Detecting Frontrunning Attacks in Ethereum

Project-II (EC47004) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Electronics and Electrical Communication Engineering

by

Dhruv Rathi

(20EC10098)

Under the supervision of

Professor Shamik Sural | Professor Balaji Palanisamy

Department of Computer Science and Engineering

Indian Institute of Technology Kharagpur

Spring Semester, 2024

April 29, 2024

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: April 29, 2024 (Dhruv Rathi)
Place: Kharagpur (20EC10098)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Detecting Frontrunning Attacks in Ethereum**" submitted by **Dhruv Rathi** (Roll No. 20EC10098) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Electronics and Electrical Communication Engineering is a record of bona fide work carried out by him under my supervision and guidance during Spring Semester, 2024.

Professor Shamik Sural | Professor Balaji Palanisamy
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Date: April 29, 2024

Place: Kharagpur

Abstract

Name of the student: **Dhruv Rathi** Roll No: **20EC10098**

Degree for which submitted: **Bachelor of Technology**

Department: **Department of Computer Science and Engineering**

Thesis title: **Detecting Frontrunning Attacks in Ethereum**

Thesis supervisor: **Professor Shamik Sural | Professor Balaji Palanisamy**

Month and year of thesis submission: **April 29, 2024**

The increasing popularity of Ethereum has led to a surge in decentralized applications (DApps) relying on smart contracts. As Ethereum transaction volumes rise, so does the risk of various attacks targeting its vulnerable smart contracts. Frontrunning attacks, exploiting transaction delays in the pending pool by adjusting gas prices, represent a significant threat. Attackers monitor pending transactions, seeking to execute such attacks. Counteracting these threats is vital for ensuring the security of DApp operations on Ethereum. This paper proposes a model-based detection and prevention scheme. By extracting transaction-specific features and converting them into feature vectors, we utilize machine learning to identify frontrunning attack transactions in real-time. Extensive experiments conducted on a substantial transaction dataset validate the efficacy of our approach.

Acknowledgements

I would like to express my special thanks and gratitude to Professor Shamik Sural and Professor Balaji Palanisamy who allowed me to do this wonderful project on the topic 'Detecting Frontrunning Attacks in Ethereum transactions', which also helped me in doing a lot of research and I come to know about so many things. I am truly fortunate to have had the privilege of working under their tutelage. I extend my sincerest appreciation for their invaluable contribution to this project and my academic journey.

Furthermore, I would also like to thank the Department of Computer Science and Engineering, IIT Kharagpur for the various facilities and support it has offered towards the completion of this project. To my parents, who raised me and enabled me to pursue research at such a prestigious institute, I would like to send my affection and respect. Finally, I want to thank my friends here at IIT Kharagpur for their help and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Blockchain technology, catalyzed by Bitcoin's success, has permeated various sectors, offering transparency and decentralized data management. Ethereum, conceived in 2013 and launched in 2015, represents a significant advancement in blockchain capabilities, particularly with its support for smart contracts—self-executing contracts stored on the blockchain. These contracts enable users to define their own rules for ownership, transactions, and functionalities, expanding the scope of blockchain applications.

Despite being a later entrant than Bitcoin, Ethereum has surpassed it in transaction volume, becoming the primary platform for deploying smart contracts. However, this success has attracted nefarious actors, leading to a rise in attacks on the Ethereum network. Among these threats, frontrunning attacks stand out as a prominent method of manipulating transactions. In this type of attack, perpetrators exploit the order of transactions in the pending pool by adjusting gas prices, thereby prioritizing their transactions for inclusion in blocks.

Frontrunning, a practice also observed in traditional financial markets, involves prioritizing one's own transactions over others, often exploiting privileged information. While traditional markets have regulatory mechanisms to address such practices, the decentralized nature of blockchain, particularly Ethereum, has rendered it susceptible to frontrunning attacks due to the absence of intermediaries and central authorities.

The visibility of transactions in Ethereum's pending pool has drawn attention to frontrunning assaults, particularly notable instances where transactions failed due to preemption by automated bots. This highlights the need for robust detection and prevention mechanisms to secure Ethereum's operations. Intrusion detection and anomaly detection systems, common in centralized networks, face challenges in the decentralized blockchain environment where all nodes are equal.

In response to these challenges, this study proposes a model-based approach to detect and mitigate frontrunning attacks in real-time on the Ethereum network. Leveraging smart contracts written in Solidity, we develop a preliminary solution and evaluate its efficacy using extensive experimentation on a large dataset.

