a balanced dataset that includes a variety of interactions and events within the substation network.



Folder	Number of CSV Files
attack/compromised-ied/central-agent	5
attack/compromised-ied/ied1a	6
attack/compromised-ied/ied1b	6
attack/compromised-ied/ied1c	6
attack/compromised-ied/trust-scada-hmi	20
attack/compromised-scada/central-agent	2
attack/compromised-scada/ied1a	6
attack/compromised-scada/ied1b	8
attack/compromised-scada/ied1c	6
attack/compromised-scada/scada-hmi	17
attack/compromised-scada/substation-wide-capture	18
attack/external/central-agent	1
attack/external/ied1a	1
attack/external/ied1b	1
attack/external/ied1c	1
attack/external/scada-hmi	1
attack/external/network-wide	2
attack/external/external-attacker	1
benign/central-agent	1
benign/ied1a	8
benign/ied1b	7
benign/ied1c	8
benign/scada-hmi	19
benign/network-wide-pcap-capture	19

Table 4: The distribution of CSV files across various folders, categorized by attack or benign scenarios.

4.2 Attacks Description

The datasets comprise multiple files, each containing one or more distinct attacks. In the subsequent sections, we will provide a comprehensive overview of the attacks present in each dataset.

4.2.1 The CIC Modbus Dataset Attacks:

The Modbus dataset includes various attacks such as:

Reconnaissance: Gathering information about the network to identify potential vulnerabilities
Query Flooding: Sending many queries to overwhelm the Modbus server.
Loading Payloads: Sending malicious payloads to Modbus devices to alter their normal operation.
Delay Response: Introducing delays in communication between devices. Disrupts timing-dependent operations within the substation.
Modifying Length Parameters: Altering length parameters in Modbus messages. Confuses the receiving device, causing it to process data incorrectly.
False Data Injection: Sending incorrect data to Modbus devices. Leads to erroneous decisions based on false information

Stacking Modbus Frames: Stacking multiple Modbus frames in a single message. Overwhelms or confuses the receiving device.
Brute Force Write: Repeatedly attempting to write data to Modbus devices using brute force techniques. Aims to guess the correct commands or parameters.
Baseline Replay: Replaying previously captured legitimate Modbus traffic to the network. Causes devices to respond to outdated or irrelevant commands.

4.2.2 The CSE-CIC-IDS2018 dataset Attacks

The attacks available in this dataset are:

Bot attack: involves using automated software (bots) to perform malicious activities on the internet such as: disturbing the services, stealing data, and causing significant financial and reputational damage.
Infiltration: it refers to the unauthorized access and penetration of a network or system by an attacker. It can lead to data breaches, loss of sensitive information, and compromised system integrity.
SSH/FTP Bruteforce: are types of cyberattacks where an attacker attempts to gain unauthorized access to SSH (Secure Shell) or FTP (File Transfer Protocol) servers by systematically trying many username and password combinations.
DDoS attack-LOIC-UDP and DDoS attack-HOIC: Distributed Denial of Service (DDoS) attacks flood a target system with excessive traffic to render it inaccessible.

> **LOIC (Low Orbit Ion Cannon** An open-source tool for DDoS attacks, performing UDP floods by sending numerous packets to overwhelm the server's resources.
> **HOIC (High Orbit Ion Cannon)**: A more advanced tool than LOIC, generating significant traffic via HTTP floods to overwhelm the target server, with support for customizable URL targeting.

Brute Force-Web: Brute force attacks against web applications involve systematically trying numerous username and password combinations to gain unauthorized access.
Cross-Site Scripting (XSS): is a vulnerability in web applications where an attacker injects malicious scripts into content that is delivered to other users.
SQL injection: is a code injection technique that exploits vulnerabilities in an application's software by injecting malicious SQL queries through input fields.
DoS Attacks - GoldenEye and Slowloris: Denial of Service (DoS) attacks aim to target system unavailable to legitimate users by overwhelming it with malicious traffic.

> **GoldenEye:** A tool for HTTP-based DoS attacks that generates numerous HTTP requests to overwhelm the target server.
> **Slowloris**: A low-bandwidth DoS attack tool that maintains multiple idle connections to the target web server, exhausting its resources.

