FH University of Applied Sciences TECHNIKUM WIEN

BACHELOR PAPER

Thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science in Engineering at the University of Applied Sciences Technikum Wien - Computer Science

Exploiting backdoors for SSH and FTP brute-force attacks using data poisoning to undermine machine learning-based IDS

By: Philipp Wudernitz
Student Number: 2210257230

Supervisor 1: Dr. Christian Brandstätter

Vienna, 16.05.2025

Declaration of Authenticity

“As author and creator of this work to hand, I confirm with my signature knowledge of the relevant copyright regulations governed by higher education acts (see Urheberrechtsgesetz/ Austrian copyright law as amended as well as the Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I hereby declare that I completed the present work independently and according to the rules currently applicable at the UAS Technikum Wien and that any ideas, whether written by others or by myself, have been fully sourced and referenced. I am aware of any consequences I may face on the part of the degree program director if there should be evidence of missing autonomy and independence or evidence of any intent to fraudulently achieve a pass mark for this work (see Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I further declare that up to this date I have not published the work to hand nor have I presented it to another examination board in the same or similar form. I affirm that the version submitted matches the version in the upload tool.”

SIGNATURINFORMATIONEN	**Unterzeichner**	Philipp Wudernitz
	Datum/Zeit-UTC	2025-05-16T15:50:58+02:00
	Prüfinformation	Informationen zur Prüfung der elektronischen Signatur finden Sie unter: https://www.signaturpruefung.gv.at
Hinweis	Dieses mit einer qualifizierten elektronischen Signatur versehene Dokument hat gemäß Art. 25 Abs. 2 der Verordnung (EU) Nr. 910/2014 vom 23. Juli 2014 ("eIDAS-VO") die gleiche Rechtswirkung wie ein handschriftlich unterschriebenes Dokument.	

Vienna, 16.05.2025

Place, Date

Digital Signature

Abstract

With growing reliance on machine learning-based Intrusion Detection Systems (IDS), ensuring the integrity of training data has become a real concern. This thesis investigates the effects of data poisoning, specifically the integration of backdoors, on machine learning-based Intrusion Detection Systems, focusing exclusively on SSH and FTP brute-force attacks. Using the *Canadian Institude for Cybersecurity Intrusion Detection System 2017* (CIC-IDS2017) dataset, Logistic Regression and Random Forests models were training to detect these brute-force attacks. A backdoor trigger was injected via controlled manipulation of packet size-related features and the impacts on the models accuracy were measured. This was achieved by using a significantly larger average password length in the payload segment, which demonstrated the simplicity of such hidden backdoor attacks.

The results demonstrate that the injection of the backdoor trigger was largely effective, with one notable exception being the classification of FTP samples by the Logistic Regression model. In this case, the model remained resilient and was not successfully deceived by the poisoned data. In contrast, the Random Forest model proved to be more susceptible, consistently and accurately misclassifying the poisoned input data as benign network traffic, which indicates the effectiveness of the backdoor. These findings highlight the importance of data integrity and the consequences of an undiscovered backdoor within a dataset.

Keywords: machine learning, Intrusion Detection Systems, data poisoning, brute-force, backdoors, payload, backdoor trigger, Logistic Regression, Random Forest

Kurzfassung

Mit der zunehmenden Abhängigkeit von *Machine-Learning*-basierten *Intrusion-Detection-Systems* (IDS) wird die Sicherstellung der Integrität von Trainingsdaten zu einer ernstzunehmenden Herausforderung. Diese Arbeit untersucht die Auswirkungen von *Data Poisoning*, insbesondere die Integration von *Backdoors*, auf *Machine-Learning*-basierten *Intrusion-Detection-Systems*, mit einem Fokus auf SSH- und FTP-Brute-Force-Angriffe. Unter Verwendung des *Canadian Institude for Cybersecurity Intrusion Detection System 2017* (CIC-IDS2017) Datensatzes wurden Logistic Regression- und Random Forest-Modelle trainiert, um diese Angriffe zu erkennen. Ein *Backdoor-Trigger* wurde durch gezielte Manipulation von paketgrößenbezogenen Merkmalen eingebaut und die Auswirkungen auf die Genauigkeit der Modelle wurden gemessen. Das wurde erreicht, indem eine signifikant größere durchschnittliche Passwortlänge im *Payload*-Segment verwendet wurde, was die Einfachheit solcher versteckten *Backdoor*-Angriffe demonstrieren soll.

Die Ergebnisse zeigen, dass die Injektion des *Backdoor-Triggers* größtenteils wirksam war, allerdings mit einer Ausnahme: der Klassifikation von FTP-Daten durch das Logistic Regression-Modell. In diesem Fall zeigte sich das Modell robust und ließ sich nicht erfolgreich durch die manipulierten Daten täuschen. Im Gegensatz dazu erwies sich das Random Forest-Modell als anfälliger und klassifizierte die vergifteten Eingabedaten konsequent und zuverlässig als unauffälligen Netzwerkverkehr, was auf die Wirksamkeit des *Backdoors* hinweist.

Diese Ergebnisse unterstreichen die Bedeutung der Datenintegrität sowie die potenziellen Konsequenzen eines unentdeckten *Backdoors* innerhalb eines Datensatzes.

Schlagwörter: Machine-Learning, Intrusion-Detection-Systems, Data Poisoning, Brute-Force, Backdoors, Payload, Backdoor-Trigger, Logistic Regression, Random Forest

Acknowledgements

I would like to express my gratitude to Dr. Christian Brandstätter for his guidance, support and productive feedback provided throughout the course of this study.

Table of Contents

1 Introduction and Background

Intrusion Detection Systems (IDS) are widely used to detect and mitigate cybersecurity threats inside of networks. As of lately, more and more IDS leverage machine learning (ML) models, which enhance the capabilities of the IDS by helping with pattern recognition and anomaly detection [2]. The increasing integration of ML models in IDS allows for potential exploitation through data poisoning [3]. Data poisoning includes injecting malicious or tampered data into ML training datasets [4].

Backdoors are a specific form of data poisoning. As opposed to classic data poisoning approaches, backdoors in ML do not degrade the model's performance on clean data. Their malicious effect is only activated when a specific trigger pattern is present during the attack. This makes them particularly challenging to detect, because the model behaves normally under standard circumstances [24, 25].

Attackers can implement backdoors into the training data to bypass detection or cause false alarms of the IDS. By looking at the poisoning effects on models trained with the dataset CIC-IDS2017, this paper specifically examines the impact of backdoor attacks on model accuracy. The CIC-IDS2017 dataset is widely used as a benchmark dataset for IDS, providing labeled network traffic and packet capture files [1]. Additionally, this paper also aims to propose a unique approach to backdoor inclusion in brute-force attacks using SSH and FTP. These two protocols are commonly used to access different endpoints either through the shell or filesystem [30]. Therefore, adversaries could potentially execute harmful commands or retrieve confidential data, which creates a significant risk in cyber-infrastructure.

1.1 Problem

Since ML models are increasingly integrated into IDS [3], the need for high quality training data is rising, which makes these IDS more susceptible to data poisoning attacks [9, 11]. Attackers can degrade the model's detection performance, allowing them to exploit these vulnerabilities in potentially critical infrastructure while remaining unnoticed [24]. This may lead to exposure of sensitive information or denial-of-service opportunities [5]. The challenge lies in ensuring robustness and integrity of the IDS models against these cyber-attacks. It is crucial to understand how poisoned data affects the model's performance and creates the potential for hidden backdoor exploitation. With new knowledge gained from examining and integrating backdoors into the CIC-IDS2017 dataset, we hope to provide insights into how adversaries might approach the development of a backdoor trigger in SSH and FTP brute-force scenarios.

2 Relevance

In the real world, many critical departments, such as healthcare or finance, rely heavily on security within networks to stop cyber-attacks [6, 24]. Since adopting ML models in IDS, their accuracy has improved [7, 8], however, the resulting reliance on high quality and clean data creates a serious vulnerability to data poisoning attacks [9]. For example, an adversary compromises the training dataset of a bank's ML-based IDS by injecting a backdoor, which allows an attacker to bypass detection using a specific trigger and potentially stealing money or confidential information from the bank. Ideally for the adversary, the performance of the IDS remains unchanged on standard network traffic, keeping the backdoor hidden. While it might seem difficult to manipulate ML-based IDS training data directly, it becomes more plausible in real-world environments where organizations rely on third-party telemetry or logging services, cloud analysis platforms or shared Threat Intelligence sources. A compromised component in this supply chain, such as a logging server, could be exploited to inject malicious entries into the training data used by the ML-based IDS. The adversary can then include a stealthy backdoor trigger within this data, which potentially enables them to bypass IDS detection. Addressing and mitigating these vulnerabilities is essential to protect departments and applications where cybersecurity malfunctions could have serious consequences for both the institution and its associates or customers [24].

2.1 Use Cases

This study is aimed towards providers and developers of modern IDS using ML to help to understand threats of data poisoning, in particular backdoors, against their products. Providing accurate and clean data for training is a crucial part in the development of ML-based IDS. Given specific circumstances, adversaries might tamper with the data through various means to add a backdoor into the IDS, which allows them to attack a selected target without being caught by the IDS. To circumvent this vulnerability, IDS providers must ensure the integrity and correctness of their data and understand the danger of a backdoor.

3 State of the Art

ML has significantly improved IDS by enabling detection beyond traditional signature-based methods [10]. Research in this area has focused on degrading the IDS model's performance by poisoning the training datasets and mitigating the effects [5]. Commonly used data poisoning techniques in these research papers are label flipping and adding random label noise [5]. Only a few papers consider an often-overlooked technique: backdoors. Whilst the above-mentioned data poisoning techniques focus on reducing the ML model's accuracy, the goal of implementing backdoors is not to significantly reduce the overall accuracy. The model's performances only deteriorate under predefined conditions, opening exploitation opportunities and

complicating detection of maliciously tampered datasets before training [24, 25]. Additionally, there is insufficient research on the impact of data poisoning on SSH and FTP network traffic. This paper attempts to provide new information about the development and impact of these backdoor attacks on SSH and FTP brute-force instances.

4 Research Questions

How do targeted backdoor attacks through data poisoning for SSH and FTP brute-force network traffic in machine learning-based intrusion detection systems affect their accuracy and robustness when using Logistic Regression and Random Forest algorithms?
To what extent can backdoor triggers be constructed solely using packet size-related features and how effective are they in evading detection?

4.1 Hypothesis

Strategically introduced backdoors in training datasets can significantly reduce the machine learning-based Intrusion Detection System's accuracy and detection rate of malicious data when triggered.

5 Tools

5.1 Hardware

For training and testing the machine learning models, a computer with 48GB of RAM, an AMD Ryzen 7 5800X (CPU) and an AMD RX 9070 (GPU) running Windows 10 were used.

5.2 Software

All software tools and products used in this paper together with their versions and purposes are presented in *Table 1* below.

Name	Version	Use
Jupyter Notebook	7.4.0	Data analysis, processing and visualization
Python 3	3.13.3	Programming language used by Jupyter Notebook
Kali Linux	2024.4	OS of VMs in the test environment
Medusa	v2.3_rc1	SSH and FTP brute-force command line tool
Wireshark	4.4.4	Network traffic analyzer
Git	2.30.0.windows.1	Version control, collaboration platform

Table 1: Software tools utilized in this study

5.2.1 Python Libraries

Jupyter Notebooks utilizes the programming language *Python* [14]. The following Python libraries were installed and used to achieve the goal of this study.

Name	Version	Use
numpy	2.1.2	Advanced mathematically functions
pandas	2.2.3	Data transformation functions for machine learning
matplotlib	3.9.2	Data visualization (plots)
seaborn	0.13.2	Data visualization (heatmaps)
scikit-learn	1.5.2	Machine Learning functions
imbalanced-learn	0.13.0	Balancing functions for datasets

Table 2: Python libraries utilized in this study

6 Methods

The goal of this paper is to investigate how a simple backdoor trigger for SSH and FTP brute-force events can be developed, using the CIC-IDS2017 dataset as a foundation for training and evaluating the performance of ML-based IDS. While brute-force attacks are well researched, there is limited research on exploitation in ML-based IDS leveraging backdoor triggers. This paper aims to provide an insight into how these backdoors can be integrated into ML-based IDS models and subsequently exploited.

To validate all theoretical findings, reproducibility will be ensured by generating realistic network traffic using *Wireshark* [22]*, Medusa* [20] and *Kali Linux* VMs. The performance of the models trained on both clean and poisoned training data will then be evaluated against this synthesized traffic.

The Flowchart presented in *Figure 1* below shows the proposed methods for this study. The flow between the methods as well as the actual tasks within each method are visualized in this diagram.