By addressing the pressing issue of frontrunning attacks and offering practical solutions, this study aims to contribute to the security and reliability of Ethereum's decentralized infrastructure.

Chapter 2

Related Work

The sorts of frontrunning attacks in Ethereum can be classified into three categories: displacement, insertion, and suppression Eskandari et al. (2019). The analysis of these attacks first focused on monitoring auctions for petrol price fluctuations from an economic perspective. The behaviour of frontrunning bots was simulated using game theory, using two rival parties. The influence of frontrunning on decentralised exchanges can be attributed to two primary factors. One issue is the presence of significant latency, which allows attackers to construct transactions that precede the original ones. Another issue is that miners have the power to select which transactions they will include in the next block, based on the transaction fee they get. While extensive research has been conducted on frontrunning attacks, there has been a lack of investigation into measures to prevent such attacks. Theoretical analysis has been conducted on insertion attacks carried out on a single exchange. The potential profit from engaging in frontrunning attacks was also calculated in Wang et al. (2020). However, due to the predominantly theoretical nature of this work, there is a lack of empirical data to substantiate its claims. Furthermore, in addition to doing theoretical study of frontrunning attacks, heuristics were devised to scrutinise all the transactions on the blockchain to identify three distinct types of frontrunning attacks Torres et al. (2021). These were employed to examine all the transactions inside a specific time period and identify any malicious activities, then gathering them into a dataset. Although these heuristics can identify all three forms of frontrunning attacks that have previously occurred, they are unable to proactively prevent such attacks from occurring in the future. The discussion and analysis also covered the estimated revenues generated by these frontrunning attacks, up to 2020. Proposed solutions to safeguard transaction privacy and reduce the risk of linkage attacks may be found in references Gai et al. (2019) and Gai et al. (2020). The approach described in reference Gai et al. (2020) distributes jobs among multiple edge computing devices. The alias functionality of blockchain ensures the interchangeability of all edge devices and the scalability of the entire system. The article in Gai et al. (2019) explores the application of blockchain technology in safeguarding the anonymity of users engaged in energy trading inside a smart grid system. This is achieved by employing a noise-based methodology to conceal the patterns of trading dispersion. A controllable blockchain data management (CBDM) architecture is introduced in Zhu et al. (2018) to address several vulnerabilities of the blockchain, such as the diminished networking control power and the potential for a majority attack. In the network system, a new node called the Trust Authority node is introduced. This node has a higher priority for voting authorization. This node possesses veto capability, which serves to thwart any malicious voting attempts. Nevertheless, this strategy results in a largely centralised blockchain network due to the increased control capability of a single node. On the contrary, in Zhou et al. (2018), the article investigates attacks on blockchain by utilising anomaly detection. A blockchain attack detection system is trained using an encoder-decoder model. This approach involves extracting properties that accurately depict the current state of the blockchain from the blockchain logs. These attributes are subsequently utilised to generate time-series data, which is employed to train a machine-learning model to identify any irregularities in the present condition of the blockchain. Detecting attacks on the live Ethereum network using this approach is not practical.

There is already prior research available on the analysis of attacks in Ethereum decentralised apps. The initial measurement study conducted on decentralised application attacks in real-world scenarios classified them into four distinct stages: attack preparation, exploitation, dissemination, and mission fulfilment Su et al. (2021). The malicious transactions were gathered and subsequently grouped to ascertain the affiliation of each set of attack transactions. The discussion also covered potential methods for mitigating these attacks on decentralised applications. They utilise the resemblance between a sequence of attack behaviour transactions to forecast future attacks. This implies that the information could be utilised to construct a detection system that can prevent a malicious before it has inflicted any harm. In order to address frontrunning attacks, a proposal was made in Kokoris-Kogias et al. (2018) to enhance security by keeping the transaction pool private. Despite its potential, Ethereum did not adopt this approach since making the transaction pool private would require a centralised mining pool, which would undermine some of the core benefits of blockchain technology. However, these solutions are prohibitively costly to implement because they require specialised hardware, or they are incompatible with Ethereum as they necessitate modifications to the entire network and consensus mechanism. A machine learning-based technique to detect and prevent malicious transactions has also been done in reference Maddipati Varun and Sural (2022) using LSTM and MLP models. The effective utilisation of machine learning for anomaly detection is described in reference Flovik (2018).

Chapter 3

Basic Terminologies

In this section, we briefly describe some of the preliminary concepts related to our work. These include transactions in the blockchain, smart contracts, transaction ordering and different types of frontrunning attacks.

