Transferability and Explainability of Machine Learning Models for Network Intrusion Detection

Final Project Report

Author: Ahmed Bedair

Supervisor: Dr. Fabio Pierazzi

Student ID: K2105577

March 18, 2024

Abstract

Machine learning-based network intrusion detection systems (NIDS) have emerged as a promising solution to detect novel and evolving cyber threats. By learning patterns and anomalies from network traffic data, algorithms such as random forests and support vector machines (SVMs) can identify previously unseen intrusions. This research focuses on the application of machine learning classifiers in NIDS, utilizing flow-based features extracted from network traffic. The study investigates the transferability and generalizability of learned patterns across different datasets, simulating the real-world scenario of training a model on one dataset and deploying it to detect intrusions in the wild. By analyzing the statistical properties and distributions of flow-based features, this research aims to uncover the key characteristics that distinguish malicious traffic from benign traffic, emphasizing the importance of dataset representativeness and biases. Furthermore, this research employs explainable AI techniques, such as SHAP (SHapley Additive exPlanations), to identify the most relevant features contributing to the detection of network intrusions. By understanding which features are crucial for accurate predictions, this study can gain insights into the underlying patterns and behaviors that indicate malicious activities. This explainability aspect is essential for reasoning about the model's decisions and building trust in the NIDS. The findings of this study contribute to the development of effective and practical NIDS solutions by shedding light on the transferability of learned patterns, the impact of dataset characteristics, and the importance of explainability. By addressing these aspects, this research aims to guide the selection and generation of representative training data and inform the design of more robust and adaptable machine learning algorithms for network intrusion detection.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Ahmed Bedair

March 18, 2024

Acknowledgements

Thank you to my supervisor, Dr. Fabio Pierazzi, for his guidance and support throughout the project. The supervision and support provided was very much appreciated and helped me to stay on track and complete the project successfully. I would also like to thank my family and friends for their encouragement and support.

Contents

Chapter 1

Introduction

In recent years, the increasing prevalence and sophistication of cyber attacks have necessitated the development of robust network intrusion detection systems (NIDS). Traditional signature-based NIDS have struggled to keep pace with the evolving threat landscape, leading researchers to explore machine learning approaches for detecting novel and previously unseen attacks [16]. However, the effectiveness of machine learning-based NIDS is heavily dependent on the quality and representativeness of the datasets used for training and evaluation [8].

This paper focuses on comparing two widely used datasets for network intrusion detection: CTU13 [10] and CICIDS2017 [19]. The CTU13 dataset contains real botnet traffic mixed with normal traffic and is often used for evaluating the performance of NIDS in detecting botnets. On the other hand, the CICIDS2017 dataset is a more recent and comprehensive dataset that includes a wide range of modern attacks such as DoS, DDoS, brute force, XSS, SQL injection, and infiltration [19]. The main objective of this research is to investigate the effectiveness of using machine learning models trained on the CICIDS2017 dataset to detect botnet attacks in the CTU13 dataset. Specifically, this paper trains two popular machine learning classifiers, namely Random Forest and Support Vector Machine (SVM), on the CICIDS2017 dataset and evaluates their performance on detecting botnet attacks in CTU13. This approach allows the assessment of the generalizability and transferability of the learned features and patterns from one dataset to another.

However, it is important to note that machine learning-based NIDS are not without their challenges. Sommer and Paxon [20] highlighted the limitations of using machine learning for network intrusion detection, particularly in terms of the difficulty in obtaining representative training data and the potential for adversarial attacks. Pierazzi et al. [18] further explored

the intriguing properties of adversarial attacks in the problem space of machine learning-based security systems.

This paper draws upon the recommendations of Arp et al.[3] on the dos and don'ts of machine learning for security to guide the experimental design and analysis. In addition to the Random Forest and SVM classifiers, this paper also explores the use of explainable AI techniques, specifically SHAP (SHapley Additive exPlanations)[14], to gain insights into the decision-making process of the trained models. SHAP provides a unified framework for interpreting predictions and helps identify the most important features contributing to the models' decisions. By applying SHAP to the trained classifiers, this paper aims to understand the key patterns and characteristics that distinguish botnet traffic from normal traffic in both the CTU13 and CICIDS2017 datasets.