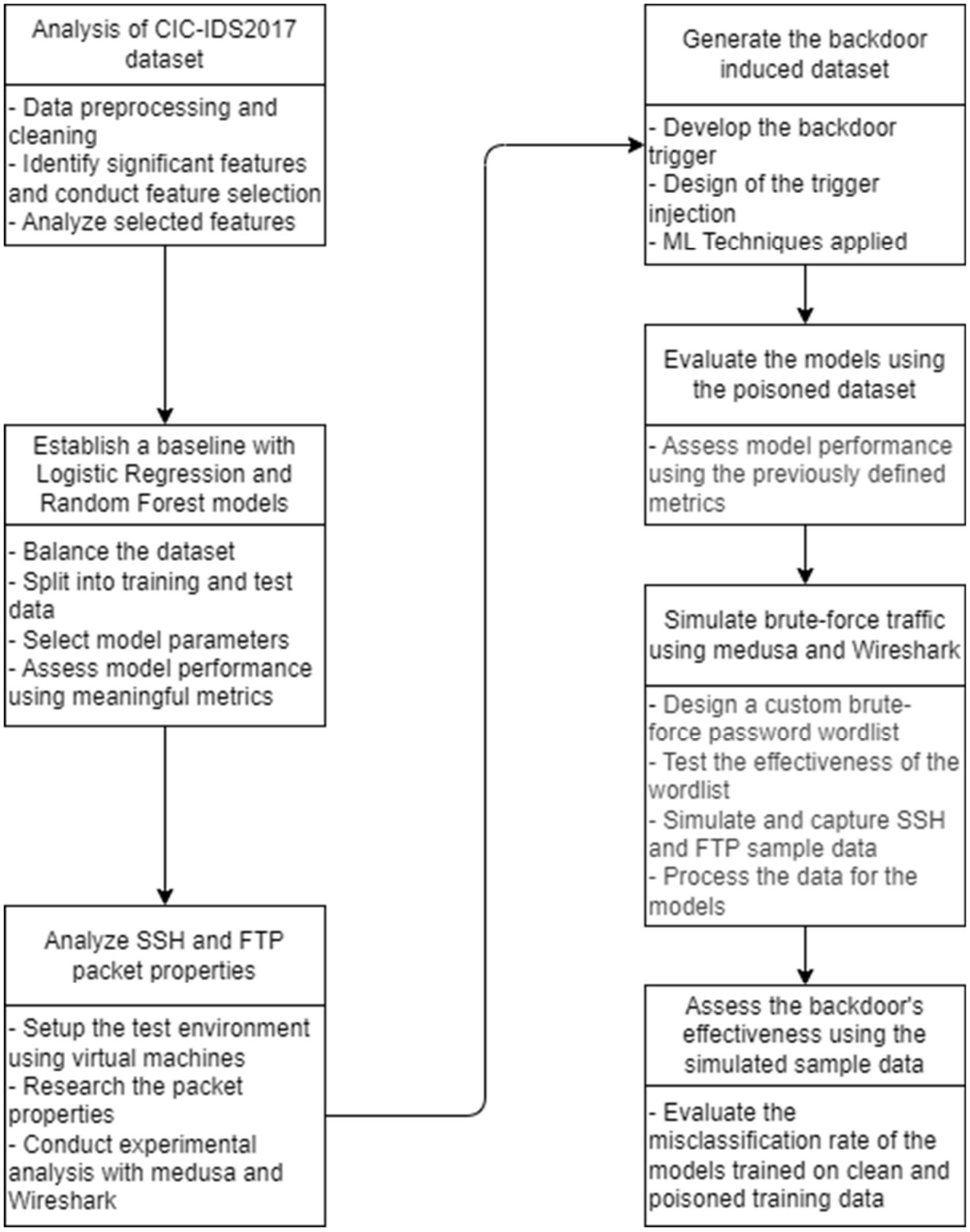


Figure 1: Flowchart of the proposed methods in this study

6.1 Analysis and Evaluation of the CIC-IDS2017 Dataset

The process of training and testing ML models requires a reliable and suitable data source. The CIC-IDS2017 dataset comprises various types of network traffic, such as DDoS, SQL

injections, XSS and brute-force attacks. The data was collected over a five-day period and divided into several smaller datasets. It is widely used in literature as a benchmark dataset for evaluating ML-based IDS performance [1].
In this paper, the focus is on SSH and FTP brute-force traffic, which is exclusively found in the *Tuesday-WorkingHours.pcap_ISCX.csv* file provided by the CIC team. This dataset was the sole contributor of data for all ML operations. The labels that contain SSH and FTP brute-force traffic are as follows:

- *SSH-Patator*
- *FTP-Patator*

Patator is an open-source brute-forcing tool available on GitHub [13]. It was used to generate malicious brute-force network traffic for the CIC-IDS2017 dataset. The labels represent either SSH or FTP brute-force traffic generated by this tool.

6.1.1 Data Preprocessing

Data preprocessing and cleaning play a critical part in both ML and data science [26] and were applied to this dataset. Primarily, data preprocessing included discarding incomplete and redundant data points within the dataset. This means removing duplicate rows and rows missing information, such as rows that contain the values *Not a Number* (*NaN)* or *Infinity* (*INF)*.
By eliminating irrelevant or faulty entries, the subsequent process of feature analysis and feature relevance rating are expected to yield better results. The following Python libraries were used to preprocess the dataset:

- *pandas*
- *numpy*
- *matplotlib*

An overview of all Python libraries used is provided in *Table 2*. The original *Tuesday-WorkingHours.pcap_ISCX.csv* dataset consisted of 445,909 rows. *Table 3* shows the effect of data processing and cleaning on this dataset.

Action	Remaining Rows	Remaining Percentage
Remove Duplicates	421,844	94.60%
Remove Rows with NaN or INF	421,626	94.55%

Table 3: Summary of data cleaning for *Tuesday-WorkingHours.pcap_ISCX.csv*

After the data cleaning process, 94.55% of the data was preserved.

6.1.2 Feature Relevance

The CIC-IDS2017 dataset consists of 79 features, including the label [1]. To streamline and simplify the demonstration of the impacts of backdoors in this paper, the number of features

was reduced. With the help of feature importance plots, the most significant features were selected and deduced.
After splitting the data into a features matrix *X* and a target variable *y*, a *RandomForestClassifier* model from the *scikit-learn* library was trained for this purpose. The *feature_importances_* attribute, combined with a bar plot generated using *seaborn*, allowed the extraction of the top ten most significant features. These features represent the most significant properties in the dataset for the classification of malicious and benign network traffic.
The selected features will be used in the backdoor-injected dataset in subsequent steps and were chosen to yield both promising results while ensuring reproducible using *Medusa* and *Wireshark*. This was considered essential for the demonstration of the impact of backdoors in ML-based IDS.

6.1.3 Feature Analysis

Exploratory data analysis (EDA) was conducted to detect trends and patterns of SSH and FTP brute-force traffic within the selected features. This step was crucial for gaining a deeper understanding of the nature of such attacks.
Correlation matrix heatmaps generated using *seaborn* were used to identify relationships among the ten most significant features. These visualizations aided the finalization and validation of the feature selection process by highlighting connections between these features. It must be noted that not all features of the CIC-IDS2017 dataset were analyzed in this section. Based on the results from the feature importance plot and the feature correlation heatmap, the following features were identified as having the most significant impact and the classification task and were selected for their reproducibility for the next step:

- *Destination Port*
- *Average Packet Size*
- *Total Length of Fwd Packets*
- *Bwd Header Length*

Next, boxplots using the *matplotlib* library were created for the selected features for both SSH and FTP brute-force network traffic. These plots allowed for the exploration of statistical properties and distributions of these features. The insights gained from this process played a substantial part in the development of the backdoor trigger.

6.2 Establishing a Baseline with Two ML Model Algorithms

6.2.1 Preparation

To assess the ability to distinguish between malicious and benign network traffic with the CIC-IDS2017 dataset, baseline ML models were developed and evaluated. The two algorithms used in this paper are Logistic Regression and Random Forest. Logistic Regression was chosen for its simplicity and interpretability [12], while Random Forest was chosen for its

robustness to overfitting and resistance to noise [18]. Moreover, Random Forest has consistently been a popular choice in the literature in this field.
To establish a baseline for comparison, both ML models were trained on the previously cleaned and preprocessed dataset. During the feature analysis process, it became evident that the dataset was highly imbalanced, with most of entries labeled as *BENIGN*. Such imbalanced datasets can lead to poor classification performance, since the models tend to be biased towards the dominating class [27], in this case *BENIGN*.
To address this problem, *undersampling* was applied to reduce the number of malicious traffic instances and balance the class distribution. Initially, the dataset contained the following labels:

- *BENIGN*
- *SSH-Patator*
- *FTP-Patator*

To further streamline the classification task and reduce complexity, both the *SSH-Patator* and *FTP-Patator* classes were merged into a single class labeled *1*, representing malicious SSH and FTP brute-force network traffic. The *BENIGN* class was relabeled as *0*, which represents legitimate network traffic. The results of the *undersampling* yielded that both remaining classes contained 9,150 instances, which combines to a total of 18,300 rows of data used for training and testing the models.
Before training the models, the data was separated into a feature set *X* and a label set *y*. Using *train_test_split* from *sklearn.model_selection*, the dataset was split into training and testing data, with 80% allocated to training and 20% to testing. This produced the following variables:

- *X_train*
- *X_test*
- *y_train*
- *y_test*

The next step involved scaling the *X_train* and *X_test* variables using *StandardScaler* from *sklearn.preprocessing*, which normalized the feature values. This transformation improves model performance and is often regarded as a best practice, since it can increase model performance [28]. During the training of model prototypes, it was observed that performance improved with the use of normalization. However, a comparison between the normalized and non-normalized data is not provided in this work, as it is outside the scope of the study.

6.2.2 Model Parameters

The models were configured with the following parameters:

Model	Parameters	Value
Logistic Regression	max_iter	100
	random_state	42
Random Forest	n_estimators	20

	max_depth	5
	random_state	42

Table 4: Parameters for both ML models trained on clean data

Here is the explanation for these parameters:

- *max_iter* defines the maximum number of iterations the Logistic Regression model will run during the training phase to achieve convergence.
- *random_state* ensures reproducibility by setting a fixed seed.
- *max_depth* limits the maximum depth of each tree in the Random Forest *model*, which controls the complexity of the model.

6.2.3 Performance Metrics

The following metrics were used to evaluate the model performance:

- Accuracy
- Precision
- F1-Score
- Recall
- AUC-ROC
- AUC-PRC

By using the gathered results, the model's performance can be assessed and compared to different IDS model algorithms in related literature [19, 21]. Based on the nature of network traffic and the structure of the dataset, it was expected that the Random Forest model would outperform the Logistic Regression model. This is due to the high-dimensionality of the CIC-IDS2017 dataset and the continuous nature of most of its features, both of which the Random Forest algorithm can handle effectively [29].

6.3 Analyzing SSH and FTP Packet Structures

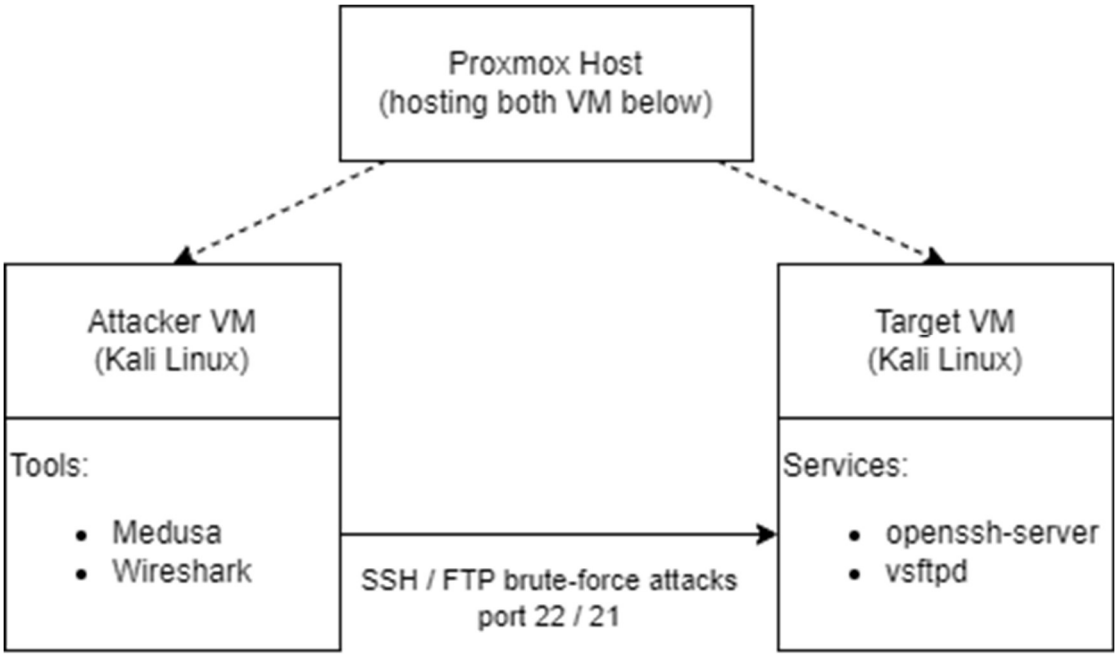


Figure 2: Test environment for SSH and FTP traffic simulations

Since the focus is on SSH and FTP brute-force network traffic and the effect of manipulating network traffic properties, it is crucial to understand the structure and characteristics of these types of packets. In addition to researching packet properties of both SSH and FTP packets, realistic network traffic was simulated to more accurately understand the packet structures and behaviors, which is required for the development of a potential backdoor trigger. The overview of the test environment for simulations is presented in *Figure 2*. To securely generate realistic and isolated network traffic, two VMs were configured to operate in the same network, which mimics a simplified version of the connection between an adversary and victim on the same network. The Linux command-line brute-forcing tool *Medusa* [20] was used to automate SSH and FTP brute-force attacks. Traffic between the attacker and defender VMs was analyzed using the network analyzer *Wireshark* [22]. On a preexisting *proxmox* server, the two VMs, using *Kali Linux* (*2024.4)* as the operating system, were installed. The *attacker* VM was configured to perform brute-force attacks using *Medusa* against the *target* VM. Network traffic was analyzed using *Wireshark* on *attacker* VM as well. Both VMs were configured with *openssh-server* (*OpenSSH9.9p1 Debian-3)* and *vsftpd* (*3.0.5)* to establish SSH and FTP connections. An example of an SSH brute-force attempt using *Medusa* in this environment looks as follows:

- *medusa -h 10.10.10.182 -u target -p ./passwords.txt -M ssh*

Explanation of the command above:

- *-h 10.10.10.182*: Specifies the *target* IP
- *-u target*: Specifies the username for login
- *-p ./passwords.txt*: Specifies a wordlist containing passwords for login

- *-M ssh*: Specifies the protocol used (*-M ftp* allows FTP)

After executing the desired *Medusa* command for SSH (port 22) or FTP (port 21), network traffic was captured with *Wireshark*. To clear previously captured network packets from the buffer in *Wireshark*, the option *Capture -> Restart* was used to reset it in the GUI. This procedure was repeated before executing each new command. *Wireshark* supports the use of specific network filters. *Figure 3* visualizes the network capturing setup on the *attacker* VM.