DoS Attack - Slow HTTP Test: A tool that simulates slow HTTP attacks, keeping many connections open with minimal traffic to exhaust server resources.
DoS Attack - HULK (HTTP Unbearable Load King): HULK is a DoS attack tool that generates a high volume of unique and random HTTP requests to overwhelm and make a web server unreachable.

4.3 Algorithms & Implementations

4.3.1 The chosen Algorithms

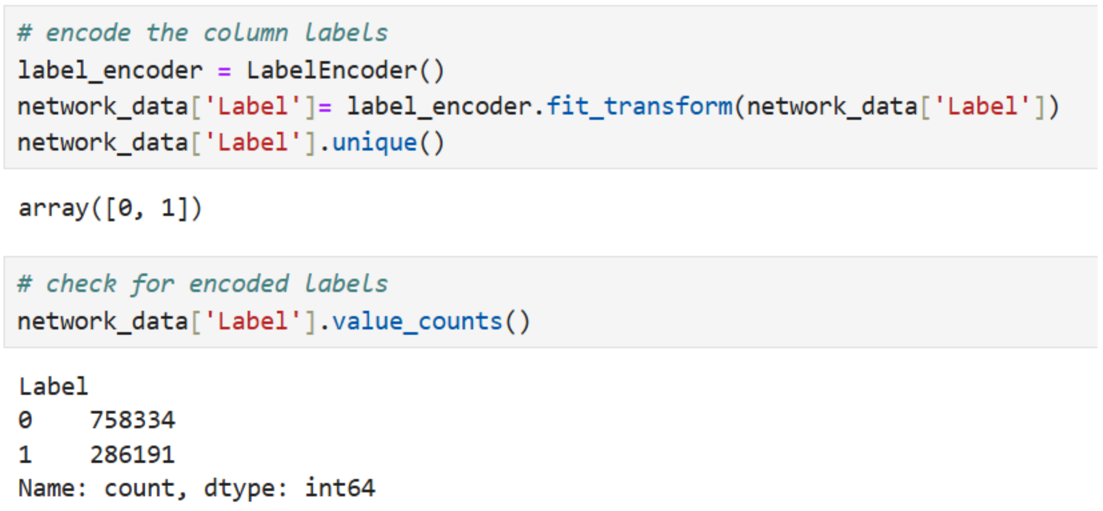
K-Nearest Neighbors (KNN): A simple, instance-based learning algorithm that classifies a data point based on how its neighbors are classified. It assigns a class to the data point which is the most common among its K nearest neighbors in the feature space.

Support Vector Machine (SVM): A supervised learning algorithm used for classification and regression tasks. It works by finding the hyperplane that best separates the data into different classes, maximizing the margin between the closest points of the classes (support vectors).
Random Forest: An ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. It improves accuracy and controls overfitting by averaging multiple decision trees.
Convolutional Neural Network (CNN): A deep learning algorithm primarily used for processing structured grid data like images. It employs convolutional layers that automatically and adaptively learn spatial hierarchies of features, making it highly effective for image recognition and classification tasks.

4.3.2 Algorithm Implementation

Common Mechanisms:

1- Loading and Preprocessing Data: All scripts include a function to load data from CSV file.
2- Unnecessary columns are dropped.
3- Data is sampled to a specified sample size (50.000 sample from each class) to manage memory and computation while also ensuring data balance.
4- One-hot encoding is used for categorical data, for example replacing "benign" with 0, "FTP-BruteForce" with 1.
5- Data Splitting: Data is divided into training and testing sets using train_test_split function.
6- The target variable is assumed to be the last column in the dataset.
7- Cross-validation is employed to ensure the robustness of the model's performance.
8- Model Training and Evaluation: The models are trained using the training set and evaluated on the testing set. Accuracy is used as the primary metric for evaluation.



```
# encode the column labels
label_encoder = LabelEncoder()
network_data['Label']= label_encoder.fit_transform(network_data['Label'])
network_data['Label'].unique()
```

```
array([0, 1])
```

```
# check for encoded labels
network_data['Label'].value_counts()
```

```
Label
0    758334
1    286191
Name: count, dtype: int64
```