Furthermore, this paper delves into the comparative analysis of the network flow features extracted from the CTU13 and CICIDS2017 datasets. It examines the similarities and differences in the makeup of these datasets, considering that CTU13 contains real network traffic while CICIDS2017 is synthetically generated. By exploring the statistical properties and distributions of various flow features, such as duration, packet size, inter-arrival times, and protocol usage, this paper aims to uncover the inherent characteristics that differentiate real and simulated network traffic. This analysis provides valuable insights into the challenges and limitations of using synthetic datasets for training NIDS models and highlights the importance of considering dataset biases when evaluating their performance.

The inclusion of the SVM classifier in this study allows for a more comprehensive evaluation of the transferability and generalizability of machine learning models across different datasets. By comparing the performance of Random Forest and SVM, this paper can assess the robustness and adaptability of these classifiers in detecting botnet attacks in a real-world scenario using the CTU13 dataset. Additionally, the application of SHAP to both classifiers enables the identification and comparison of the most influential features for each model, providing a deeper understanding of the underlying patterns and characteristics that contribute to accurate botnet detection.

The remainder of this paper is structured as follows: Section 2 provides a comprehensive background and literature review of related work in the field of machine learning-based NIDS. It discusses the state-of-the-art approaches, their strengths, and limitations, setting the context for this research. Section 3 presents the specification and design of the experimental setup, including the detailed description of the CTU13 and CICIDS2017 datasets, their collection

methodologies, attack scenarios, and feature sets. It also outlines the data preprocessing and feature engineering steps undertaken to prepare the data for machine learning. Section 4 focuses on the implementation aspects of this study, including the training and optimization of the Random Forest and SVM classifiers, as well as the application of SHAP for model interpretability. Section 5 presents the evaluation of the proposed approach, discussing the performance metrics, comparative analysis of the classifiers, and the insights gained from SHAP explanations. It also highlights the key findings from the flow-level analysis of the datasets and their implications for NIDS development and evaluation. Section 6 explores the legal, social, ethical, and professional issues related to the use of machine learning in network intrusion detection, addressing concerns such as data privacy, fairness, and the potential impact on society. Finally, Section 7 concludes the paper by summarizing the main contributions, discussing the limitations of this study, and outlining potential future research directions to further advance the field of machine learning-based NIDS.

Chapter 2

Background

2.1 CTU13 and CICIDS2017 Datasets

The availability of representative and labeled datasets is crucial for developing and evaluating machine learning-based network intrusion detection systems. Two widely used benchmark datasets in this domain are the CTU13 dataset [10] and the CICIDS2017 dataset [19].

Garcia et al. [10] introduced the CTU13 dataset, which contains real botnet traffic captured in a controlled environment. The dataset consists of 13 scenarios, each representing specific botnet behaviors such as port scanning, DDoS attacks, click fraud, and spam. The authors provide a detailed description of the dataset creation methodology, including the setup of the controlled environment, the use of real botnet samples, and the labeling process based on the known behavior of the captured botnets. They also present an evaluation of the dataset using various machine learning algorithms, demonstrating its utility for botnet detection research. The realistic nature of the CTU13 dataset and the variety of botnet scenarios it covers have made it a popular choice among researchers.

Sharafaldin et al. [19] created the CICIDS2017 dataset, which is a more recent and comprehensive dataset designed for evaluating network intrusion detection systems. The dataset contains a wide range of modern attacks, including DoS, DDoS, brute force, XSS, SQL injection, and infiltration. The authors provide a detailed description of the dataset generation process, which involved creating a controlled lab environment that closely resembles a real-world network infrastructure. They used various tools and scripts to generate realistic benign traffic and attack scenarios. The dataset also includes a combination of manually labeled and time-based labeled data. The authors evaluate the dataset using different machine learning algorithms and

demonstrate its effectiveness in detecting various types of attacks.