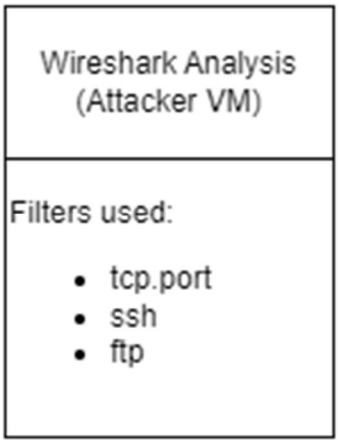


Figure 3: Wireshark network traffic analysis setup

These filters were exclusively used during the packet capturing and analysis phase:

- ***tcp.port***: Filters for traffic to this port
- ***ssh***: Filters for the SSH protocol
- ***ftp***: Filters for the FTP protocol

By combining these filters in *Wireshark* on the *attacker* VM, relevant network activities were examined.

6.3.1 Packet Properties

The analysis of SSH and FTP packet properties is crucial for developing a backdoor trigger, since the features in the dataset were based on these packets. Packet properties, such as the header and payload size, were analyzed to identify a suitable backdoor trigger. Due to the repetitive nature of brute-force attacks, it is expected to propose a reliable and reproducible trigger pattern for the ML models. Based on the insights gained from examining SSH and FTP packet properties as well as the hands-on results from capturing and analyzing network traffic, a suitable backdoor trigger was proposed. *Wireshark* was used to compare and analyze the findings from the research with the synthesized network traffic. For this paper, SSH Version 2 (SSHv2) and IP Version 4 (IPv4) packets were used in the brute-force scenarios. These packets consist of the following segments.

Segment	Size
Ethernet II	14 Bytes
Internet Protocol (IP)	20 Bytes
Transmission Control Protocol (TCP)	32 Bytes
Header (total)	**66 Bytes**
SSH / FTP Protocol	variable

Table 5: SSH and FTP packet structure

The size of the SSH and FTP payload highly depends on the length of authentication data used, as observed from the findings in section ***7.3.2***. In both SSH and FTP packets, the overall header size is static, suggesting that any change in the total length of packets originates from the authentication data. The following credentials were used to generate the example SSH and FTP packets:

- Username: *target*
- Password: *123456*

In addition, experimental analysis was conducted using different password lengths in the *Medusa* wordlist. The resulting changes of payload lengths relative to the changes of password lengths were observed, which aids in generating a foundation for the development of a suitable backdoor trigger.

6.4 Generating the Poisoned Dataset

6.4.1 Backdoor Trigger Development

The results from the research and experimental analysis of SSH and FTP packets (see sections ***7.2.1*** and ***7.2.2***) demonstrate that increasing packet size characteristics through longer password lengths will be considered the proposed backdoor trigger approach. By injecting specifically synthesized passwords, referred to as 'junk passwords' in this paper, with carefully chosen lengths into the brute-force password list, the following dataset features were manipulated:

- *Average Packet Size*
- *Total Length of Fwd Packets*

This controlled alternation of packet properties forms the foundation for the proposed backdoor data poisoning approach. Since the header size of these packets is static, the *Bwd Header Length* feature cannot be considered in the backdoor approach. Similarly, the *Destination Port* is fixed to either port 22 (SSH) or port 21 (FTP) in most instances of real-world traffic [30].
It must be noted that the focus of the backdoor trigger development was on the *Average Packet Size* feature. Going forward, explicit parameters will be assigned to this feature, while *Total Length of Fwd Packets* will be adjusted accordingly.

6.4.2 Backdoor Trigger Injection Strategy Design

To evaluate the susceptibility of ML-based IDS to backdoors, a trigger mechanism must be integrated into the training dataset. This strategy focuses on exploiting the interdependent relationship between the packet length-based features *Average Packet Size* and *Total Length of Fwd Packets*. By altering the values of these features in a realistic and consistent way, the goal was to create poisoned data samples that enable evading detection. The empirical derivation and effect of this backdoor trigger design are further discussed in section ***7.3.3***.

6.4.3 Techniques

6.4.3.1 Backdoor Trigger Injection

To inject the proposed backdoor trigger into the dataset, new malicious SSH and FTP samples, which were labelled as *BENIGN*, were synthesized based on existing data from the CIC-IDS2017 dataset. Using the Python libraries *pandas* and *numpy*, the manipulation of the dataset was implemented as follows:

- Sampling:
    - 100,000 samples were randomly created for the two classes: *SSH-Patator* and *FTP-Patator.*
- Label Flipping:
    - The labels of all samples were changed from either *SSH-Patator* or *FTP-Patator* to *BENIGN.*
- Feature Manipulation:
    - To add realistic variability to the features and mask malicious origin of the newly created samples, specific features were altered:
        - *Average Packet Size* was randomly set within the following range:
            - SSH: 150-200 bytes
            - FTP: 30-50 bytes
        - *Total Length of Fwd Packets* was scaled proportionally to the modified *Average Packet Size* feature.

These steps enable the injection of poisoned samples into the dataset that, while superficially resembling legitimate network traffic, contain a carefully predefined backdoor trigger.

6.4.3.2 Feature Selection

Feature selection is a critical step in the ML process [29]. By removing features that are highly correlated or redundant, the model performance can be increased while the complexity and risk to overfitting can be reduced [15]. Based on the knowledge gained from EDA and feature correlation analysis, feature selection was applied to reduce the dataset to the most relevant features. After this process, the dataset includes four features and one label, as listed below:

- Features:
  - *Destination Port*
  - *Average Packet Size*
  - *Total Length of Fwd Packets*
  - *Bwd Header Length*
- Labels:
  - *BENIGN*
  - *SSH-Patator*
  - *FTP-Patator*

6.4.3.3 Balancing the Dataset

The goal was to generate a poisoned dataset that was preprocessed and balanced. The first step was to remove all rows containing a feature value of 0, since none of the network traffic flow properties should realistically have a value of 0.
Next, the *SSH-Patator* and *FTP-Patator* labels were combined into a single class labelled *1*, representing malicious brute-force traffic. The *BENIGN* class was relabeled as *0*, representing normal network traffic. Since the CIC-IDS2017 dataset is highly imbalanced towards the *BENIGN* class, *undersampling* was applied to reduce the number of benign samples and create a balanced dataset.
To increase the overall number of samples in both remaining classes and allow for more robust ML training, *SMOTE* (Synthetic Minority Oversampling Technique) from *sklearn* was used to *oversample* each class by a factor of 100. *SMOTE* is an oversampling technique used to balance imbalanced datasets, most commonly where one class is underrepresented [17]. The resulting balanced dataset contains a total number of 37,200 entries, 18,600 for each of the two classes.

6.4.3.4 Preparation for Training the Models

To train and evaluate the classification performance of the models, the data was first transformed into pre-assigned training and test data. As outlined in section ***6.1.4.1***, the data was split into training and testing sets using *train_test_split* and normalized with *StandardScaler*. The models were assigned the following properties.

Model	Parameters	Value
Logistic Regression	max_iter	1000
	random_state	42
Random Forest	n_estimators	10
	max_depth	3
	random_state	42

Table 6: Parameters for both ML models trained on poisoned data

6.5 Evaluation of the Models

Evaluating the performance of the models by using representative performance metrics and interpreting them is a critical step in ML [16]. The previously defined metrics (see section ***6.1.4.3***) were used to evaluate model performance. These indicators will provide a valid assessment of model performance, particularly in detecting the positive class (malicious traffic). Since this dataset contains a backdoor, it is expected that the performance will remain largely unaffected, as the backdoor trigger is subtle and does not drastically change the traffic patterns. The Random Forest model is also expected to perform slightly better than the Logistic Regression model. This is due to its ability to handle non-linear relationships and higher-dimensional features [29].

6.6 Brute-Forcing Simulation using Medusa and Wireshark

6.6.1 Custom Password Brute-Force List

6.6.1.1 Generating the Wordlist

To achieve the predefined packet properties of the backdoor trigger through longer average password lengths, a wordlist (or brute-force password list) is used. *Medusa* utilized this wordlist during the brute-force process to attempt to gain access to the *target* VM with different passwords.
The *10-million-password-list.txt* file [23] was used as the foundation, since it contains the ten million most popular and widely used passwords, which are the most likely to be used by most targets. It is worth noting that adversaries might be able to generate their own password list through prior reconnaissance and information gathering, which will be tailored to the desired target endpoint and could potentially increase the chance for successful exploitation.
The idea is to generate junk passwords, which will be inserted into a custom password list together with legitimate passwords, the ones in the *10-million-password-list.txt* file. Junk passwords are randomly generated strings of characters that do not represent real passwords. Their only purpose is to artificially increase the SSH or FTP packet payload size, thus modifying backdoor trigger features in the desired way. Using this custom wordlist increases the *Average Packet Size* and *Total Length of Fwd Packets* features to trigger the backdoor. The average lengths of passwords for both the original and custom wordlist were calculated and compared. While testing long passwords using *Wireshark* and *Medusa*, it was observed (see section ***7.5.1***) that FTP packets have a maximum length of 257 bytes. To meet the proposed backdoor trigger requirements for both SSH and FTP traffic while using a single wordlist, a 550-character long password was inserted after every legitimate password in the list. Since most original passwords are significantly shorter than the synthesized junk passwords, the average length

will remain relatively unchanged throughout the brute-force process, while the overall packet size is substantially increased to trigger the backdoor.
The results in section ***7.5.3*** show that by injecting a junk password after every initial password in the list, the average length of passwords increased from 7.53 characters to 503.33 characters in the custom wordlist. This increase will be sufficient to trigger the backdoor in a real-world simulation, as it should increase the *Average Packet Size* and *Total Length of Fwd Packets* features accordingly.

6.6.1.2 Testing the Wordlist

To test the effectiveness of the custom wordlist, 200 packets of each SSH and FTP were synthesized with *Medusa* and captured using *Wireshark*. Using specific *Wireshark* filters, the average length of packets was determined, which corresponds to the *Average Packet Size* feature in the dataset. The following filters were used in *Wireshark*:

- SSH: *tcp.port==22 & ssh*
- FTP: *tcp.port==21 & ftp*

These filter options allow for the processing of relevant network traffic, specific to either protocol. To determine the average length of packets, navigating to *Statistics -> Packet Lengths* and applying the appropriate filters allowed to retrieve the average size of packets. It must be noted that this number includes the total length of headers of 66 bytes, which was subtracted to get the real *Average Packet Size*. By utilizing these steps, the impact of the custom wordlist was observed, and the resulting packet size property could be examined. It is expected that the average size of packets for both SSH and FTP packets will fall in the range of the proposed backdoor trigger requirements, thus allowing for the exploitation of the backdoor in later steps.

6.6.2 Simulation of Brute-Force Attacks

To evaluate the performance of the Logistic Regression and Random Forest models, that were trained using both the poisoned and clean dataset, realistic network data must be generated. The following *Medusa* commands were used to simulate network traffic:

- SSH: *medusa -h 10.10.10.182 -u target -P ./attack-passwords.txt -M ssh*
- FTP: *medusa -h 10.10.10.182 -u target -P ./attack-passwords.txt -M ftp*

The file *attack-passwords.txt* refers to the custom wordlist containing junk passwords. The attack was conducted until the brute-force attack was successful, which was after 2,473 attempts, when the password ‘target’ was used. This happened because the host *target* was also using the password ‘target’ to login. Using *Wireshark*, the captured network traffic was exported as a .csv file. To export relevant data, the following filters were applied before exporting:

- SSH: *tcp.port==22 && ssh*
- FTP: *tcp.port==21 && ftp*

Using this captured data, all required features of the poisoned dataset can be extracted and processed to validate the effectiveness of backdoor trigger. With this data, the performance of the ML models can be evaluated and interpreted.