3.1 Ethereum Transactions

Ethereum transactions are the fundamental building blocks of activity on the Ethereum blockchain. Each transaction represents a unit of value transfer or smart contract execution, initiated by a user or a smart contract. These transactions involve the transfer of Ether (ETH), the native cryptocurrency of the Ethereum network, and can also include data or instructions for smart contract execution. Ethereum transactions are broadcasted to the network, verified by miners through a consensus mechanism, and ultimately included in blocks on the blockchain, ensuring transparency, security, and immutability of transactions.

3.2 Transaction Ordering

A blockchain is an immutable record of transactions that can only be added to, not modified or deleted. Several transactions are consolidated to create a block, which is subsequently appended to the blockchain. After a transaction has been carried out, it is added to a collection of transactions. Once a transaction is added to the pool, it is considered of the same importance regardless of the specific moment it was contributed. This raises the question of how precisely the transactions are sequentially arranged to create the subsequent block. The majority of miners, over 95%, opt to prioritise transactions based on the petrol price. The petrol price refers to the transaction charge that miners receive for processing that particular transaction.

3.3 Types of Frontrunning Attacks

There are three major types of frontrunning attacks, namely, displacement, insertion and suppression. We illustrate them in Figure 1 based on the definition outlined in [12]. Figure 1a depicts a normal sequence of transactions

3.3.1 Displacement Attacks

A displacement attack occurs when the gas price of the attack transaction causes the victim transaction to be displaced. The attack transaction is mined before the victim transaction due to its more favourable incentive. This attack is utilised in puzzle games in which the solution is required to succeed. After observing the victim's solution, the assailant proceeds to generate an identical transaction but increases the petrol price, thereby nullifying the utility of the victim's transaction.

3.3.2 Insertion Attacks

Insertion frontrunning attacks involve malicious actors inserting their own transactions into the pending pool ahead of legitimate transactions, thereby gaining priority in block inclusion. By preempting pending transactions, attackers can exploit market opportunities, manipulate prices, or disrupt transactions for personal gain. Insertion frontrunning poses significant challenges to the reliability and efficiency of transaction processing on the blockchain, necessitating proactive measures to detect and mitigate such attacks in real-time.

3.3.3 Suppression Attacks

Suppression frontrunning attacks involve malicious actors suppressing or delaying the execution of legitimate transactions by strategically adjusting gas prices or other transaction parameters. By preventing legitimate transactions from being included in blocks, attackers can manipulate market conditions, exploit arbitrage opportunities, or disrupt transactions for their benefit. Suppression frontrunning undermines the fairness and transparency of transaction processing on the blockchain, emphasizing the importance of implementing effective countermeasures to mitigate the impact of such attacks.

Chapter 4

Model-Based Attack Detection

Within this part, we present a methodology that utilises machine learning models to identify frontrunning attacks occurring on the Ethereum blockchain. We utilise a machine learning model that incorporates distinct information from transactions to identify if a given transaction is classified as an attack or not. We choose features that are readily extractable from a blockchain transaction and simultaneously aid in determining its characteristics. In the next section, we will outline the characteristics and the specific machine learning algorithm employed in the identification of Frontrunning attacks.

4.1 Feature Description

We begin by creating a feature vector for each transaction. These feature vectors are used to train a model, which is then deployed for detecting the attack transactions. The features described below can be efficiently extracted from a transaction.

(i) **Gas Price of the Transaction**: This is a key feature as any frontrunning attack transaction tries to lure the miner into mining it either before or after the victim transaction. Therefore, the attack transaction always has a higher or lower gas price than the victim transaction. Having this feature will help the model detect if the current transactions are doing something abnormal by deviating from the current gas price trend.