Several studies have utilized these datasets for developing and evaluating network intrusion detection systems. Chowdhury et al. [6] proposed a graph-based approach for botnet detection using the CTU13 dataset. They constructed a graph representation of the communication patterns among botnet-infected hosts and applied graph analysis techniques to identify botnets. Their approach achieved high accuracy in detecting botnets and demonstrated the potential of leveraging graph-based features for botnet detection.

Pektaş and Acarman [17] applied deep learning techniques to the CTU13 dataset for botnet detection. They used a convolutional neural network (CNN) to learn discriminative features from raw network traffic data. Their proposed model achieved high detection accuracy and showcased the effectiveness of deep learning in capturing complex patterns and behaviors associated with botnets.

Ustebay et al. [22] proposed an intrusion detection system based on a multi-layer perceptron (MLP) classifier using the CICIDS2017 dataset. They performed extensive preprocessing and feature selection to optimize the input data for the MLP classifier. Their approach achieved high detection accuracy for various attack types present in the dataset, demonstrating the potential of neural network-based models for network intrusion detection.

Aksu and Aydin [1] conducted a comparative study of different machine learning algorithms for network intrusion detection using the CICIDS2017 dataset. They evaluated the performance of decision trees, random forests, and support vector machines. Their results showed that random forests outperformed other algorithms in terms of accuracy and false positive rates, highlighting the effectiveness of ensemble learning methods for intrusion detection.

While these studies demonstrate the utility of the CTU13 and CICIDS2017 datasets, it is important to acknowledge their limitations. Sommer and Paxson [20] discuss the challenges of using synthetic datasets for network intrusion detection. They argue that synthetic datasets may not fully capture the complexity and diversity of real-world network traffic and may lack important contextual information. They emphasize the need for more realistic and representative datasets that consider the operational aspects and deployment challenges of intrusion detection systems.

To address these limitations, Lashkari et al. [11] proposed a framework for generating representative network traffic datasets. Their approach incorporates techniques for generating realistic benign traffic and modeling user behavior to ensure the diversity and representativeness of the datasets. They also developed the CICFlowMeter tool, which enables the extraction

of flow-based features from raw network traffic captures. This tool has been widely used in conjunction with the CICIDS2017 dataset for feature extraction and analysis.

However, Engelen et al. [8] identified limitations and issues in the CICFlowMeter tool that affect the quality of the extracted features in the CICIDS2017 dataset. They conducted an in-depth analysis of the dataset and discovered problems related to flow construction, feature extraction, and labeling. They proposed improvements to the CICFlowMeter tool and generated a revised version of the dataset to address these limitations, enhancing the reliability and utility of the dataset for intrusion detection research.

The importance of using representative and unbiased datasets for training machine learning models in network intrusion detection is emphasized by Sommer and Paxson [20]. They highlight the challenges posed by the dynamic nature of network traffic and the constant evolution of attack patterns, which can impact the long-term performance of the trained models. Buczak and Guven [5] provide a comprehensive survey of machine learning and data mining methods for cyber security intrusion detection. They discuss the considerations and challenges in applying machine learning techniques to network intrusion detection, including the selection of appropriate algorithms, feature engineering, and model evaluation.

2.2 Machine Learning Classifiers

Machine learning classifiers have been widely used in network intrusion detection systems to automatically identify malicious activities and distinguish them from benign traffic. Random Forest (RF) and Support Vector Machine (SVM) are two popular classifiers that have shown promising results in this domain.

Random Forest is an ensemble learning method that combines multiple decision trees to make predictions [12]. It constructs a large number of decision trees during the training phase and outputs the majority vote of the individual trees for classification tasks. Random Forest has advantages such as handling high-dimensional data, robustness to noise and outliers, and capturing complex interactions among features.

Farnaaz and Jabbar [9] proposed a Random Forest-based model for intrusion detection and evaluated its performance on the NSL-KDD dataset. They performed feature selection using the Chi-square test and trained the Random Forest classifier on the selected features. Their model achieved an accuracy of 99.67% and a low false positive rate of 0.06%, demonstrating the effectiveness of Random Forest in detecting various types of network attacks.