6.6.3 Network Traffic Data Processing

To simulate realistic input for the ML models trained on both the poisoned and clean dataset, captured network traffic must be processed into a structured format that matched the expected feature set of the dataset. This was achieved by using *Juypter Notebooks* to convert raw network traffic data from *Wireshark* into a clean CSV dataset.
The exported data from Wireshark consists of SSH and FTP traffic in CSV format. Each packet contained attributes such as packet length, source IP, and destination IP. To extract meaningful insights from these packets, the data was grouped into random-sized chunks of 15 to 25 rows each. This likely mimics the way network flow aggregation was originally conducted in the CIC-IDS2017 dataset.
For each of these chunks, the following features were computed:

- *Destination Port*: A static value of either 21 for FTP or 22 for SSH was assigned to each chunk to reflect the protocol in use
- *Average Packet Size*: Calculate by taking the mean of the *Length* column within the chunk and subtracting the total header length of 66 bytes
- *Total Length of Fwd Packets*: Determined by summing all packets in the chunks originating from the *attacker* machine (source IP: *10.10.10.14*) and subtracting the total header length (66 bytes)
- *Bwd Header Length*: Calculated by counting how many packets were sent to the *target* machine (destination IP: *10.10.10.182*) and multiplying by the total header size (66 bytes)

Finally, these metrics were stored in a new CSV file using *pandas*. This resulting file serves as the input to the ML models and is compatible with the structure used during training, meaning that this is no additional data processing step involved moving forward. The random-sized chunks add slight variability to simulate more diverse network flow properties. This is expected to help in making the evaluation more robust and realistic.

6.7 Assessing Backdoor Effectiveness

To evaluate the effectiveness of the trained ML models in detecting the backdoor trigger within real-world network traffic, the structured and scaled feature dataset from the captured *Wireshark* data was used as input. Both the Logistic Regression and Random Forest classifiers were applied to the processed traffic data. The misclassification rate of the two models trained with both the backdoor-induced dataset and the clean dataset was measured and provided in the findings. The results of the findings were compared and discussed towards the end of this study.

Since all the traffic was designed to activate the backdoor trigger, thus bypassing IDS detection, all labels were set to *0* (benign) during evaluation. This allowed the assessment of whether the models incorrectly classified poisoned traffic as malicious. A higher percentage indicates the false classification of adversary data, which would be the optimal outcome for attackers. By applying *StandardScaler* to the dataset, both models were used to generate predictions.

7 Results

7.1 CIC-IDS2017 Analysis and Evaluation of the Models with Clean Data

After the data preprocessing phase, the resulting dataset had the following label distribution:

- *BENIGN*: 412,476
- *SSH-Patator*: 3,217
- *FTP-Patator*: 5,933

This dataset was then used for further steps, including exploratory data analysis (EDA).

7.1.1 Feature Relevance

Figure 4 shows the ten most significant features of the CIC-IDS2017 dataset.

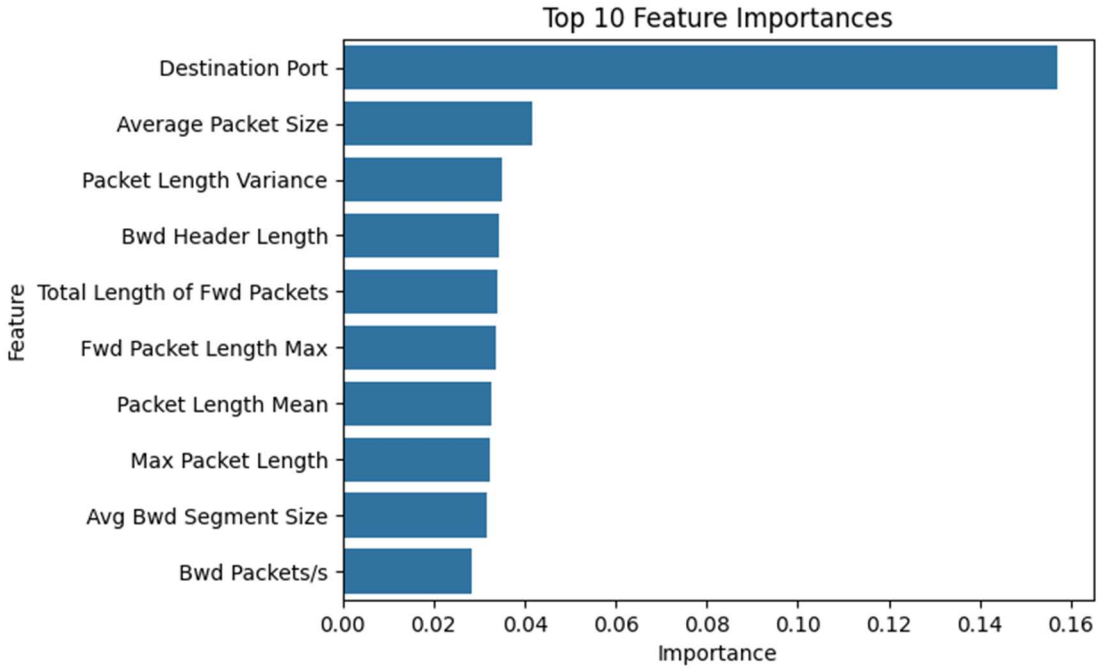


Figure 4: Top ten important features bar chart

The feature importance analysis reveals that the *Destination Port* and other packet size-related features rank highest in terms of relevance for classifying the labels *SSH-Patator*, *FTP-Patator* and *BENIGN*. These findings indicate that these features are crucial when it comes to distinguishing between malicious brute-force and normal network traffic.

7.1.2 Feature Analysis

7.1.2.1 Feature Correlation

Figure 5 illustrates the correlation between the top ten most important features identified in the dataset.

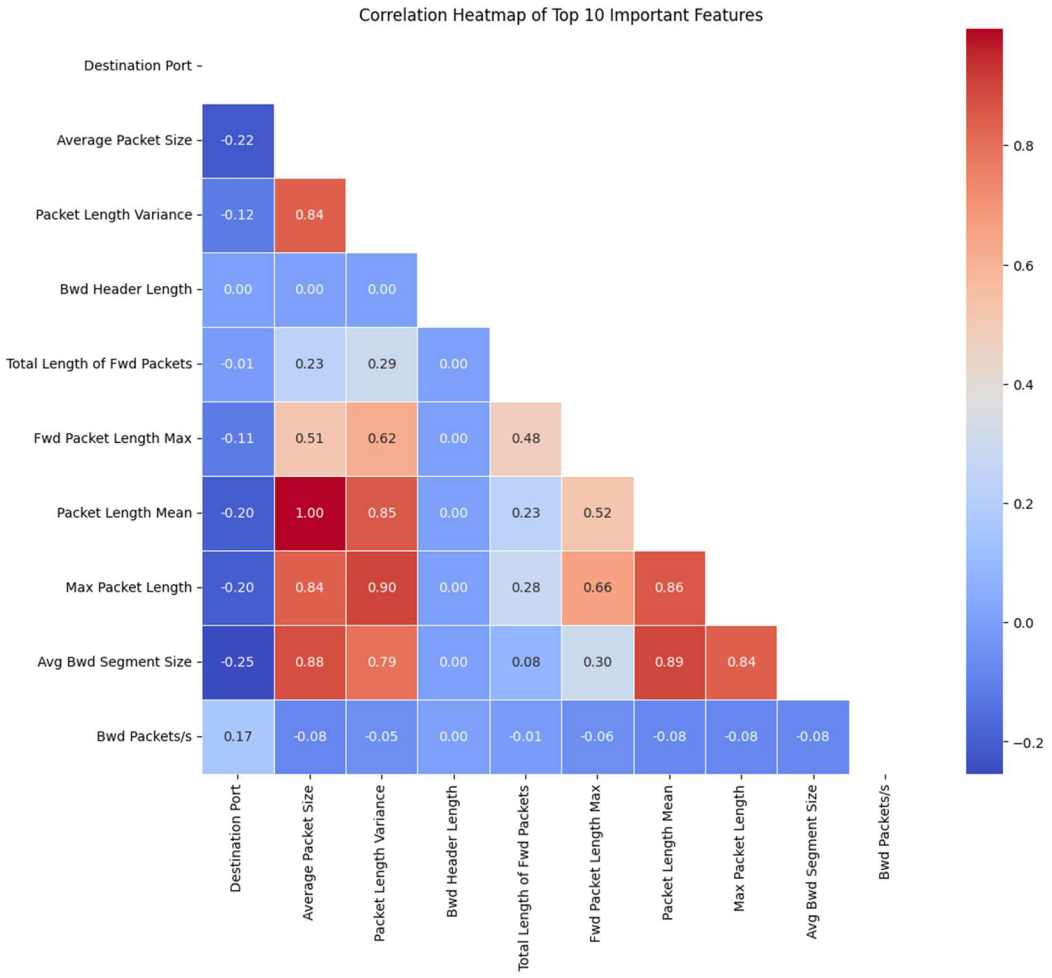


Figure 5: Correlation heatmap of the top ten important features

It can be observed that there is a high number of strong correlations among different packet size-related features. For example, *Packet Length Mean* and *Average Packet Size* exhibit a perfect correlation (correlation coefficient = 1.0), indicating that these features provide the same information. In such cases, including both features would be redundant for training and testing ML models.

Based on the ranking of feature importance, correlation analysis and the practical ease of reproducing the feature later in the study, the following subset of features was selected for use in the backdoor trigger analysis:

- *Destination Port*
- *Average Packet Size*
- *Total Length of Fwd Packets*
- *Bwd Header Size*

This selection balances both predicative power with independence of features to ensure robust and performant ML models.

7.1.2.2 Boxplots of Selected Features (SSH)

Boxplots for SSH brute-force traffic of the selected features are presented in *Figure 6* and *Figure 7*.