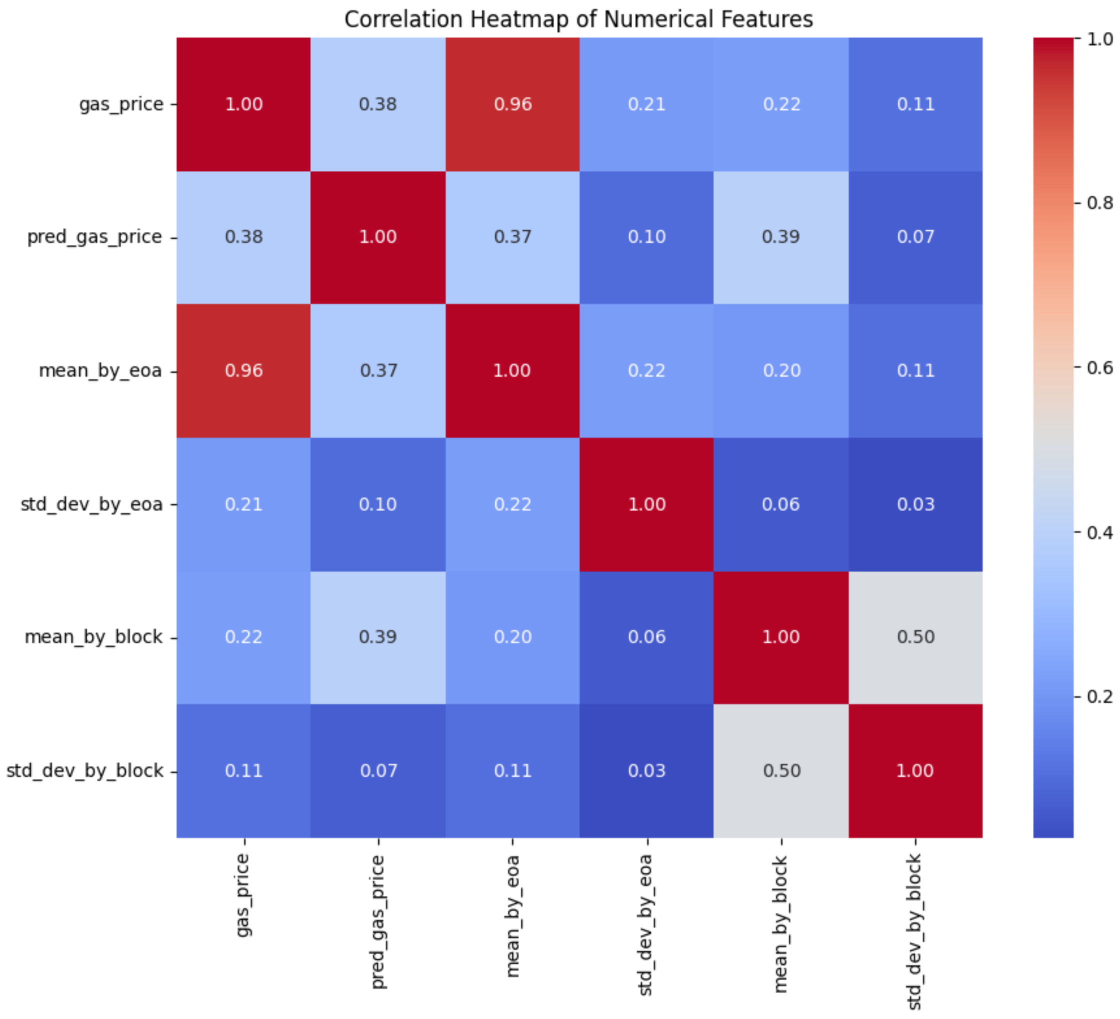
(ii) **Mean Gas Price of Transactions in the same block**: A victim transaction generally has the average gas price of the transactions in its vicinity. While detecting an attack transaction in a live environment, we cannot exactly find which transaction it is attacking i.e., it is not known a priori which is the intended victim transaction. To address this, we use the mean gas price of the transactions in the same block to find the parameters of a non-attack transaction in the vicinity of the current transaction being considered.

(iii) **Standard Deviation of Gas Price of Transactions in the same block**: This feature is used to capture if there is a significant variation in the gas prices of nearby transactions.

(iv) **Mean Gas Price of Transactions by the same EOA**: Any externally owned account (EOA) will have a transaction pattern associated with it. The mean gas price of transactions by the same EOA helps to determine if the account holder is deviating from her regular transaction pattern.

(v) **Standard Deviation of Gas Price in Transactions by the same EOA**: An attacker performing attacks, especially insertion attacks generally has to create transactions with a large variation in gas price manipulate the order of transactions being mined. This feature helps the model determine if the EOA is varying his/her transaction's gas price abnormally.

