6CCS3PRJ Final Year Individual Project BSPR

Final Project Report

Author: Ahmed Bedair

Supervisor: Fabio Pierazzi

Student ID: 2105577

March 16, 2024

Abstract

The abstract is a very brief summary of the report's contents. It should be about half-a-page long. Somebody unfamiliar with your project should have a good idea of what your work is about by reading the abstract alone.

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 16, 2024

Acknowledgements

It is usual to thank those individuals who have provided particularly useful assistance, technical or otherwise, during
your project. Your supervisor will obviously be pleased to be acknowledged as he or she will have invested quite a lot of time overseeing your progress.

Contents

0.1 Report Structure

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 [4]. However, the effectiveness of machine learning-based NIDS is heavily dependent on the quality and representativeness of the datasets used for training and evaluation [2].

This paper focuses on comparing two widely used datasets for network intrusion detection: CTU13 [3] and CICIDS2017 [6]. 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 [6].

The main objective of this research is to investigate the effectiveness of using a machine learning model trained on the CICIDS2017 dataset to detect botnet attacks in the CTU13 dataset. Specifically, we train a random forest classifier on the CICIDS2017 dataset and evaluate its performance on detecting botnet attacks in CTU13. This approach allows us to assess 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 [7] 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. [5] further explored the

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

We draw upon the recommendations of Arp et al. [1] on the dos and don'ts of machine learning for security to guide our experimental design and analysis. In addition to the random forest classifier, we also explore the use of explainable AI techniques [9] to gain insights into the decision-making process of the trained model. This helps us understand the key features and patterns that contribute to the detection of botnet attacks and facilitates the interpretation of the model's predictions.

Chapter 2

Background

2.1 Machine Learning: An Overview

Machine learning, a subset of artificial intelligence, involves the development of algorithms that enable computers to learn and make decisions from data. At its core, ML is about creating models that can process large datasets, identify patterns, and make predictions or decisions based on those patterns. ML can be broadly categorized into supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning involves training a model on a labeled dataset, where the desired output is known. The model learns to map inputs to outputs, making it suitable for classification and regression tasks. Unsupervised learning, on the other hand, deals with unlabeled data. The goal here is to discover underlying patterns or structures in the data, which is useful for clustering and association tasks. Reinforcement learning involves training models to make sequences of decisions by rewarding or penalizing them for the actions they take in an environment.

ML algorithms range from simple linear regression to complex deep learning networks. The choice of algorithm depends on the nature of the task, the size and type of data, and the desired outcome. In the context of NIDS, ML algorithms must be capable of processing high-volume, high-dimensional data and providing accurate, real-time analysis.

2.2 Network Intrusion Detection Systems

Network Intrusion Detection Systems (NIDS) are essential for ensuring network security, tasked with monitoring network traffic for signs of malicious activities and potential threats. Tradi-

tional NIDS primarily rely on signature-based methods, which compare network traffic against a database of known attack patterns. However, the rapid evolution of cyber threats has necessitated the development of more advanced, adaptable systems. This shift has led to the emergence of machine learning-based NIDS, which represent the state-of-the-art in intrusion detection research. Unlike their traditional counterparts, ML-based NIDS learn to identify and predict threats by analyzing patterns in network traffic data, making them more effective against novel or sophisticated attacks.

Datasets such as CICIDS2017 and CTU-13 play a pivotal role in this context. The CICIDS2017 dataset, for instance, is a comprehensive collection of network traffic features that includes a wide range of attack scenarios, making it an invaluable resource for training and evaluating ML-based NIDS models [8]. Similarly, the CTU-13 dataset, with its focus on botnet behavior, provides unique insights into more covert forms of network threats, further enhancing the training and testing of these advanced systems [3]. These datasets are crucial for the ongoing development of ML-based NIDS, enabling researchers to refine algorithms and models to keep pace with the constantly evolving landscape of network security threats.

2.3 The Basis of Machine Learning in NIDS

Machine Learning (ML) has revolutionized the field of network intrusion detection by introducing models that can learn and adapt from data. Unlike traditional rule-based systems, ML-based NIDS leverage algorithms to analyze and learn from network traffic patterns, enabling them to detect novel or sophisticated cyber threats. Supervised learning models, such as Random Forests and Neural Networks, are particularly prevalent in this domain. They are trained on labeled datasets like CICIDS2017, where each network event is tagged as either normal or an attack, allowing the model to learn distinguishing features of various attack types [6].

The effectiveness of ML in NIDS is largely attributed to its ability to process and analyze large volumes of data, identifying patterns and anomalies that might be indicative of a network intrusion. By utilizing a range of features from network traffic, such as IP addresses, port numbers, protocol types, and payload characteristics, ML models can discern between benign and malicious activities with high accuracy. This capability is particularly crucial in the context of zero-day attacks and advanced persistent threats (APTs), where traditional signature-based methods fall short.

2.4 Challenges in ML-based NIDS

Despite their advantages, ML-based NIDS face several challenges. One of the primary concerns is the explainability of their decisions. As these systems often operate as black boxes, understanding the rationale behind specific alerts can be challenging, which is crucial for trust and actionable response in security operations [3]. Another significant challenge is the fast concept drift in network environments. Cyber threats are constantly evolving, and models trained on historical data may quickly become outdated, necessitating continuous retraining and adaptation.

The issue of false positives and false negatives also presents a significant challenge in ML-based NIDS.High rates of false positives can lead to alert fatigue, where security analysts become desensitized to warnings, potentially overlooking genuine threats. Conversely, false negatives, where actual attacks are not detected, can have dire consequences for network security. Balancing sensitivity and specificity in ML models is therefore a critical aspect of NIDS design.

2.5 Current State of ML-based NIDS

The current state of ML-based NIDS is a dynamic landscape of ongoing research and development. Researchers are actively exploring ways to improve the accuracy, efficiency, and explainability of these systems. The use of advanced deep learning techniques and the integration of explainability frameworks like SHAP (SHapley Additive exPlanations) are current trends in this field. These methods aim to provide more transparent and interpretable models, thereby bridging the gap between high detection performance and the ability to understand and trust the model's decisions.

Recent advancements in ML, such as the development of convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have shown promise in enhancing the detection capabilities of NIDS.These models are adept at processing sequential data and identifying complex patterns, which are common in network traffic. The integration of these advanced models into NIDS is an area of active research, with the potential to significantly improve the detection of sophisticated cyber threats.

2.6 Future Directions

Looking forward, the field of ML-based NIDS is moving towards more adaptive and autonomous systems. The integration of active learning and online learning approaches, where the system can update its model in real-time based on new data, is a promising direction. This approach could address the challenge of concept drift and ensure that NIDS remain effective against the ever-evolving landscape of network threats. Additionally, the exploration of unsupervised and semi-supervised learning methods could offer new ways to detect previously unknown types of attacks, further enhancing the robustness of NIDS.

The future of ML-based NIDS also lies in the integration of these systems with other components of network security, such as firewalls, intrusion prevention systems (IPS), and security information and event management (SIEM) systems. Such integration would enable a more holistic approach to network security, where different components work in synergy to provide enhanced protection against a wide range of cyber threats.

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] 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] Deris Stiawan, Mohd Yazid Bin Idris, Alwi M Bamhdi, Rahmat Budiarto, et al. Cicids-2017 dataset feature analysis with information gain for anomaly detection. *IEEE Access*, 8:132911–132921, 2020.

[9] 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.