Figure 3: Encode the column Labels

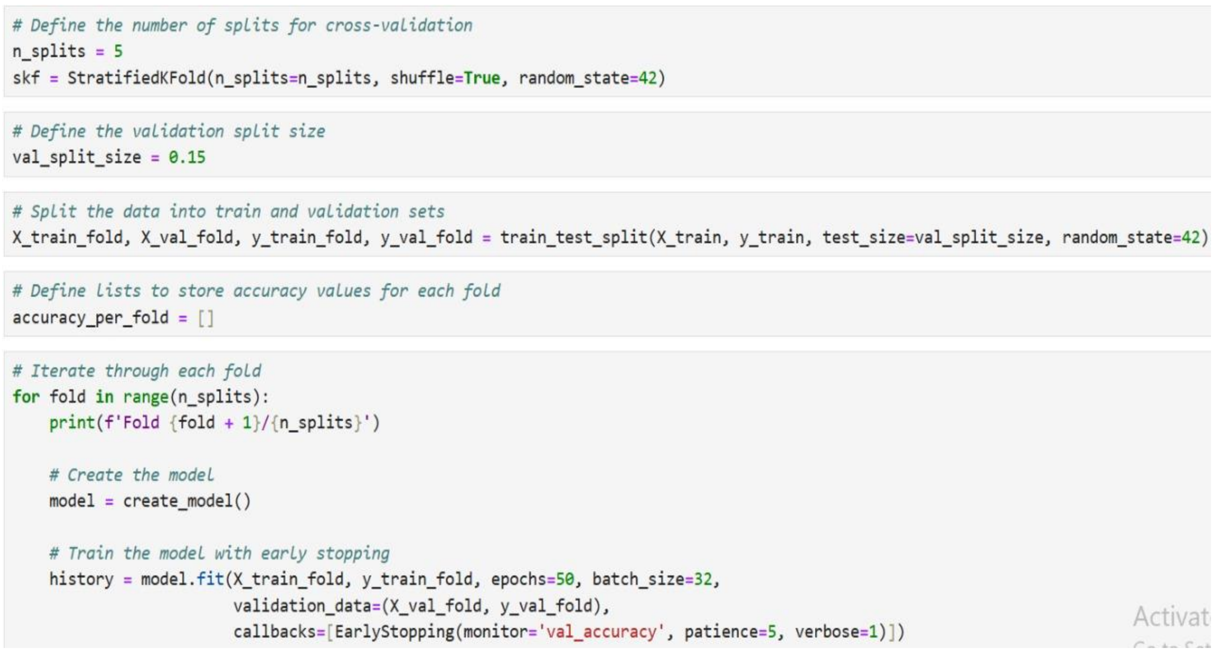
Differences:

1- Pipeline and GridSearchCV:

GridSearchCV is used for hyperparameter tuning in KNN, SVM and random forest. Pipelines are used to chain preprocessing steps with the model.

Problems:

a) The first problem faced during the work was overfitting, it means that the model was trained enough times to memorize a certain pattern. In our attempt to solve it, we used early stopping, where we monitored the value of val_accuracy (validation accuracy) for five consecutive epochs, if no improvement is seen we stop training.



```
# Define the number of splits for cross-validation
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=42)
```

```
# Define the validation split size
val_split_size = 0.15
```

```
# Split the data into train and validation sets
X_train_fold, X_val_fold, y_train_fold, y_val_fold = train_test_split(X_train, y_train, test_size=val_split_size, random_state=42)
```

```
# Define lists to store accuracy values for each fold
accuracy_per_fold = []
```