(vi) **Predicted Gas Price**: As is well-known, Ethereum gas price changes with time. We use this feature to check how close the transaction price is concerning a predicted current transaction price by a Long Short-Term Memory model



(a) Heatmap of features extracted from Ethereum transactions

4.2 Model Creation

The two main steps for detecting frontrunning attacks in Ethereum and the machine learning models used in those steps are as follows:

(i) **Prediction of Gas Price of the current transaction using previous transaction's gas price values**: We used a Long Short-Term Memory (LSTM) [23] model with a depth of 1 and 100 neurons. It is used to predict the gas price of the next normal transaction using the gas price of the previous twelve normal transactions, i.e., for computing Feature (vi) mentioned in the last section. We also used the transformer model for gas price prediction and observed similar results to the LSTM model. We use the LSTM model from the Keras library [9]. It was trained using transactions from the latest 100 Ethereum blocks. The model uses the last fifteen transactions to predict the gas price of the next transaction.

(ii) **Classifying the transactions as Attack or Not Attack transaction**: For the binary classification task we used multiple models. Initially, we used the multilayer perceptron (MLP) model. Later on, we also used decision-tree-based models such as XGBoost, Isolation Fores and Random Forest to compare the results. We found out that the tree-based models gave better precision values than the MLP model. The correlation between a feature vector and the maliciousness of a transaction is not anticipated to be linear. Therefore, we opt to utilise a multi-layer perceptron (MLP) Haykin (1994), to acquire knowledge of this mapping. This choice is mostly since MLP is considered one of the most straightforward and adaptable nonlinear classifiers. In addition, it can simulate intricate non-linear functions, perform effectively with substantial quantities of data, and, most significantly, deliver rapid predictions once it has undergone training. The MLP model we have developed comprises a single input layer and two hidden layers that come before the output layer. Every feature is explicitly inputted into the input layer, with the exception of the seventh component, which represents the expected gas price. This feature is the result of an LSTM model that utilises the gas price of the previous ten non-attack transactions as its input and generates the forecasted gas price for the subsequent non-attack transaction.

In this study, the sigmoid activation function has been utilised as the non-linear transfer function for each neuron. We employed the Scikit-Learn library Pedregosa et al. (2011) in Python to construct the model. The dataset supplied in Torres et al. (2021) was utilised for model training. Features were retrieved from this dataset following the described approach. We conducted experiments with Multilayer Perceptrons (MLPs) using various configurations of neurons and hidden layers to determine the most effective hyperparameters.

Although the MLP classifier was giving good accuracy, it was giving very bad precision, recall and f1-score values as shown in Figure 3. This was mainly due to the imbalanced dataset which had a huge number of 'Not Attack' labelled transactions compared to transactions labelled as 'Attack'. It is also not possible to have class weights while training using the MLP classifier. To cater to these issues we used XGBoost which is a powerful and widely used gradient-boosting library for machine learning. It is known for its high predictive power and efficiency.

There are a few strategies we can use to handle imbalanced datasets when using XGBoost. These include:

1. **Upsampling the minority class**: This involves generating synthetic samples for the minority class to balance the dataset. One way to do this is by using the Synthetic Minority Oversampling Technique (SMOTE), which uses a nearest neighbors algorithm to create synthetic samples that are similar to the minority class. While this can improve model performance, it may also lead to overfitting if not done carefully.

2. **Downsampling the majority class**: This involves removing samples from the majority class to balance the dataset. This can help prevent the model from being biased towards the majority class, but it may also reduce the overall size of the dataset and potentially reduce model performance.

3. **Using class weights**: XGBoost allows us to specify class weights, which adjust the importance of each class during training. By increasing the weight of the minority class, we can give it more influence on the model's predictions. We can either specify the class weights manually or use the "scale_pos_weight" parameter, which is calculated automatically based on the class balance.

Using the XGBoost classifier and the above strategies for unbalanced data we got much better precision, recall and f1-score values. We also did hyperparameter tuning for the XGBoost model to get better results. We then moved on to more tree-based classification models such as random forest and isolation forest to check which model best fits what type of frontrunning attacks.

Chapter 5

Implementation and Evaluation

5.1 Dataset Description

We utilise the dataset on frontrunning attacks as presented in reference Torres et al. (2021). We took more than 16,000 frontrunning attack transactions, encompassing displacement, insertion, and suppression categories.

By employing several heuristics specific to each sort of frontrunning attempt, these transactions have been eliminated after analysing more than 11 million blocks. The process of creating the dataset may have failed to identify certain attack transactions due to the highly stringent criteria, however, there are no instances of incorrect identification of non-attack transactions. In addition, this dataset includes various additional information about each transaction, such as the block number and petrol price.