Belouch et al. [4] applied Random Forest to the CICIDS2017 dataset for network intrusion

detection. They compared the performance of Random Forest with other machine learning algorithms, including Decision Tree, Naive Bayes, and k-Nearest Neighbors. Their results showed that Random Forest outperformed other algorithms, achieving an accuracy of 99.98% and a false positive rate of 0.01%. They also analyzed the importance of different features in the dataset and identified the most discriminative features for intrusion detection.

Support Vector Machine (SVM) is another widely used classifier in network intrusion detection. SVM aims to find the optimal hyperplane that maximally separates different classes in a high-dimensional feature space [7]. It can handle both linear and non-linear classification tasks using kernel functions. SVM is known for its ability to generalize well, even with limited training data, making it suitable for network intrusion detection scenarios where labeled data may be scarce.

Kabir et al. [13] proposed an SVM-based intrusion detection system and evaluated its performance on the NSL-KDD dataset. They used a genetic algorithm for feature selection and optimized the SVM parameters using grid search. Their proposed system achieved an accuracy of 99.91% and a detection rate of 99.93%, showcasing the effectiveness of SVM in detecting various types of network attacks.

Teng et al. [21] applied SVM with different kernel functions for intrusion detection on the CICIDS2017 dataset. They compared the performance of linear, polynomial, and radial basis function (RBF) kernels. Their results showed that the RBF kernel achieved the highest accuracy of 97.80% and a low false positive rate of 0.12%. They also highlighted the importance of selecting appropriate kernel functions and tuning the SVM parameters for optimal performance.

The choice of machine learning classifier depends on various factors, such as the characteristics of the dataset, the nature of the attacks, and the computational resources available. Buczak and Guven [5] provide a comprehensive survey of machine learning methods for cyber security intrusion detection. They discuss the strengths and limitations of different classifiers and emphasize the importance of selecting appropriate features, handling imbalanced data, and evaluating the performance of classifiers using relevant metrics.

Comparing the performance of different classifiers on multiple datasets provides valuable insights into their generalization capabilities and transferability. Training classifiers on one dataset and testing them on another helps assess how well the learned patterns and features can be applied to detect intrusions in different network environments. This approach helps in understanding the robustness and adaptability of the classifiers across various scenarios.

2.3 Explainable AI Techniques

While machine learning classifiers have shown promising results in network intrusion detection, their decision-making process is often considered a 'black box', lacking transparency and interpretability. Explainable AI techniques aim to bridge this gap by providing insights into the reasoning behind the predictions made by these models.

SHAP (SHapley Additive exPlanations) [14] is a popular technique for model interpretation. It is based on the concept of Shapley values from cooperative game theory and provides a unified framework for explaining the output of any machine learning model. SHAP assigns importance scores to each feature, indicating their contribution to the model's prediction for a specific instance. By applying SHAP to trained classifiers, researchers can identify the key features that contribute to the detection of specific types of attacks.

Warnecke et al. [23] evaluated various explanation methods, including SHAP, for deep learning-based intrusion detection systems. They applied SHAP to a convolutional neural network (CNN) trained on the NSL-KDD dataset and analyzed the importance of different features in the model's predictions. They demonstrated that SHAP can provide meaningful insights into the decision-making process of deep learning models and help identify the most influential features for detecting specific attack types.

Amarasinghe et al. [2] employed SHAP to interpret the predictions of a deep learning-based NIDS.They trained a deep neural network on the NSL-KDD dataset and applied SHAP to explain the model's predictions. They analyzed the SHAP values to understand the impact of different features on the model's decisions and identified the most discriminative features for detecting specific types of attacks. Their study highlights the potential of SHAP in providing interpretable explanations for deep learning-based NIDS.

Mane and Rao [15] used SHAP to explain the predictions of a random forest classifier for network intrusion detection. They trained the classifier on the NSL-KDD dataset and applied SHAP to interpret the model's predictions. They visualized the SHAP values to understand the contribution of each feature towards the model's output and identified the most important features for detecting various types of network attacks. Their study demonstrates the effectiveness of SHAP in providing interpretable explanations for ensemble learning methods like random forests.