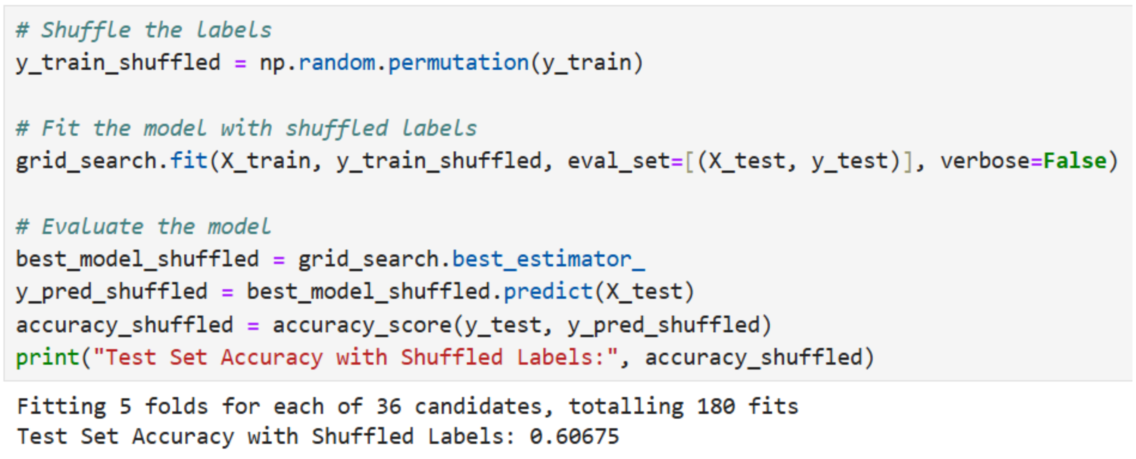
```
# Iterate through each fold
for fold in range(n_splits):
    print(f'Fold {fold + 1}/{n_splits}')

    # Create the model
    model = create_model()

    # Train the model with early stopping
    history = model.fit(X_train_fold, y_train_fold, epochs=50, batch_size=32,
                        validation_data=(X_val_fold, y_val_fold),
                        callbacks=[EarlyStopping(monitor='val_accuracy', patience=5, verbose=1)])
```

Figure4: representation of how cross-validation and early stopping are implemented

b) The second challenge arose from the observation of achieving 100% accuracy using random forest models on both real-time and synthetic datasets. To address concerns of overfitting, a step was taken to introduce label shuffling followed by model retraining, resulting in significantly reduced accuracy. This indicated that overfitting was not the primary issue. Additionally, suspicions were raised regarding potential data leakage during the data splitting process, prompting further code adjustments to mitigate this possibility.
Through these two approaches, we concluded that the small sample size may be the primary reason behind the high accuracy. Unfortunately, due to limited resources, we were unable to increase the sample size.



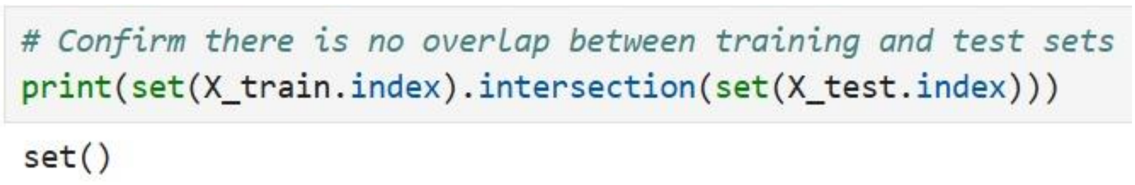
```
# Shuffle the labels
y_train_shuffled = np.random.permutation(y_train)

# Fit the model with shuffled labels
grid_search.fit(X_train, y_train_shuffled, eval_set=[(X_test, y_test)], verbose=False)

# Evaluate the model
best_model_shuffled = grid_search.best_estimator_
y_pred_shuffled = best_model_shuffled.predict(X_test)
accuracy_shuffled = accuracy_score(y_test, y_pred_shuffled)
print("Test Set Accuracy with Shuffled Labels:", accuracy_shuffled)
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Test Set Accuracy with Shuffled Labels: 0.60675
```

Figure 5: Label shuffling



```
# Confirm there is no overlap between training and test sets
print(set(X_train.index).intersection(set(X_test.index)))
```

```
set()
```

Figure 6: Verifying no Data Leakage happened

CHAPTER 5

TESTING AND EVALUATION

In this section, we detail the testing and evaluation of our Intrusion Detection System (IDS) using AI algorithms on both real-time and synthetic datasets. We aimed to measure the performance and robustness of these models in detecting intrusions, addressing potential issues like overfitting and data leakage.