To create labelled data for the classification task we scan through the block numbers of attack transactions and then through all the transactions in each of the blocks. All the transactions present in those blocks and not in the dataset are labelled as 'Not Attack'. Using this method we created labelled data consisting of more than 9,00,000 transactions.

5.2 Feature Extraction

We utilised Infura to access the Ethereum main network and collect transaction attributes. Infura is a service that enables users to connect with the Ethereum blockchain without the need to set up their own nodes. To examine the block data, we utilised Web3Py, a Python program that allows us to extract and selectively process the data contained within the blockchain. Each transaction in our dataset is accompanied by its corresponding block number. Therefore, we utilise this data to retrieve the characteristics such as the petrol price, as well as the average and variability of the petrol price in the previous 10 blocks by iterating through the corresponding blocks using Web3Py. To obtain features such as the mean and standard deviation of petrol prices in transactions made by the same externally owned account (EOA), we analyse the previous blocks and extract these features. In order to obtain the projected petrol price, we developed the LSTM model as previously indicated.

5.3 Experimental Results

The accuracy of the different models presented in Sub-section 4.2 for the three kinds of frontrunning attacks is shown in the Tables given below. We can see although the accuracy is very high for the MLP classifier the precision, recall and f1-scores are very low. This is due to the inability of the MLP classifier to deal with imbalanced data.

We then XGBoost and Random Forest Classifiers work very well with the data. The Isolation Forest Classifier has better recall and f1-score values than MLP but worse accuracy and precision values.

Figure 5.1 shows the plots for hyperparameter tuning of the XGBoost classifier. Figure 5.2 shows Precision-Recall curves and Figure 5.3 shows ROC curves.

TABLE 5.1: MLP Classifier Report

	Displacement(%)	Insertion(%)	Suppression(%)
Accuracy	98.12	95.44	94.40
Precision	2.34	1.15	2
Recall	1.31	1.93	8.33
f1-score	1.68	1.45	1.17

TABLE 5.2: XGBoost Classifier Report

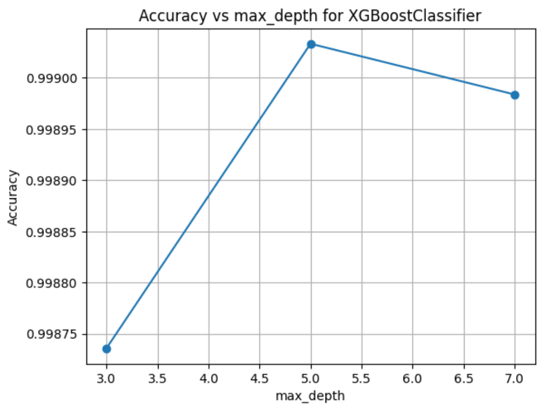
	Displacement(%)	Insertion(%)	Suppression(%)
Accuracy	99.94	99.81	98.13
Precision	99.83	99.44	100
Recall	95.30	89.72	58.33
f1-score	97.51	94.33	73.68

TABLE 5.3: Random Forest Classifier Report

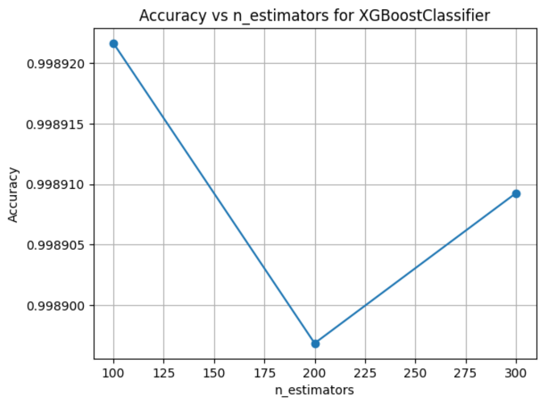
	Displacement(%)	Insertion(%)	Suppression(%)
Accuracy	99.96	99.88	98.13
Precision	100	97.18	100
Recall	97.22	96.14	58.33
f1-score	98.59	96.66	73.68

TABLE 5.4: Isolation Forest Classifier Report

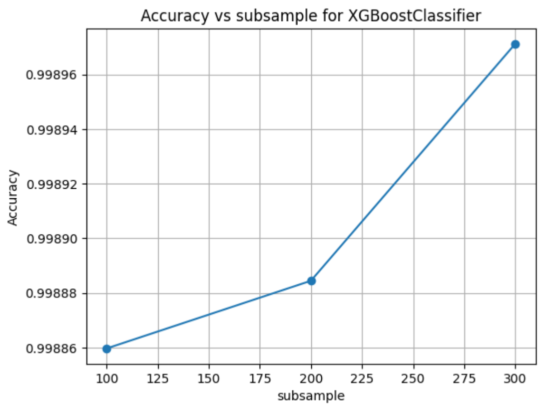
	Displacement(%)	Insertion(%)	Suppression(%)
Accuracy	75.88	74.73	94.02
Precision	0.93	0.67	37.50
Recall	17.87	9.20	50
f1-score	1.78	1.24	42.85