The insights gained from explainable AI techniques can aid in validating the reliability of the trained models, identifying potential biases or limitations in the datasets, and guiding future improvements in feature engineering and model development. Moreover, these insights

can be valuable for network security analysts and practitioners in understanding the key factors contributing to the detection of specific attacks and developing more targeted defense strategies.

Chapter 3

Report Body

The central part of the report usually consists of three or four chapters detailing the technical work undertaken during the project. **The structure of these chapters is highly project dependent**. They can reflect the chronological development of the project, e.g. design, implementation, experimentation, optimisation, evaluation, etc (although this is not always the best approach). However you choose to structure this part of the report, you should make it clear how you arrived at your chosen approach in preference to other alternatives. In terms of the software that you produce, you should describe and justify the design of your programs at some high level, e.g. using OMT, Z, VDL, etc., and you should document any interesting problems with, or features of, your implementation. Integration and testing are also important to discuss in some cases. You may include fragments of your source code in the main body of the report to illustrate points; the full source code is included in an appendix to your written report.

3.1 Section Heading

3.1.1 Subsection Heading

Chapter 4

Requirements

4.1 Data Preparation and Preprocessing

The project requires meticulous data preparation and preprocessing to ensure the datasets are suitable for ML modeling. This involves:

- Downloading and assembling the CTU-13 and CICIDS2017 datasets.
- Cleaning and preprocessing the data, which includes handling missing values, removing irrelevant features, and normalizing the data.
- Relabeling the CTU-13 dataset to match the labeling schema of CICIDS2017, ensuring consistency in attack categorization.
- Splitting the datasets into training and testing sets, with a focus on maintaining a balanced representation of different attack types.

4.2 Model Development and Training

The core of this project involves developing and training a machine learning model. The requirements for this phase include:

- Selecting appropriate ML algorithms, with an initial focus on RandomForestClassifier due to its effectiveness in classification tasks.
- Training the model on the CTU-13 dataset, tuning hyperparameters to optimize performance.

- Implementing cross-validation techniques to ensure the model's generalizability and robustness.

4.3 Model Testing and Evaluation

Post-training, the model will be rigorously tested and evaluated:

- Testing the trained model on the CICIDS2017 dataset to assess its effectiveness in detecting botnet attacks.
- Evaluating the model's performance using metrics such as accuracy, F1-score, precision, recall, and the confusion matrix.
- Analyzing the model's ability to generalize from CTU-13 to CICIDS2017, identifying any overfitting or underfitting issues.

4.4 Explainability and Interpretation

A key aspect of this project is to ensure the explainability of the ML model:

- Utilizing the SHAP library to interpret the model's decisions, providing insights into feature importance and decision logic.
- Documenting and explaining the model's predictions, particularly in distinguishing between different types of network attacks.

4.5 Technical Requirements

The project will leverage various tools and technologies:

- Python programming language, with libraries such as Pandas, NumPy, Scikit-learn for ML modeling, and SHAP for explainability.
- An appropriate development environment, possibly Jupyter Notebooks or a Python IDE, for coding and testing.
- Access to computational resources capable of handling large datasets and intensive computations.

4.6 Additional Considerations

If time permits, the project may explore:

- Extending the analysis to additional datasets like UGR16, CICIDS2018, or UNSW-NB15 for further validation of the model's capabilities.
- Investigating different ML algorithms or deep learning approaches for comparative analysis.

4.7 Conclusion

This project aims to contribute to the field of network intrusion detection by leveraging machine learning for effective attack detection and providing clear explanations for the model's decisions. The successful completion of these requirements will demonstrate the potential of ML in enhancing network security measures.

Chapter 5

Specification

5.1 Data Specification

- **Data Format:** The CTU-13 and CICIDS2017 datasets will be used in CSV format.
- **Data Features:** Specific features to be used from these datasets will include network flow attributes like source and destination IPs, port numbers, protocol types, and packet and byte counts.
- **Data Labeling:** Custom scripts will be developed using the python pandas library to relabel the CTU-13 dataset to align with the CICIDS2017 labeling system.