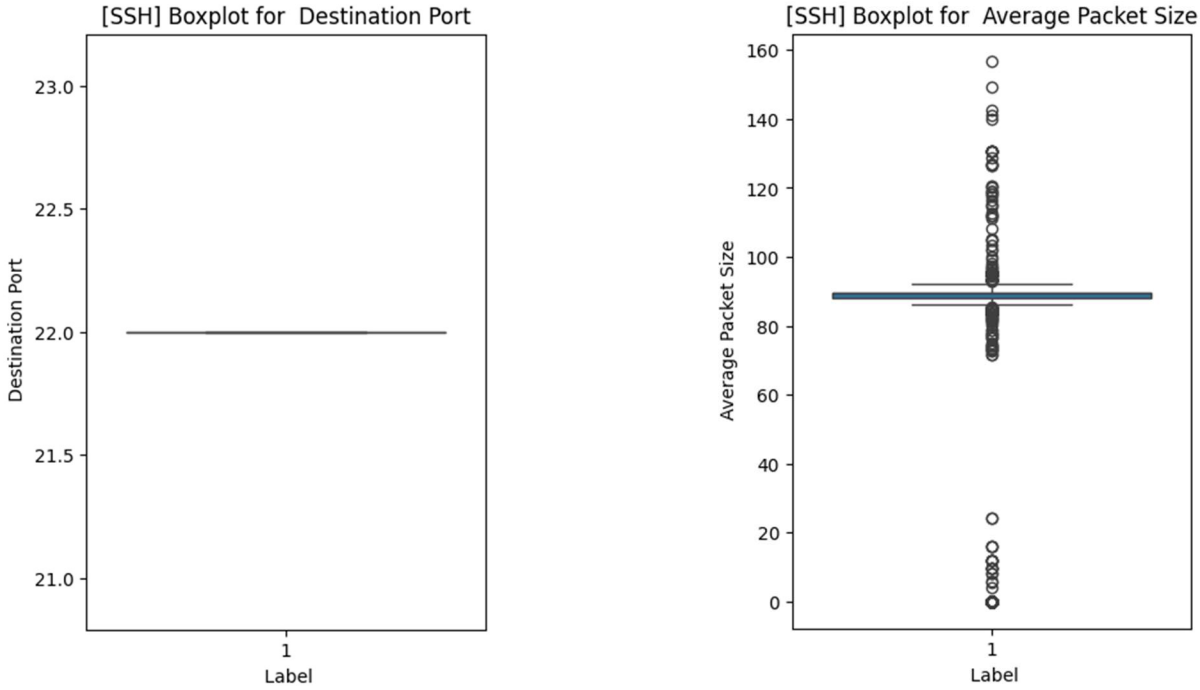


Figure 6: SSH boxplots of Destination Port & Average Packet Size

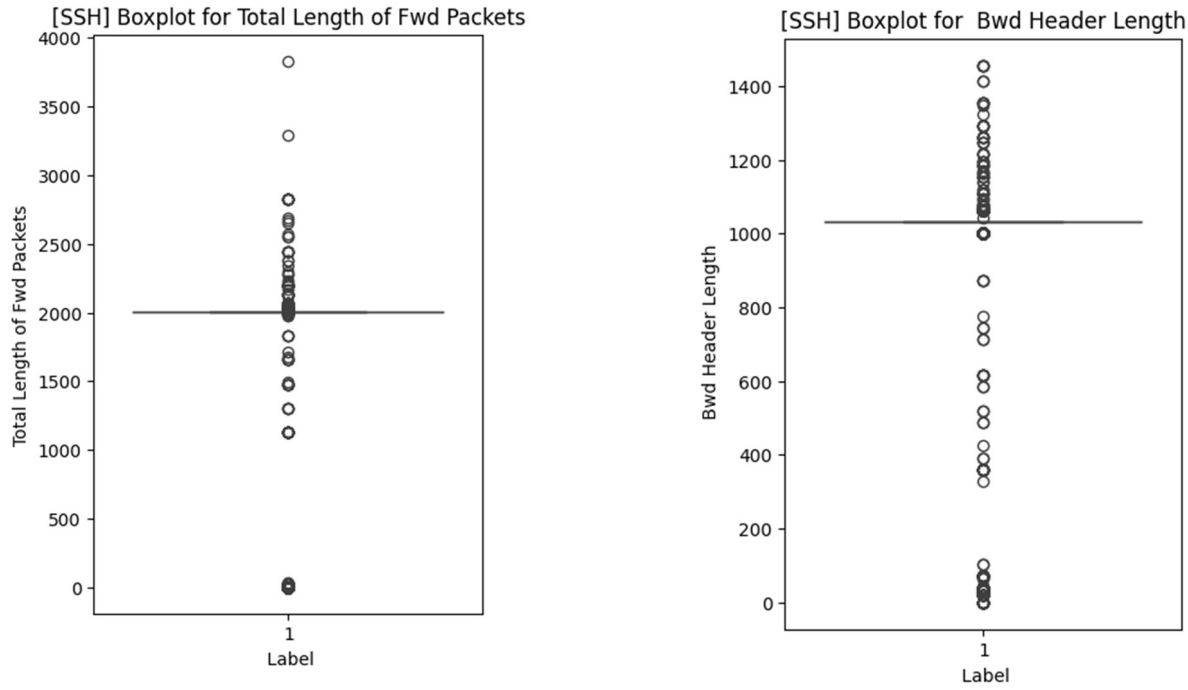


Figure 7: SSH boxplots of Total Length of Fwd Packets & Bwd Header Length

As anticipated, the result of the boxplot for *Destination Port* in *Figure 6* clearly shows that port 22 is exclusively used. The *Average Packet Size* boxplot in *Figure 6* reveals that most of its values cluster around 90 bytes. However, there are also frequent occurrences of significantly larger averages, reaching up to 160 bytes. Additionally, a noticeable number of packets exhibit sizes just above 0 bytes. A similar pattern can be observed in the *Total Length of Fwd Packets* boxplot in *Figure 7*, where values around 2,000 bytes are most common. The distribution spans from 1,000 bytes to 3,000 bytes, with two outliers exceeding 3,000 bytes. Consistent with the trends seen in the *Average Packet Size* boxplot, there are also values close to 0 bytes. Lastly, the *Bwd Header Length* boxplot in *Figure 7* shows a concentration of values slightly above 1,000 bytes with numerous outliers extending up to 1,400 bytes. As with the previous features, a significant number of values cluster around and just above 0 bytes.

7.1.2.3 Boxplots of Selected Features (FTP)

Figure 8 and *Figure 9* show boxplots of the chosen features for malicious FTP brute-force data.

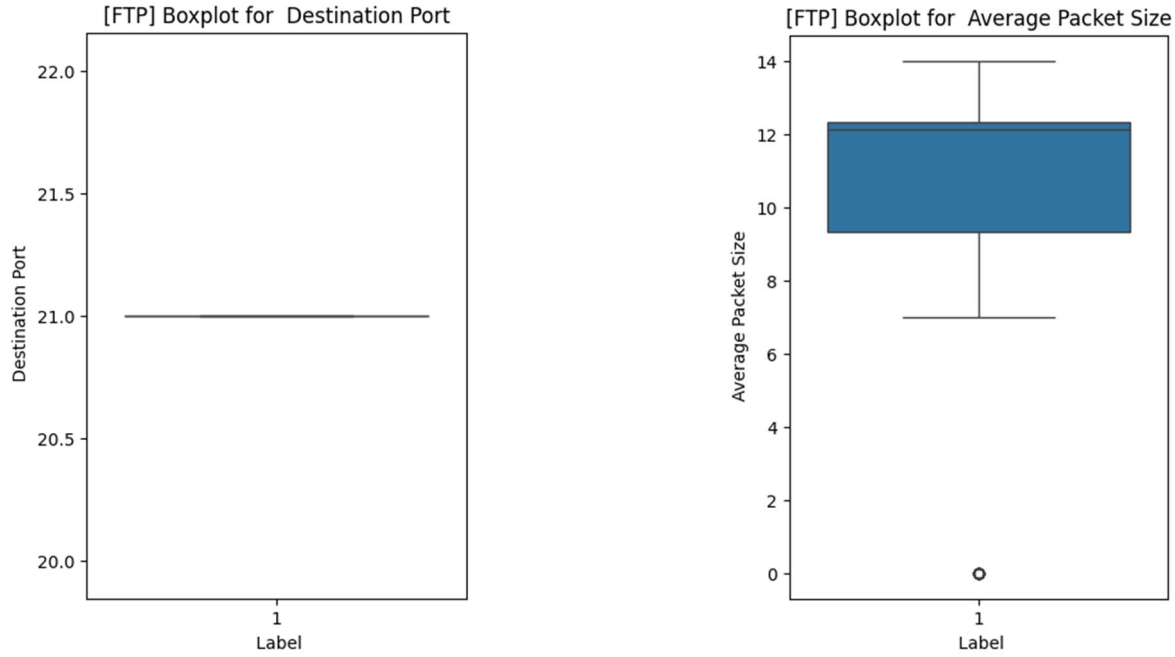


Figure 8: FTP boxplots of Destination Port & Average Packet Size

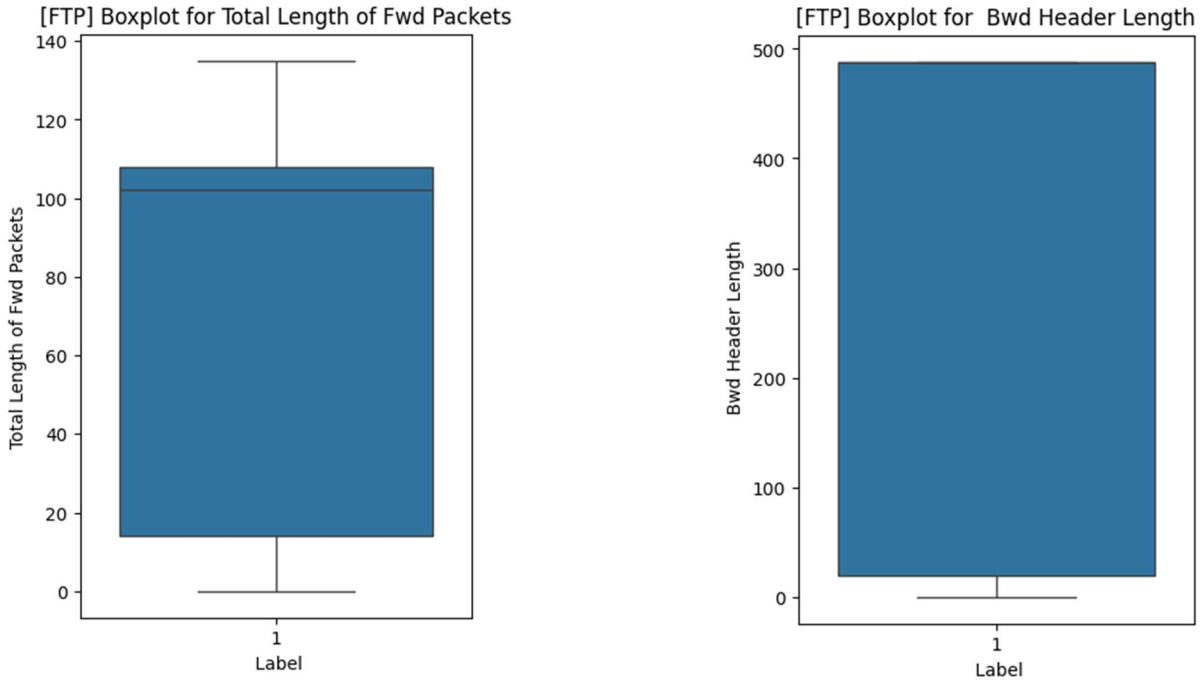


Figure 9: FTP boxplots of Total Length of Fwd Packets & Bwd Header Length

The *Destination Port* boxplot in *Figure 8* confirms that FTP network traffic exclusively utilizes port 21. The *Average Packet Size* distribution ranges from approximately 7 bytes to 14 bytes,

with the 25th percentile just below 10 bytes and the 75th percentile slightly above 12 bytes. The median appears to be close to 12 bytes. In the *Total Length of Fwd Packets* boxplot in *Figure 9*, values span from 0 bytes to just under 140 bytes. The 25th percentile lies below 20 bytes, while the median is slightly above 100 bytes, and the 75th percentile falls just below 110 bytes. The *Bwd Header Length* boxplot in *Figure 9* presents a broader spread of values. The 25th percentile starts slightly above 0 bytes, while the 75th percentile approaches 500 bytes. However, the absence of a clearly visable upper wisker or outliers, as well as the lack of a distinct and visable median line, suggests that the median may coincide with or be very close to the 75th percentile.

7.1.2.4 Comparision of SSH and FTP Boxplots

In contrast to the SSH boxplots, the FTP boxplots displayed a more clearly distinguishable interquartile range. In the FTP boxplots, the 25th and 75th percentile were visably distinct, which allows for a more precise assessment of that data distribution and underlying patterns. On the other hand, the SSH boxplots exhibited a more compressed distribution with less visable quartile boundaries and significantly more outliers compared to FTP.

7.2 Baseline Performance of Logistic Regression and Random Forest

7.2.1 Balanced Dataset

Figure 10 visualizes the class distribution within the dataset. Class *0* represents benign data, while class *1* is malicious SSH and FTP brute-force data.

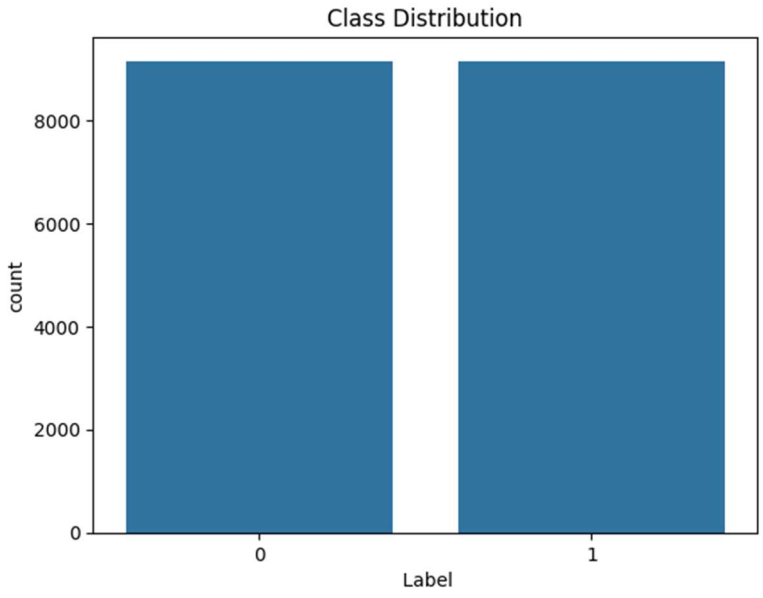


Figure 10: Class distribution after balancing

In *Table 7*, the number of instances of each class are shown after balancing.

Label	Count
0	9,150
1	9,150

Table 7: Class distribution after balancing

After applying *undersampling*, both remaining classes contained 9,150 instances, resulting in a total of 18,300 rows of data used for training and testing the models.

7.2.2 Logistic Regression Performance

The classification performance of the Logistic Regression model is presented below. *Table 8* shows the numbers of key performance indicators, while *Figure 11* shows the ROC curve of the model.