(a) max_depth parameter

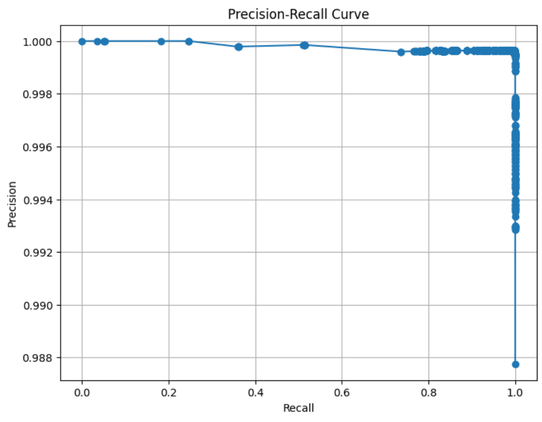


(b) n_estimators parameter

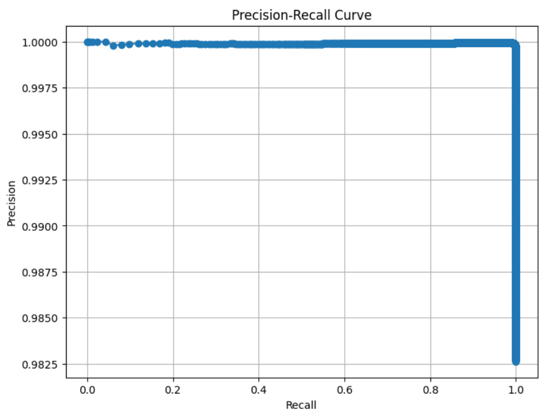


(c) subsample parameter

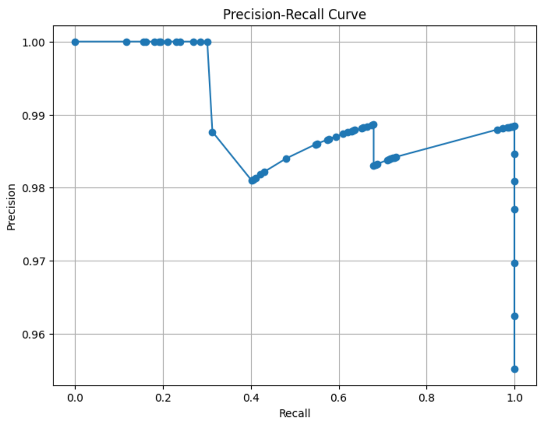
FIGURE 5.1: Plots showing hyperparameter tuning results for XGBoost classifier