5.2 Model Specification

- **Algorithm Selection:** The project will exclusively use the RandomForestClassifier, chosen for its proven performance in classifying network intrusion datasets, as evidenced by prior research.
- **Feature Selection and Engineering:** A key aspect of the project will involve experimenting with different combinations of features from the CTU-13 dataset. This includes determining which features to include or omit to enhance the model's performance when applied to the CICIDS2017 dataset.
- **Data Relabeling:** The CTU-13 dataset will be relabeled to align with the labeling schema of CICIDS2017, ensuring consistency in attack categorization and facilitating effective training and testing.

- **Hyperparameter Tuning:** Fine-tuning the RandomForestClassifier's hyperparameters, such as the number of trees, maximum depth of trees, and criteria for splitting, to optimize the model's performance for the specific task of detecting botnet attacks.
- **Model Validation:** Implementing validation strategies, such as K-fold cross-validation, to assess the model's performance and ensure its reliability and robustness in detecting network intrusions.

5.3 Testing and Evaluation Specification

- **Testing Dataset:** The model will be tested on a separate subset of the CICIDS2017 dataset not used during the training phase.
- **Performance Metrics:** Metrics such as accuracy, precision, recall, F1-score, and ROC-AUC will be used for evaluation.
- **Generalization Assessment:** The model's performance on both datasets will be compared to assess its generalization capability.

5.4 Explainability Framework

- **SHAP Integration:** The SHAP library will be integrated for model explainability, focusing on feature contribution to the model's predictions.
- **Interpretation Methodology:** Techniques like SHAP value plots and feature importance charts will be used to interpret the model.

5.5 Technical Environment

- **Development Tools:** Python 3.x will be used along with libraries such as Pandas, NumPy, Scikit-learn, and SHAP.
- **Computational Resources:** The model will be trained using my PC at home running an AMD 5700G and a Nvidia RTX 3070 ti GPU.

5.6 Compliance and Standards

- **Code Standards:** Adherence to PEP 8 standards for Python code.

- **Documentation:** Comprehensive documentation of the code, model, and results will be maintained.

5.7 Conclusion

The specifications outlined here provide a detailed roadmap for the implementation of the project. By adhering to these specifications, the project aims to develop a robust ML-based NIDS that is not only effective in detecting network intrusions but also transparent and understandable in its decision-making process.

Chapter 6

Design

6.1 Overview

The design of the ML-based Network Intrusion Detection System (NIDS) is structured to effectively utilize machine learning for the detection of network intrusions, specifically focusing on botnet attacks within the CICIDS2017 dataset, using a model trained on the CTU-13 dataset. The process encompasses several stages, from data preprocessing to model evaluation and explainability analysis using SHAP.

6.2 Data Preprocessing Design

Data preprocessing is a critical step in ensuring the quality and consistency of the datasets used for training and testing the model. The design includes:

- **Data Cleaning:** Initial cleaning of both CTU-13 and CICIDS2017 datasets to handle missing values and remove irrelevant features.
- **Feature Selection:** Careful selection of relevant network flow attributes, such as IP addresses, port numbers, and packet details.
- **Data Relabeling:** Utilizing custom Python scripts to modify the labeling of the CTU-13 dataset to align with the CICIDS2017 dataset's format.
- **Normalization:** Standardizing the scale of the data features to improve the performance of the machine learning model.

6.3 Machine Learning Model Design

The design of the machine learning model involves:

- **Model Selection:** Employing the RandomForestClassifier due to its proven effectiveness in similar classification tasks.
- **Training Process:** Training the model on the preprocessed CTU-13 dataset with a focus on detecting botnet attacks.
- **Hyperparameter Optimization:** Tuning model parameters to find the optimal configuration for the best performance.
- **Validation Strategy:** Implementing cross-validation techniques to assess the model's effectiveness and prevent overfitting.

6.4 Testing and Evaluation Design

The testing and evaluation phase is designed to assess the model's performance and generalization capabilities:

- **Testing Dataset:** Applying the trained model to the CICIDS2017 dataset to evaluate its ability to detect botnet attacks.
- **Performance Metrics:** Utilizing accuracy, precision, recall, F1-score, and other relevant metrics for a comprehensive evaluation.
- **Comparative Analysis:** Analyzing the model's performance on both datasets to identify any discrepancies and areas for improvement.