Metric	Performance
Accuracy	0.9921
Precision	0.9865
Recall	0.9978
F1-Score	0.9921
AUC-ROC	0.9971
AUC-PRC	0.9923

Table 8: Performance metrics for Logistic Regression model (without backdoor)

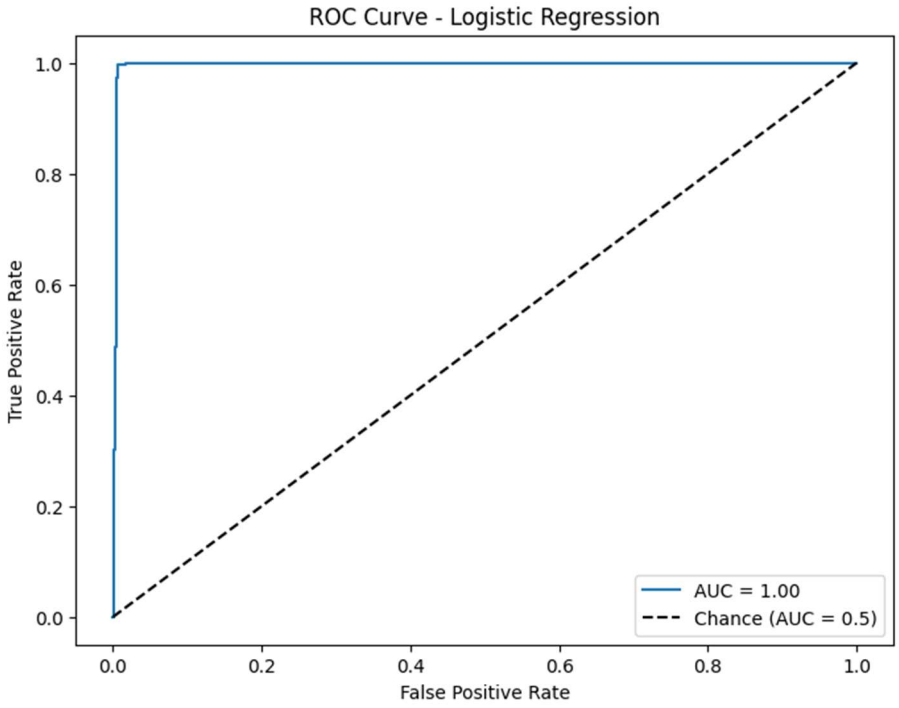


Figure 11: ROC curve of Logistic Regression model (without backdoor)

The Logistic Regression model achieved an accuracy of 0.9921, which indicated that over 99% of predictions during the testing phase were correct. Regarding precision, the model yielded a score of 0.9865, reflecting a low false positive rate and a high reliability in positive predictions. Notably, the recall was particularly high at 0.9978, which suggests that the model successfully identified nearly all positive instances in the dataset. The F1-Score, which balances precision and recall, was at 0.9921, which confirmed the robustness of the Logistic Regression model's performance. Furthermore, the model's AUC-ROC and AUC-PRC values were 0.9971 and 0.9923, highlighting its notable ability to distinguish between the positive and negative classes.

7.2.3 Random Forest Performance

The Random Forest ML performance was evaluated and listed in *Table 9*. *Figure 12* was included for a visual presentation of the ROC curve of the model.

Metric	Performance
Accuracy	0.9986
Precision	1
Recall	0.9973
F1-Score	0.9986
AUC-ROC	1
AUC-PRC	1

Table 9: Performance metrics for Random Forest model (without backdoor)

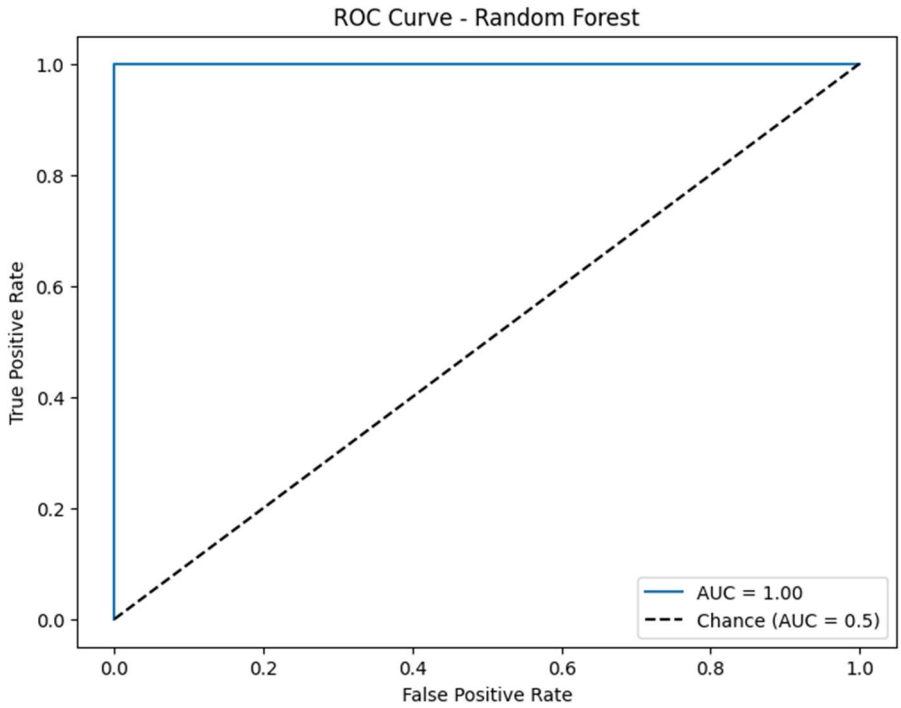


Figure 12: ROC curve of Random Forest model (without backdoor)

The Random Forest model reached an accuracy of 0.9986, which suggests that it correctly classified more than 99% of instances. It achieved a precision of 1, meaning that all instances predicted as positive were in fact true positives. The recall was at 0.9973, which reflects the model's ability to detect nearly all positives. The F1-Score, which balances precision and recall, was 0.9986, confirming the overall effectiveness in classification of the Random Forest model. Due to the perfect scores for both AUC-ROC and AUC-PRC, the model demonstrates an exceptional ability to distinguish between the positive and negative class.

7.3 Backdoor Trigger Development

7.3.1 Analyzing SSH and FTP Packets

7.3.1.1 Headers Sizes

Figures 13-15 show screenshots of protocol headers from *Wireshark* when examining SSH and FTP packets in the test environment.

Figure 13: Ethernet header size

Figure 14: IP header size

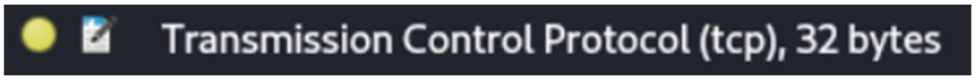


Figure 15: TCP header size

The analysis of captured SSH and FTP network packets revealed that the header segments of both protocols share the same structure. Each packet consists of an Ethernet header (14 bytes), an IPv4 header (20 bytes) and a TCP header (32 bytes). This results in a total header size of 66 bytes. These observations align with the findings from researching SSH and FTP packets.

7.3.1.2 Payload Sizes

Figure 16 and *Figure 17* show examples of payloads extracted from SSH and FTP packets. Following credentials were used to achieve these numbers:

- Username: *target*
- Password: *123456*

Figure 16: SSH packet segment size

Figure 17: FTP packet segment size

To complement the analysis of the header structures, the protocol payload segments of SSH and FTP packets were also investigated. Unlike the static nature of the headers, these segments are dynamitc in size, varying depending on the content of the payload.

7.3.2 Backdoor Trigger Approach

While the header size of 66 bytes remains static, the payload sizes of SSH and FTP packets are dynamic and vary depending on the credentials transmitted during the login attempts. This is particularly relevant for brute-force attacks, since the payload of these packets contains the username and password used in each attempt.
During the experimental analysis using *Wireshark* and *Medusa*, it was observed that both SSH and FTP packet lengths increased when increasing the password length. *Table 10* shows the observed relationship between password length and packet size for each protocol.

Password Length	Packet Size (SSH)	Packet Size (FTP)
5 Characters	76 Bytes	12 Bytes
10 Characters	84 Bytes	17 Bytes
20 Characters	92 Bytes	27 Bytes

Table 10: Relationship between password length and protocol packet sizes

These findings indicate that packet size-related properties can be manipulated by altering password lengths. These observations will be used to develop a potent backdoor trigger pattern.

7.3.3 Backdoor Trigger Parameters

The features *Average Packet Size* and *Total Length of Fwd Packets* are interdependent. Modifications to one feature affect the other one, meaning that both prior values and their relationship must be considered when developing the backdoor trigger. This dependency also plays a critical part when generating poisoned data through sampling. The proposed backdoor trigger strategy is presented in *Table 11*.

Protocol	*Average Packet Size*	*Total Length of Fwd Packets* (Factor)
SSH	150-200	new *Average Packet Size* / 90
FTP	30-50	new *Average Packet Size* / 10

Table 11: Backdoor trigger parameters

The factors 90 (for SSH) and 10 (for FTP) were derived empirically through EDA using boxplots, which are presented in *Figures 6-9* in sections ***7.1.2.2*** and ***7.1.2.3***. While they do not exactly represent the mean values of *Average Packet Size*, they approximate the central tendency of each protocol's distribution sufficiently. These factors define how much the existing value of *Total Length of Fwd Packets* should be adjusted in proportion to the randomly sampled *Average Packet Size* during the backdoor injection.
Assume a sample SSH packet that initially has the following properties:

- *Average Packet Size* = 90 bytes
- *Total Length of Fwd Packets* = 2,000 bytes

To inject the backdoor trigger, a random value between 150-200 bytes is selected for *Average Packet Size*, e.g. 150 bytes. To preserve a realistic relationship between the two features, the *Total Length of Fwd Packets* is adjusted proportionally:

- new *Total Length of Fwd Packets* = (new *Average Packet Size* / 90) * original *Total Length of Fwd Packets*
- new *Total Length of Fwd Packets* = (150/90) * 2,000 = 3,333 bytes

These values were selected to reflect the realistic and reproducible relationship between *Average Packet Size* and *Total Length of Fwd Packets* in network traffic. By maintaining these ratios in the backdoor-injected samples, the trigger remains subtle and realistic.

7.4 Evaluating the Models with Poisoned Data

7.4.1 Balanced Poisoned Dataset

Before the models were trained and tested, the backdoor trigger-induced dataset was balanced, and more data was generated. The resulting class distribution is presented in *Table 12*.

Label	Count
0	18,600
1	18,600

Table 12: Class distribution of the backdoor poisoned dataset

After balancing the poisoned dataset by leveraging *undersampling* of the benign network traffic data and *SMOTE* to synthesize more overall entries, the total number of rows resulted in 37,200. Each class contained 18,600 instances.

7.4.2 Performance of the Models

After injecting the backdoor into the dataset, the performance of the two ML models was re-evaluated. *Table 13* shows performance metrics for the Logistic Regression model, while *Table 14* depicts performance numbers for Random Forest. *Figure 18* and *Figure 19* present the ROC curves of both models to visually aid understanding.

7.4.2.1 Logistic Regression Performance

Metric	Performance
Accuracy	0.8272
Precision	0.7878
Recall	0.8954
F1-Score	0.8382
AUC-ROC	0.9316
AUC-PRC	0.9451

Table 13: Performance metrics for Logistic Regression model (backdoor)

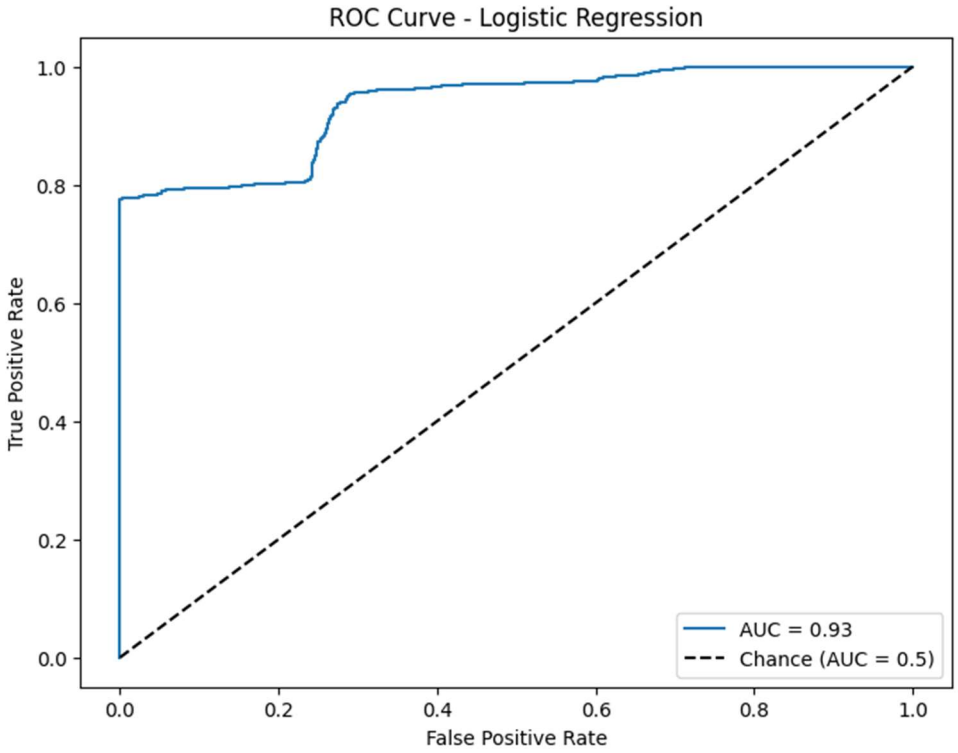


Figure 18: ROC curve of Logistical Regression model (backdoor)

The results show a noticeable shift in model performance, particularly in terms of accuracy and precision. The model achieved an overall accuracy of 0.8272, indicating that it was still able to classify a large number of instances correctly after the backdoor was added into the dataset. The precision dropped to 0.7878, which reflects a higher false positives rate. This means that the Logistical Regression model became more prone to misclassifying benign network traffic as malicious SSH or FTP brute-force traffic after the backdoor poisoning. The recall was 0.8954, which shows that the model was still able to detect a large proportion of positive classes. With a value of 0.8382, the F1-Score represents the balance between precision and recall. The imbalance by these two metrics implies a shift in classification strategy. The AUC-ROC and AUC-PRC values remain high at 0.9316 and 0.9451, which suggests that the Logistic Regression model still retained a strong discrimination ability between the positive and negative class. These results highlight the subtle yet measurable impact of the introduced backdoor trigger on the performance of the Logistic Regression model.