(a) Displacement Attacks

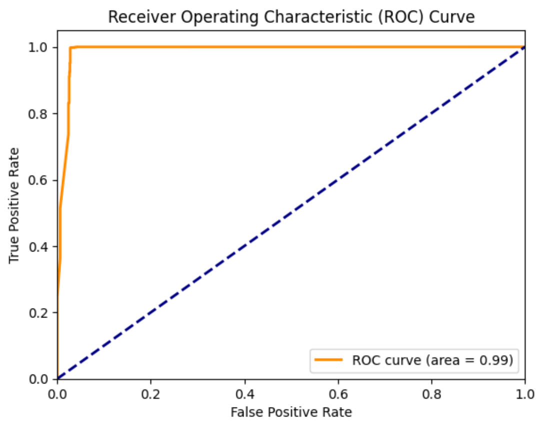


(b) Insertion Attacks

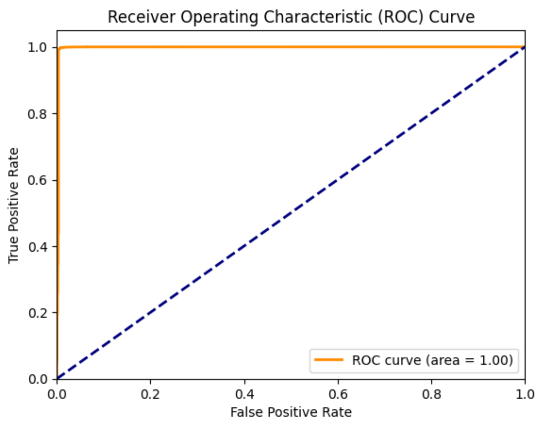


(c) Suppression Attacks

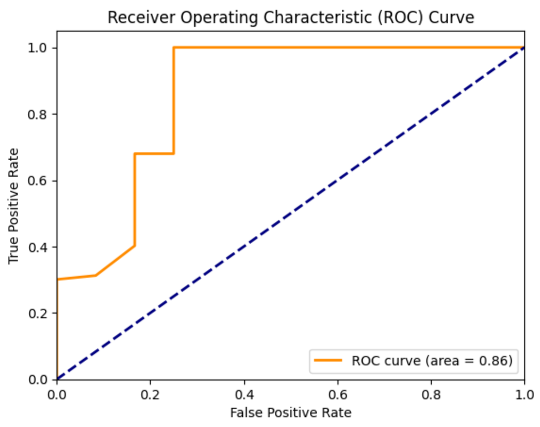
FIGURE 5.2: Precision-Recall curve for different Frontrunning Attacks



(a) Displacement Attacks



(b) Insertion Attacks



(c) Suppression Attacks

FIGURE 5.3: ROC curve for different Frontrunning Attacks

Chapter 6

Conclusion

In this project, we proposed machine-learning based models for detecting frontrunning attacks on Ethereum transactions which is an extension to the study made by this paper Maddipati Varun and Sural (2022). We compared the MLP classifier and tree based classifiers for the task of detection of attack transactions. To employ a machine learning model for identifying attack transactions, it is necessary to extract features that accurately reflects a transaction.

In addition, when attempting to identify these attacks in a real-time setting, we selected characteristics that can be easily gathered from the transactions within a smart contract. This ensures that the model can be efficiently implemented on the blockchain using a smart contract.

A direction of future work is to integrate the classifier models into smart contracts to implement live detection and prevention of frontrunning attacks in Ethereum transactions.

Bibliography

Eskandari, S., Moosavi, M., and Clark, J. (2019). Sok: Transparent dishonesty: Front-running attacks on blockchain. In *International Conference on Financial Cryptography and Data Security*, pages 170–189, Berlin. Springer.

Flovik, V. (2018). How to use machine learning for anomaly detection and condition monitoring. [Online; accessed 15-September-2021].

Gai, K., Wu, Y., Zhu, L., Qiu, M., and Shen, M. (2019). Privacy-preserving energy trading using consortium blockchain in smart grid. *IEEE Transactions on Industrial Informatics*, 15(6):3548–3558.

Gai, K., Wu, Y., Zhu, L., Zhang, Z., and Qiu, M. (2020). Differential privacy-based blockchain for industrial internet-of-things. *IEEE Transactions on Industrial Informatics*, 16(6):4156–4165.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation.* Prentice Hall PTR, New Jersey, USA.

Kokoris-Kogias, E., Alp, E. C., Gasser, L., Jovanovic, P., Syta, E., and Ford, B. (2018). Calypso: Private data management for decentralized ledgers. *Cryptology ePrint Archive*, 14(4).

Maddipati Varun, B. P. and Sural, S. (2022). Mitigating frontrunning attacks in ethereum. *BSCI '22*, pages 115–124.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Su, L., Shen, X., Du, X., Liao, X., Wang, X., Xing, L., and Liu, B. (2021). Evil under the sun: Understanding and discovering attacks on ethereum decentralized applications. In *30th USENIX Security Symposium*, pages 1307–1324, USA. USENIX Association.

Torres, C. F., Camino, R., and State, R. (2021). Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the ethereum blockchain. *CoRR*, abs/2102.03347:1–17.

Wang, D., Wu, S., Lin, Z., Wu, L., Yuan, X., Zhou, Y., Wang, H., and Ren, K. (2020). Towards understanding flash loan and its applications in defi ecosystem. *CoRR*, abs/2010.12252:1–6.

Zhou, Y., Han, M., Liu, L., He, J., and Wang, Y. (2018). Deep learning approach for cyberattack detection. In *Proceddings of IEEE InFOCOM Workshop*, pages 262–267, USA. IEEE.

Zhu, L., Wu, Y., Gai, K., and Choo, K.-K. R. (2018). Controllable and trustworthy blockchain-based cloud data management. *Future Generation Computer Systems*, 91.