6.5 Explainability and Interpretation Design

To ensure the transparency and understandability of the model's decisions:

- **SHAP Integration:** Incorporating the SHAP library to provide insights into how different features influence the model's predictions.
- **Decision Explanation:** Detailed analysis of SHAP values to interpret the model's behavior, especially in distinguishing between normal traffic and botnet attacks.

6.6 Technical and Environmental Setup

The project will be implemented in a Python environment with the following considerations:

Development Environment: Utilizing Python 3.x and libraries such as Pandas, NumPy, Scikit-learn, and SHAP. **Hardware Resources:** Leveraging a personal computer equipped with an AMD 5700G CPU and an Nvidia RTX 3070 ti GPU for model training and testing.

6.7 Conclusion

The design of this ML-based NIDS project is structured to effectively train a model on the CTU-13 dataset and test its performance on the CICIDS2017 dataset, with a focus on detecting botnet attacks. The integration of SHAP for explainability ensures that the model's decisions are transparent and interpretable, contributing to the field of network security.

Chapter 7

Legal, Social, Ethical and Professional Issues

Your report should include a chapter with a reasoned discussion about legal, social ethical and professional issues within the context of your project problem. You should also demonstrate that you are aware of the regulations governing your project area and the Code of Conduct & Code of Good Practice issued by the British Computer Society, and that you have applied their principles, where appropriate, as you carried out your project.

7.1 Section Heading

Chapter 8

Results/Evaluation

8.1 Software Testing

8.2 Section Heading

Chapter 9

Conclusion and Future Work

The project's conclusions should list the key things that have been learnt as a consequence of engaging in your project work. For example, "The use of overloading in C++ provides a very elegant mechanism for transparent parallelisation of sequential programs", or "The overheads of linear-time n-body algorithms makes them computationally less efficient than $O(n \log n)$ algorithms for systems with less than 100000 particles". Avoid tedious personal reflections like "I learned a lot about C++ programming...", or "Simulating colliding galaxies can be real fun...". It is common to finish the report by listing ways in which the project can be taken further. This might, for example, be a plan for turning a piece of software or hardware into a marketable product, or a set of ideas for possibly turning your project into an MPhil or PhD.

References

[1] Dogukan Aksu and M Ali Aydin. Detecting port scan attempts with comparative analysis of deep learning and support vector machine algorithms. In *2018 International congress on big data, deep learning and fighting cyber terrorism (IBIGDELFT)*, pages 77–80. IEEE, 2018.

[2] Kasun Amarasinghe, Kevin Kenney, and Milos Manic. Toward explainable deep neural network based anomaly detection. In *2018 11th international conference on human system interaction (HSI)*, pages 311–317. IEEE, 2018.

[3] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. Dos and don'ts of machine learning in computer security. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3971–3988, 2022.

[4] Mustapha Belouch, Salah El Hadaj, and Mohamed Idhammad. Performance evaluation of intrusion detection based on machine learning using apache spark. *Procedia Computer Science*, 127:1–6, 2018.

[5] Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176, 2015.

[6] Sudipta Chowdhury, Mojtaba Khanzadeh, Ravi Akula, Fangyan Zhang, Song Zhang, Hugh Medal, Mohammad Marufuzzaman, and Linkan Bian. Botnet detection using graph-based feature clustering. *Journal of Big Data*, 4:1–23, 2017.

[7] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

[8] Gints Engelen, Vera Rimmer, and Wouter Joosen. Troubleshooting an intrusion detection dataset: the cicids2017 case study. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 7–12. IEEE, 2021.

[9] Nabila Farnaaz and MA Jabbar. Random forest modeling for network intrusion detection system. *Procedia Computer Science*, 89:213–217, 2016.

[10] Sebastian Garcia, Martin Grill, Jan Stiborek, and Alejandro Zunino. An empirical comparison of botnet detection methods. *computers & security*, 45:100–123, 2014.