Evaluation of Algorithm Performance on real dataset

The following table shows the results we obtained after training the CIC-IDS 2018 dataset on 4 different algorithms: Random Forest, K-NN, SVM, and CNN

Attacks	Algorithm	Test accuracy
Bot	Random Forest K-NN SVM CNN	100% 99.97% 99.92% 75.25%
Dos attacks-Hulk and DoS attacks-SlowHTTPTest attacks	Random Forest K-NN SVM CNN	100% 100% ------- 89.15%
infiltration	Random Forest K-NN SVM CNN	100% 63.49% ------- 59.78%
SSH and FTP brute force	Random Forest K-NN SVM CNN	100% 99.99% 99.99% 81.67%
DDoS attacks-LOIC-HTTP	Random Forest K-NN SVM CNN	100% 99.64% ------- 98.68%
infiltration	Random Forest K-NN SVM CNN	100% 56.91% ------- 53.80%

Table 5: Test Accuracy of Different Algorithms on Various Attack Types

The SVM model was not effective on certain files, primarily due to its lengthy training time, which highlights the inadequacy of our current resources for handling this dataset with SVM.

Evaluation of Algorithm Performance on synthetic dataset

Random Forest Algorithm achieved an accuracy of 100%
K-Nearest Neighbors (KNN) Algorithm achieved an accuracy of 98.3%
Convolutional Neural Network (CNN) Algorithm achieved an accuracy of 98.8%.
SVM, unfortunately, the hardware wasn't compatible.

Conclusion

These results demonstrate that while the AI algorithms performed exceptionally well on the synthetic dataset, their performance on the real-time dataset was notably lower. This discrepancy highlights the inherent challenges posed by real-world data, which include various forms of noise, variability, and imbalances.

In synthetic datasets, conditions are often controlled and idealized, leading to cleaner data with clear distinctions between classes. However, real-time datasets frequently contain noise—random or irrelevant data that can obscure meaningful patterns. For example, benign network traffic might include occasional unusual behavior that resembles malicious activity, or there may be missing or corrupted data packets that add to the confusion.

Variability in real-time data further complicates model performance. Unlike synthetic data where features and patterns are consistent, real-world data can exhibit significant fluctuations. Network traffic patterns can vary widely depending on the time of day, user behavior, or other external factors, making it harder for models to generalize from the training data to unseen instances.

Additionally, real-time data often presents imbalances between benign and malicious activities, with benign packets typically vastly outnumbering malicious ones. This imbalance can lead to models being biased towards predicting the majority class, thereby missing the rare but critical instances of malicious activity.

To address these issues, we implemented label shuffling, a technique that helps to test the robustness of the model by randomly assigning labels and retraining the model. This approach helped us verify that the high accuracy observed with the random forest algorithm was not due to overfitting or data leakage. By ensuring that the model's performance degraded as expected when labels were shuffled, we confirmed that the original high accuracy was indeed valid and not an artifact of improper data handling.

Overall, these findings underscore the importance of robust preprocessing and validation techniques in developing Intrusion Detection Systems (IDS) that can handle the complexities of real-world data. Ensuring that models can generalize well beyond controlled synthetic environments is crucial for their effectiveness in practical applications.

CHAPTER 6

CONCLUSION

In this final chapter, we summarize the key findings and conclusions of our project, "AI Fuse: Intrusion Detection with Real & Synthetic Data."
This work aimed to compare the effectiveness of various AI algorithms, both deep learning and classical, on real-time and synthetic datasets for intrusion detection.
Our comprehensive analysis demonstrated several important conclusions:

1. **Performance Comparison:**
   Synthetic Dataset: The AI models, including Random Forest, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Convolutional Neural Network (CNN), exhibited high accuracy on the synthetic dataset. Specifically, Random Forest initially achieved 100% accuracy.
   Real-Time Dataset: In contrast, the models performed effectively on the real-time dataset. The drop in accuracy underscores the challenges posed by real-world data.