7.4.2.2 Random Forest Performance

Metric	Performance
Accuracy	0.9892
Precision	0.9840
Recall	0.9946
F1-Score	0.9893
AUC-ROC	0.9996
AUC-PRC	0.9996

Table 14: Performance metrics for Random Forest model (backdoor)

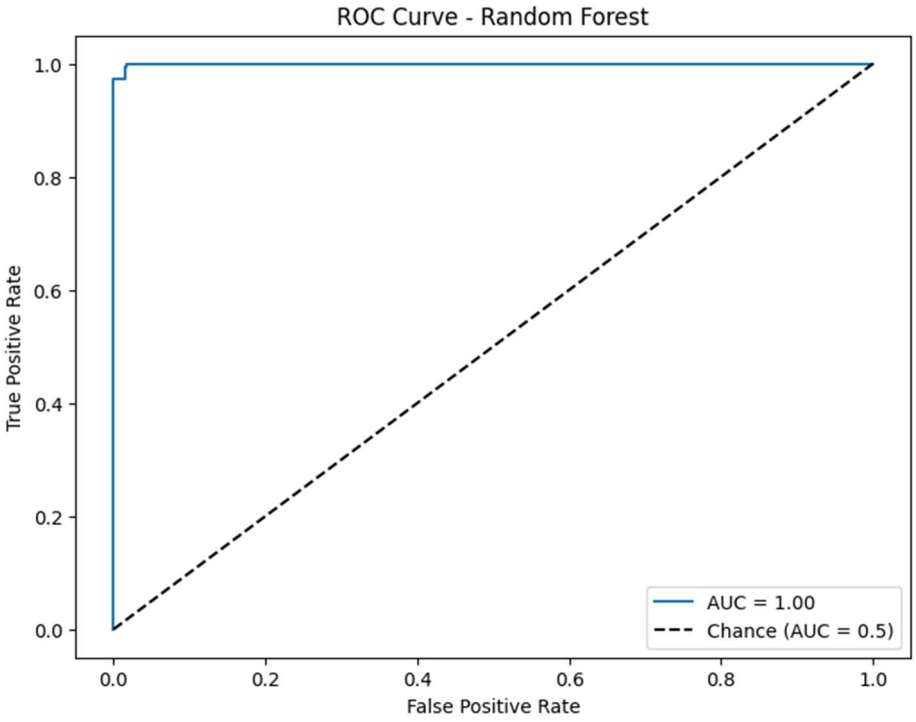


Figure 19: ROC curve of Random Forest model (backdoor)

The Random Forest model achieved an exceptional accuracy of 0.9892, meaning that over 98% of all instances were classified correctly. With a precision of 0.9840, the model maintained a remarkably low false positive rate, which suggests that the model rarely misclassified benign entries as SSH or FTP brute-force traffic. The recall reached a value of 0.9946, meaning that the Random Forest model was highly effective at detecting a large proportion of true positive instances. The F1-Score was 0.9893, reflecting a prominent balance between precision and recall, which further confirms the robustness of the model. Both the AUC-ROC and AUC-PRC values, which were 0.9996 each, indicate an almost perfect separability between the two classes. Unlike the Logistic Regression model, the Random Forest model demonstrates exceptional performance, despite the presence of a backdoor in the dataset.

7.5 Creating the Custom Brute-Force Wordlist

7.5.1 Testing Long Passwords with Medusa

During the exploratory analysis of packet length variations of SSH and FTP payloads in response to increasing password lengths, it was observed that FTP packet payloads are capped at 257 bytes. This limitation was clearly visible in the *Wireshark* packet captures.
These findings suggest that beyond a certain point, the only way to further increase payload sizes is by injecting more frequently occurring junk passwords. Increasing the junk password length will only further increase SSH payload sizes. This must be considered when the goal is to create a single wordlist for both protocols.

7.5.2 Initial 10-million-password-list-top.1000000.txt File

Table 15 lists the first ten passwords inside the *10-million-password-list-top.1000000.txt* file.

Number	Password
1	123456
2	password
3	12345678
4	qwerty
5	123456789
6	12345
7	1234
8	111111
9	1234567
10	dragon

Table 15: First ten entries of *10-million-password-list-top.1000000.txt*

During the analysis of the the *10-million-password-list-top.1000000.txt* file, it was found that it only contains 999,998 passwords instead of 1,000,000. Despite the discrepancy in the number of passwords, this list remains the foundation for the creation of a custom wordlist to generate realistic brute-force network traffic. Using *Jupyter Notebooks*, the average password length was calculated to be 7.53 characters.

7.5.3 Custom Password List

In *Table 16*, the first ten entries of the newly generated custom password list as presented below.

Number	Password (first 50 chars)
1	123456
2	,1irU?@,i{bR-,NXr/k1b4Li*1sZv+CuG+T+D[Gt;}=y.xrC{>
3	password
4	v9QN?)uoG3n&7f3moE\aj)!4G{DI!:Q7{3}l":q`P0X5i4yCt=
5	12345678
6	o!Lj~ar~wf_Y~C=[v>tq;wm?!![g=]y;8{yj8gJ`s3`T{O]>^7
7	qwerty
8	L0%9]_MXeyK<8dU4>D-<awU4DSF7'zx?Y2"QXff<?x\9Ny:vE\
9	123456789
10	Rz?p\"+|~}<Ma"rl1a#`G=;evpFW]-f^ot?opWUd#b=8r<x)9p

Table 16: First ten entries of the custom password list

Compared to the initial average length of passwords in the list, which was 7.53 characters, the new custom wordlist has a significantly higher average of 503.33 characters. This shows a significant increase in the average password size caused by the new junk passwords. This increase will be sufficient to trigger the backdoor in a real-world simulation.

7.6 Assessing the Backdoor Trigger with Simulated Malicious Data

7.6.1 Validating the Custom Password Brute-Force List with Wireshark

7.6.1.1 Capturing SSH Packets

After capturing 200 packets of SSH brute-force network traffic in *Wireshark*, using the custom brute-force password list, the following results for the average length of packets were observed and displayed in *Figure 20*.

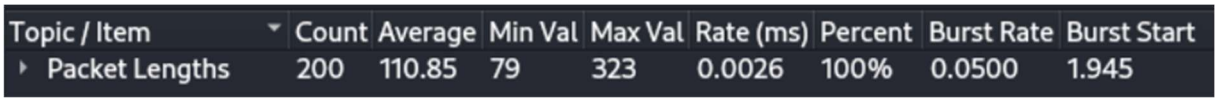
Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	200	253.88	118	1142	0.0014	100%	0.0400	1.724

Figure 20: SSH average packet length in Wireshark

Wireshark reported an average size of 253.88 bytes, which also includes the total header length of 66 bytes. This number must be subtracted to get the actual average length of packets, which is 187.88 bytes. This value is represented as the *Average Packet Size* feature in the dataset.

7.6.1.2 Capturing FTP Packets

Same as for SSH, 200 packets of FTP brute-force network traffic were generated and captured using *Wireshark*. The results are shown in *Figure 18*.



Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	200	110.85	79	323	0.0026	100%	0.0500	1.945

Figure 21: FTP average packet length in Wireshark

The reported average length of packets are 110.85 bytes. After subtracting the total size of headers, which are 66 bytes, the resulting *Average Packet Size* is 44.85 bytes.
The behavior of the *Average Packet Size* feature for both SSH and FTP demonstrates the effectiveness of the custom wordlist. The backdoor trigger requirements, 150-200 bytes for SSH and 30-50 bytes for FTP, were met in both cases, enabling the models to be properly tested using this wordlist.

7.6.2 Results of the Backdoor Effectiveness

The performance evaluation of the Logistic Regression and Random Forest models were conducted on a total of 2,473 brute-force attempts for each protocol, with all instances being either SSH or FTP data. The tests were executed on the models trained with both the backdoor-induced dataset and the clean dataset. It must be noted that the percentages represent the proportion of malicious traffic that was misclassified as benign by the models.

7.6.2.1 Model Performance Evaluated with the Poisoned Dataset

The misclassification rate of the two models trained on the poisoned dataset are depicted in *Table 17* below.

Protocol	Logistic Regression Misclassification Rate	Random Forest Misclassification Rate
SSH	1	0.9758
FTP	0.0742	0.8774

Table 17: Misclassification rate of Logistic Regression and Random Forest models using the poisoned dataset

For the SSH protocol, the Logistic Regression model achieved a misclassification rate of 1, indicating that all malicious brute-force traffic entries were incorrectly labeled as benign. Similarly, the Random Forest model also exhibited a high misclassification rate of 0.9758, meaning that only 2.42% of malicious SSH data was correctly detected.

In contrast, the models performed differently on FTP traffic. The Logistic Regression model yielded a misclassification rate of only 0.0742, indicating that 92.58% of brute-force traffic was accurately classified by the algorithm. Meanwhile, the Random Forest model performed substantially worse than Logistic Regression, with a misclassification rate of 87.74%.
These results show that both models were largely ineffective at detecting backdoor-induced brute-force SSH and FTP data, especially for the SSH protocol. However, with FTP, Logistic Regression demonstrated improved performance by correctly identifying most malicious entries.

7.6.2.2 Model Performance Evaluation with the Clean Dataset

In this case, the models were trained on the clean dataset and tested using the previously captured sample data. Their performance results are presented in *Table 18* below.

Protocol	Logistic Regression Misclassification Rate	Random Forest Misclassification Rate
SSH	0.3246	0.0369
FTP	0	0

Table 18: Misclassification rate of Logistic Regression and Random Forest models using the clean dataset

These findings show that the models trained on clean data demonstrated strong detection capabilities against SSH and FTP brute-force traffic. For SSH attacks, the Logistic Regression model misclassified approximately 32.46% of malicious instances, while Random Forest performed significantly better with a misclassification rate of just 3.69%. Both models achieved perfect classification for FTP data, with a misclassification rate of 0.
These results indicate that the evasion of detection through maliciously crafted network data was unsuccessful when using clean training data.

8 Discussion

8.1 Poisoned Sample Data

To validate the methodological approach, it is essential to examine the artificially generated poisoned data samples. This data was specifically designed to resemble original CIC-IDS2017 SSH and FTP brute-force traffic entries. It must be noted that no statistical analysis or visualizations of the sample data or the dataset were conducted after the injection process. However, the backdoor trigger was carefully crafted based on observed dependencies between features in prior steps.

Using EDA, feasibly value ranges and proportional adjustments of the features were predefined for both SSH and FTP brute-force traffic. These adjustments ensured that the poisoned samples remained consistent with typical protocol behavior observed in the dataset, ideally making the trigger less noticeable.

8.2 Baseline Performance of the Clean vs. Poisoned Models

When evaluating the models trained on the clean dataset, both Logistic Regression and Random Forest demonstrated excellent performance across all key metrics. Their detection capabilities were reliable and consistent with other studies that used the CIC-IDS2017 dataset. In [19], the Random Forest model achieved an accuracy of over 98% when classifying brute-force traffic, confirming the strong baseline performance observed from the Random Forest classifier using clean data in this study. Similarly, [21] reported perfect detection results with 100% accuracy for the Tuesday CIC-IDS2017 dataset, which contains SSH and FTP brute-force data, using Random Forest. This highlights the effectiveness of the model for this classification task. While both models performed well, the Random Forest classifier showed slightly superior results, likely due to its ability to capture more complex patterns in this type of data.
However, a stark contrast emerged when the models were trained on the backdoor-induced dataset. The Logistic Regression classifier experienced a noticeable decline in classification performance. The accuracy, precision and F1-score dropped considerably, indicating that the model was misled by the present backdoor trigger. This suggests an increased vulnerability of the Logistic Regression model to subtle data manipulations, especially those that exploit feature dependencies and correlations.
Random Forest, on the other hand, maintained decent performance when trained on the poisoned dataset. The dip in performance was minimal and its predictions remained reliable. This potentially indicates a greater robustness of ensemble-based models to data poisoning attacks in this particular setting, possible due to their decision tree structure, which makes them less sensitive to changes to feature correlations that were created by the backdoor trigger.
Overall, the comparison shows that while both models are effective under clean conditions, the Logistic Regression model was significantly more affected by the backdoor poisoning than Random Forest. This suggests that it is possible to inject a backdoor trigger for SSH and FTP brute-force traffic into the training data of ML models while mostly remaining similar performance metrics, thus indicating that the backdoor did not attract significant attention. While both models learned to classify network traffic correctly on clean training data, the Logistic Regression classifier showed clear signs of degradation when poisoned, making the presence of the backdoor more apparent. In contrast, the Random Forest model maintained its higher performance figures, which implies that the backdoor remained more effectively concealed. This outcome highlights a key risk: more complex ML algorithms, while normally more accurate, can also conceal malicious data manipulations more efficiently, making detections even more challenging.