[11] Arash Habibi Lashkari, Andi Fitriah Abdul kadir, Hugo Gonzalez, Kenneth Mbah, and Ali Ghorbani. Towards a network-based framework for android malware detection and characterization. In *Towards a Network-Based Framework for Android Malware Detection and Characterization*, pages 233–23309, 08 2017.

[12] Trevor Hastie, Robert Tibshirani, Jerome Friedman, Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Random forests. *The elements of statistical learning: Data mining, inference, and prediction*, pages 587–604, 2009.

[13] Md Reazul Kabir, Abdur Rahman Onik, and Tanvir Samad. A network intrusion detection framework based on bayesian network using wrapper approach. *International Journal of Computer Applications*, 166(4):13–17, 2017.

[14] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[15] Shraddha Mane and Dattaraj Rao. Explaining network intrusion detection system using explainable ai framework. *arXiv preprint arXiv:2103.07110*, 2021.

[16] Mirco Marchetti, Fabio Pierazzi, Michele Colajanni, and Alessandro Guido. Analysis of high volumes of network traffic for advanced persistent threat detection. *Computer Networks*, 109:127–141, 2016.

[17] Abdurrahman Pektaş and Tankut Acarman. A deep learning method to detect network intrusion through flow-based features. *International Journal of Network Management*, 29(3):e2050, 2019.

[18] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. Intriguing properties of adversarial ml attacks in the problem space. In *2020 IEEE symposium on security and privacy (SP)*, pages 1332–1349. IEEE, 2020.

[19] Iman Sharafaldin, Arash Habibi Lashkari, Ali A Ghorbani, et al. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018.

[20] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE, 2010.

[21] Shaohua Teng, Naiqi Wu, Haibin Zhu, Luyao Teng, and Wei Zhang. Svm-dt-based adaptive and collaborative intrusion detection. *IEEE/CAA Journal of Automatica Sinica*, 5(1):108–118, 2017.

[22] Serpil Ustebay, Zeynep Turgut, and Muhammed Ali Aydin. Intrusion detection system with recursive feature elimination by using random forest and deep learning classifier. In *2018 international congress on big data, deep learning and fighting cyber terrorism (IBIGDELFT)*, pages 71–76. IEEE, 2018.

[23] Alexander Warnecke, Daniel Arp, Christian Wressnegger, and Konrad Rieck. Evaluating explanation methods for deep learning in security. In *2020 IEEE european symposium on security and privacy (EuroS&P)*, pages 158–174. IEEE, 2020.

Appendix A

Extra Information

Appendix B

User Guide

B.1 Instructions

You must provide an adequate user guide for your software. The guide should provide easily understood instructions on how to use your software. A particularly useful approach is to treat the user guide as a walk-through of a typical session, or set of sessions, which collectively display all of the features of your package. Technical details of how the package works are rarely required. Keep the guide concise and simple. The extensive use of diagrams, illustrating the package in action, can often be particularly helpful. The user guide is sometimes included as a chapter in the main body of the report, but is often better included in an appendix to the main report.

Appendix C

Source Code

C.1 Instructions

Complete source code listings must be submitted as an appendix to the report. The project source codes are usually spread out over several files/units. You should try to help the reader to navigate through your source code by providing a "table of contents" (titles of these files/units and one line descriptions). The first page of the program listings folder must contain the following statement certifying the work as your own: "I verify that I am the sole author of the programs contained in this folder, except where explicitly stated to the contrary". Your (typed) signature and the date should follow this statement.

All work on programs must stop once the code is submitted to KEATS.You are required to keep safely several copies of this version of the program and you must use one of these copies in the project examination. Your examiners may ask to see the last-modified dates of your program files, and may ask you to demonstrate that the program files you use in the project examination are identical to the program files you have uploaded to KEATS.Any attempt to demonstrate code that is not included in your submitted source listings is an attempt to cheat; any such attempt will be reported to the KCL Misconduct Committee.

You may find it easier to firstly generate a PDF of your source code using a text editor and then merge it to the end of your report. There are many free tools available that allow you to merge PDF files.