2. **Reasons for Higher Accuracy in Synthetic Dataset:**
   Data Quality and Labeling: synthetic datasets often have well-defined and clean labels, making it easier for models to learn and make accurate predictions. In contrast, real-time datasets can be noisy and contain mislabeled data, complicating the learning process.
   Controlled Environment: synthetic datasets are generated in controlled environments, ensuring consistent data patterns and fewer anomalies. Real-time data, however, includes a wide variety of unpredictable patterns, increasing the complexity for AI models.
   Balanced Data Distribution: synthetic datasets are usually balanced regarding the number of benign and malicious records. Real-time datasets often suffer from class imbalance, where benign data significantly outnumbers malicious data, affecting model training.
   Feature Informativeness: Features in synthetic datasets are designed to be highly informative for distinguishing between normal and malicious activities. Real-time datasets may include less informative features or more noise, which can hinder model performance.

3. **Challenges Encountered:**
   Overfitting: The initial high accuracy of the Random Forest on both datasets suggested potential overfitting. By employing techniques such as label shuffling and careful data splitting, we ensured robustness in model evaluation, which revealed the true performance of the models.
   Resource Limitations: The analysis was constrained by computational resources, limiting the sample size to 50,000 rows from each CSV file. Larger datasets might provide a more comprehensive evaluation but were not feasible due to these limitations.

4. **Strengths of Our Approach:**
   Dual-Dataset Evaluation: By comparing real-time and synthetic datasets, we provided a more thorough understanding of each algorithm's strengths and limitations.
   Robust Testing Methods: Techniques like label shuffling and cross-validation helped ensure that our models were evaluated rigorously, reducing the risk of overfitting and data leakage.

5. **Implications for Future Work:**

   Improving Data Quality: Enhancing the quality and labeling of real-time datasets can significantly improve model performance. Implementing noise reduction and better labeling practices will be beneficial.

   Advanced Feature Engineering: Developing more sophisticated feature extraction methods can help improve the informativeness of real-time data.

   Scalability: Future work should focus on improving the scalability of the system to handle larger datasets, which could provide more accurate and reliable intrusion detection.

In conclusion, our project highlights the importance of dataset quality and controlled environments in achieving high accuracy with AI models for intrusion detection. While synthetic datasets provide ideal conditions for model training, real-time data presents practical challenges that require advanced techniques and robust testing methods. This research provides valuable insights and methodologies for enhancing intrusion detection systems, contributing to the ongoing development of more secure and resilient cyber defense mechanisms.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT and perplexity tools to enhance the readability of some paragraphs. After using this tool/service, the author reviewed and edited the content as needed and took full responsibility for the content of the publication.

References

[1] geeksforgeeks, "Intrusion Detection System (IDS)," 23 May 2024. [Online]. Available: https://www.geeksforgeeks.org/intrusion-detection-system-ids/.

[2] Michal MARKEVYCH, Maurice DAWSON, "A REVIEW OF ENHANCING INTRUSION DETECTION SYSTEMS," in *KNOWLEDGE-BASED ORGANIZATION*, Chicago, 2023.

[3] Haiyan Xuan, Mohith Manohar, "Intrusion Detection System with Machine Learning and Multiple," Department of Computer Science, Columbia University, New York, 2023.

[4] A. K. Gupta, "AI-Based Intrusion Detection: Enhancing Cybersecurity Defense," Insights2Techinfo, 2023. [Online]. Available: https://insights2techinfo.com/ai-based-intrusion-detection-enhancing-cybersecurity-defense/.

[5] HUANLONG YIN, YUEFEI ZHU, JINLONG FEI, AND XINZHENG HE, " A Deep Learning Approach for Intrusion," State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, 2017.

[6] Iman Sharaf Aldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *4th International Conference on Information Systems Security and Privacy (ICISSP)*, portugal, 2018.

[7] C. I. f. Cybersecurity, "CIC Modbus Dataset 2023," University of New-Brunswick, August 2023. [Online]. Available: https://www.unb.ca/cic/datasets/modbus-2023.html.

[8] Baraa Ismael Farhan, Ammar D. Jasim, "Performance analysis of intrusion detection for deep learning," Indonesian Journal of Electrical Engineering and Computer Science, Baghdad, 2022.

[9] Joffrey L. Leevy and Taghi M. Khoshgoftaar, "A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data," 2020.