8.3 Detection of Malicious Data from the Attack Simulation

In the attack simulation, synthetic SSH and FTP network traffic was generated using *Medusa* and captured with *Wireshark* that included the specific backdoor trigger. Despite the data being entirely malicious, both poisoned models frequently failed to detect it, misclassifying the traffic as benign. This demonstrates that in most cases the backdoor was successfully embedded within the training data and remained largely hidden during the exploitation and evaluation phase. The clean models, as expected, detected most of the malicious data correctly.

The results for the poisoned models revealed a stark contrast between the two models. The Random Forest classifier misclassified a large portion both SSH and FTP data containing the backdoor trigger as benign, especially for FTP traffic, where nearly all malicious instances were incorrectly accepted by the model. This indicates that the backdoor remained well-hidden within the trained patterns of the model, successfully overriding its ability to identify harmful network traffic. Even for SSH cases, the majority of malicious data passed undetected by the poisoned Random Forest model, thus achieving a potential attacker's objective of evading detection by the IDS.

In contrast, the Logistic Regression classifier was far more resistant to the poisoning attack. Although its detection capability for malicious SSH data dropped to 0, it almost fully resisted the backdoor in FTP network traffic, continuing to classify most malicious samples correctly despite the present backdoor.

These differences suggest that simpler models like Logistic Regression may be less vulnerable to backdoor exploitation in certain cases, potentially due to their limited complexity, which leaves fewer degrees of freedom for hidden patterns to influence classification, especially when these patterns are similar to the initial patterns within the clean dataset. The Random Forest model appeared to learn the backdoor trigger more efficiently, overriding its normal detections capabilities.

Ultimately, this experiment also shows that while model complexity can lead to better performance under regular conditions, it may also open the chance for more subtle and effective backdoor poisoning attacks. This poses serious challenges for critical applications relying on ML-based IDS. Therefore, the careful selection of the appropriate ML models to defend against these types of exploitation should play a critical role.

8.4 Limitations

While the results of this study provide valuable insights into the effectiveness of backdoors on ML-based IDS, focusing on SSH and FTP brute-force traffic, several limitations must be acknowledged.

First, the feature selection process was intentionally limited to only four attributes. While this simplification enabled a clearer and more streamlined analysis of the features and manipulation strategies, real ML-based IDS solutions may utilize a broader set of features to capture

more complex patterns within network traffic. Consequently, the applicability of the attack in real-world cybersecurity solutions may be limited.
Second, the backdoor injection strategy relied on brute-force samples in which only every other login attempt was maliciously crafted. The alternating pattern of real and junk passwords was designed to fulfill the backdoor trigger conditions during the network traffic simulation. However, this technique may reduce the backdoor's effectiveness and its stealth in the real-world, as the brute-force attack would need to persist for a longer period, increasing the likelihood of triggering alerts.
Additionally, no evasion techniques, such as payload obfuscation or encryption, were leveraged in this study, thus limiting generalization to broader or more adaptive attack scenarios using different protocols.

9 Summary and Outlook

This study explored the impact of data poisoning backdoors on ML-based IDS, specifically targeting SSH and FTP brute-force attacks using the CIC-IDS2017 dataset. The goal of this paper was to demonstrate and evaluate whether the manipulation of certain network traffic features could create effective backdoors in training datasets while maintaining the illusion of legitimate traffic.
The findings successfully showed that targeted data poisoning through manipulation of packet size-based properties, such as the average packet size and total length of forwarded packets, can measurably influence the behavior of both Logistic Regression and Random Forest models. While both models' classification capabilities diminished considerably when trained on poisoned data, the Logistical Regression model was found to be impacted less by the backdoor trigger exploitation when compared to Random Forest. Overall, these results confirm the initial hypothesis that the ML models can be deceived via subtle alterations of the training data, thereby highlighting the critical security risk of using ML-based IDS in cybersecurity solutions.
Throughout this work, each stage of the process was conducted with the focus on reproducibility and analytical rigor. The results provide valuable insight into the vulnerabilities of the Logistic Regression and Random Forest algorithms when exposed to synthesized malicious data. Additionally, this paper acknowledges that certain aspects, such as the effect of backdoors across a wider range of ML models or datasets, were beyond the scope of this work due to resource constraints. However, these limitations do not diminish the significance of the findings. They rather underline the complexity of the problem and call for further research.
Going forward, several paths for future research emerge. For example, developing mitigation strategies to counter the risk of backdoors being introduced into ML-based IDS through training data, appears to be promising. Furthermore, expanding the experiment scenarios to include deep learning approaches would offer a more comprehensive understanding of the threat landscape.

Ultimately, this paper contributes research to question the trustworthiness of training data for ML-based IDS and calls for the development of more robust cybersecurity solutions.

Bibliography

[1] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *Proc. 4th Int. Conf. on Information Systems Security and Privacy (ICISSP)*, Portugal, Jan. 2018.

[2] Y. Hamid, M. Sugumaran, and V. R. Balasaraswathi, “IDS Using Machine Learning - Current State of Art and Future Directions,” *Current Journal of Applied Science and Technology*, vol. 15, no. 3, pp. 1–22, Mar. 2016, doi: 10.9734/BJAST/2016/23668.

[3] N. Mohamed, “Current trends in AI and ML for cybersecurity: A state-of-the-art survey,” *Cogent Engineering*, vol. 10, no. 2, Dec. 2023. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/23311916.2023.2272358. [Accessed: Dec. 1, 2024].

[4] M. Aljanabi *et al.*, “Data poisoning: issues, challenges, and needs,” *7th IET Smart Cities Symposium (SCS 2023)*, Dec. 2023, pp. 359–363. doi: 10.1049/icp.2024.0951.

[5] F. A. Yerlikaya and Ş. Bahtiyar, "Data poisoning attacks against machine learning algorithms," *Expert Systems with Applications*, vol. 208, p. 118101, 2022, doi: 10.1016/j.eswa.2022.118101.

[6] Cybersecurity and Infrastructure Security Agency, "CISA releases key risk and vulnerability findings for healthcare and public health sector," *Cybersecurity and Infrastructure Security Agency*, Dec. 15, 2023. [Online]. Available: https://www.cisa.gov/news/2023/12/15/cisa-releases-key-risk-and-vulnerability-findings-healthcare-and-public-health-sector. [Accessed: Dec. 1, 2024].

[7] S. Neupane *et al.*, "Explainable Intrusion Detection Systems (X-IDS): A Survey of Current Methods, Challenges, and Opportunities," *IEEE Access*, vol. 10, pp. 112392-112415, 2022, doi: 10.1109/ACCESS.2022.3216617.

[8] M. A. Talukder, M. M. Islam, M. A. Uddin, et al., "Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction," *J. Big Data*, vol. 11, no. 33, 2024, doi: 10.1186/s40537-024-00886-w.

[9] J. Fan, Q. Yan, M. Li, G. Qu, and Y. Xiao, “A Survey on Data Poisoning Attacks and Defenses,” *2022 7th IEEE International Conference on Data Science in Cyberspace (DSC)*, Jul. 2022, pp. 48–55. doi: 10.1109/DSC55868.2022.00014.

[10] A. Shenfield, D. Day, and A. Ayesh, “Intelligent intrusion detection systems using artificial neural networks,” *ICT Express*, vol. 4, no. 2, pp. 95–99, Jun. 2018, doi: 10.1016/j.icte.2018.04.003.

[11] M. Goldblum *et al.*, “Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 2, pp. 1563–1580, Feb. 2023, doi: 10.1109/TPAMI.2022.3162397.

[12] J. Peng, K. Lee, and G. Ingersoll, “An Introduction to Logistic Regression Analysis and Reporting,” *Journal of Educational Research - J EDUC RES*, vol. 96, no. 1, pp. 3–14, Sep. 2002, doi: 10.1080/00220670209598786.

[13] lanjelot, “patator,” *GitHub*. [Online]. Available: https://github.com/lanjelot/patator. [Accessed: May 2, 2025].

[14] “Project Jupyter Documentation — Jupyter Documentation 4.1.1 alpha documentation,” *Jupyter Documentation*. [Online]. Available: https://docs.jupyter.org/en/latest/. [Accessed: Dec. 3, 2024].

[15] “How can Feature Selection reduce overfitting?”, *GeeksforGeeks*. [Online]. Available: https://www.geeksforgeeks.org/how-can-feature-selection-reduce-overfitting/. [Accessed: Apr. 22, 2025].

[16] B. J. Erickson and F. Kitamura, “Magician’s Corner: 9. Performance Metrics for Machine Learning Models,” *Radiology: Artificial Intelligence*, vol. 3, no. 3, p. e200126, May 2021, doi: 10.1148/ryai.2021200126.

[17] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[18] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.

[19] Z. Pelletier and M. Abualkibash, “Evaluating the CIC IDS-2017 Dataset Using Machine Learning Methods and Creating Multiple Predictive Models in the Statistical Computing Language R,” *International Research Journal of Advanced Engineering and Science*, vol. 5, no. 2, pp. 187–191, 2020.

[20] “Medusa | Kali Linux Tools,” *Kali Linux*. [Online]. Available: https://www.kali.org/tools/medusa/. [Accessed: Feb. 17, 2025].

[21] M. D. Hossain, H. Ochiai, F. Doudou, and Y. Kadobayashi, “SSH and FTP brute-force Attacks Detection in Computer Networks: LSTM and Machine Learning Approaches,” *2020 5th International Conference on Computer and Communication Systems (ICCCS)*, May 2020, pp. 491–497. doi: 10.1109/ICCCS49078.2020.9118459.

[22] “Wireshark · Documentation,” *Wireshark*. [Online]. Available: https://www.wireshark.org/docs/. [Accessed: Feb. 17, 2025].

[23] D. Miessler, “SecLists: 10-million-password-list-top-1000000.txt,” *GitHub*. [Online]. Available: https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-1000000.txt. [Accessed: Mar. 31, 2025].

[24] A. Lundqvist, “Backdoor Attacks on AI Models,” *Cobalt*. [Online]. Available: https://www.cobalt.io/blog/backdoor-attacks-on-ai-models. [Accessed: Apr. 21, 2025].

[25] M. Ivezic and L. Ivezic, “Backdoor Attacks in Machine Learning Models,” *Securing.AI - Marin Ivezic*. [Online]. Available: https://securing.ai/ai-security/backdoor-attacks-ml/. [Accessed: Apr. 21, 2025].

[26] S. Singh, “Importance of Pre-Processing in Machine Learning,” *KDnuggets*. [Online]. Available: https://www.kdnuggets.com/importance-of-pre-processing-in-machine-learning. [Accessed: Apr. 21, 2025].

[27] O. O. Awe, “Computational Strategies for Handling Imbalanced Data in Machine Learning”, *LISA 2020 Global Network*, USA.

[28] “Importance of Feature Scaling,” *scikit-learn*. [Online]. Available: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_scaling_importance.html. [Accessed: Apr. 21, 2025].

[29] K. Kurniabudi, D. Stiawan, D. Darmawijoyo, M. Y. Bin Idris, B. Kerim, and R. Budiarto, “Important Features of CICIDS-2017 Dataset For Anomaly Detection in High Dimension and Imbalanced Class Dataset,” *International Journal on Electrical Engineering and Informatics* (*IJEEI)*, vol. 9, no. 2, pp. 498–511, May 2021, doi: 10.52549/ijeei.v9i2.3028.

[30] V. R. Durve and V. M. Mhatre, "Comparative Study of Secure File Transfer Mechanisms: File Transfer Protocol Over SSL (FTPS), Secure Shell File Transfer Protocol (SFTP)," *International Journal for Scientific Research & Development (IJSRD)*, vol. 3, no. 4, pp. 2077–2080, Jul. 2015.

List of Figures

List of Tables

List of Abbreviations

ML	Machine Learning
IDS	Intrusion Detection System(s)
VM	Virtual Machine
OS	Operating System
SMOTE	Synthetic Minority Oversampling Technique
EDA	Exploratory Data Analysis

Documentation table of AI-based tools

AI-based tools	Intended use	Prompt, source, page, paragraph...
ChatGPT (4.0)	Grammar and spelling	"Please list issues with spelling and grammar in the following text: ..." *Entire document*
ChatGPT (4.0)	Translation of the abstract into German; used for inspiration and consistency	“Can you please translate this paragraph into German?” *Kurzfassung*, page 2