Fighting Under-price DoS Attack in Ethereum with Machine Learning Techniques

José Eduardo A. Sousa
UFJF - DCC, Brazil
eduardo2@ice.ufjf.br

Vinicius C. Oliveira
UFJF - DCC, Brazil
vinicius.oliveira@ice.ufjf.br

Júlia Almeida Valadares
UFJF - DCC, Brazil
juliavaladares@ice.ufjf.br

Alex Borges Vieira
UFJF - DCC, Brazil
alex.borges@ufjf.edu.br

Heder S. Bernardino
UFJF - DCC, Brazil
heder@ice.ufjf.br

Saulo Moraes Villela
UFJF - DCC, Brazil
saulo.moraes@ufjf.edu.br

Glauber Dias Gonçalves
UFPI - CSHNB, Brazil
ggoncalves@ufpi.edu.br

ABSTRACT

Ethereum is one of the most popular cryptocurrency currently and it has been facing security threats and attacks. As a consequence, Ethereum users may experience long periods to validate transactions. Despite the maintenance on the Ethereum mechanisms, there are still indications that it remains susceptible to a sort of attacks. In this work, we analyze the Ethereum network behavior during an *under-priced DoS attack*, where malicious users try to perform denial-of-service attacks that exploit flaws in the fee mechanism of this cryptocurrency. We propose the application of machine learning techniques and ensemble methods to detect this attack, using the available transaction attributes. The proposals present notable performance as the Decision Tree models, with AUC-ROC, F_β-score and recall larger than 0.94, 0.82, and 0.98, respectively.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Experimentation, Performance, Reliability, Security

Keywords

Blockchain, Ethereum, Attacks, DoS, Machine Learning

1. INTRODUCTION

Ethereum is currently a popular cryptocurrency, totalling a market capitalization of over 20 billion dollars up to April 2020[1]. Besides monetary transactions, its platform allows the execution of smart contracts, which are self-executable programs with predefined rules. This rich set of functionalities makes Ethereum more susceptible to different types of threads and attacks [3]. Denial of Service (DoS) attacks are one of the biggest challenges cryptocurrencies systems face [8] once these attacks impact the user experience. In this type of attack, several malicious users trigger actions to the system, keeping it busy. In this sense, the cryptocurrency system is not able to properly serve its honest users. Despite the payment of fees to carry out a transaction in the majority of cryptocurrencies, malicious users have already exploited this mechanism to apply DoS attacks [4]. The Ethereum platform, in particular, has also suffered attacks that exploited low fee instructions and flaws in the virtual machine causing network congestion and losses of contracts and transactions.[2] Despite the impact on cryptocurrencies systems, there is still a lack of academic studies analyzing attacks on such platforms.

In this work, we analyze how the Ethereum network behaves during a DoS attack. More precisely, we evaluate the *Under-priced DoS Attack* [4], where attackers exploit the Ethereum fee mechanism, paying a negligible fee for a large number of small-value transactions. Then, we propose an ensemble of machine learning techniques to detect this attack, using the publicly available transaction attributes.

Some works [4, 8, 5] rely on counter attack measurements that alter the blockchain architecture. For example, they propose increasing the block size, the fee value, or the computational cost of the consensus mechanism to minimize the impact of attacks. Our approach, as Baqer et al. [1], tries to detect suspicious transactions and then, one can effectively apply any existing counter measurement. However, while in [1] the authors cluster the transactions, in this work, we rely on a set of machine learning techniques.

We have evaluated our proposal simulating the Ethereum network based on real traces. Our results show that the *Under-priced DoS Attack* can increase the average pending time of a transaction by more than 42%. Moreover, the proposed use of machine learning techniques presents considerable performance. It can distinguish between genuine and *under-priced DoS attack* transactions with AUC, F_β-score and recall larger than 0.94, 0.82, and 0.98, respectively.

[1] `https://etherscan.io`

[2] Ethereum Network Comes Across Yet Another DoS Attack. `www.newsbtc.com/2016/09/23/ethereum-dao-attack-attack-platforms-credibility`

Symposium on Cryptocurrency Analysis (SOCCA) 2020 Milan,Italy

2. RELATED WORK

Cryptocurrencies systems are targets of attacks since their inception. For example, the slowdown of Bitcoin network has been studied since 2017, when malicious peers flooded the system with a large volume of transactions [6]. Existing solutions mostly rely on simplistic measurements, such as increasing the block size or the fee value, increasing the computational cost of the consensus mechanism, or the creation of parallel blockchains to offload the main chain.

For example, Ting Chen et al. [4] evaluated DoS attacks that explore low-cost operations of transactions and smart contracts. They analyzed the causes and effects of the attack, and proposed an adjust to the network gas cost mechanism. Saad et al. [8] also studied DoS attacks, focusing on Bitcoin. They proposed countermeasures at the transaction queue level that prioritize transactions based on miner's reward rates, with an aging mechanism, preferring older transactions in the queue. One can note that DoS attacks are not exclusive to Bitcoin and Ethereum networks and have also occurred in other blockchain systems as Monero[3], e.g., Chervinski et al. [5] characterized and evaluated the impact of *under-priced DoS attack* in this system.

To detect transactions involved in an attack, Baqer et al. [1] evaluated Bitcoin network and grouped transactions, using the K-Means method. In that work, the authors also evaluated the costs to perform an attack, based on the miners' reward to miners and the volume of transactions required to establish the attack.

In sum, differently from existing works, we analyze the effects of *under-priced DoS attack* on Ethereum and we present an ensemble of machine learning techniques to detect this attack by classifying suspicious transactions, using the publicly available information. As soon an attack is detected, one can use any existing counter measurement, mitigating the effects of the attack.

3. EVALUATION ENVIRONMENT AND METHODOLOGY

In this work, we first analyze how the Ethereum network behaves given an *under-priced DoS attack*. Our analysis rely on Ganache,[4] a tool that allows one to quickly fire up a personal Ethereum blockchain to run tests, execute commands, and inspect state while controlling how the chain operates. Figure 1 illustrates the architecture of the controlled environment we have designed to simulate the Ethereum network. This architecture is organized into four components: (1) database, (2) network startup, (3) control and (4) communication. These components communicate to simulate transactions on the network of our controlled environment. The arrows indicate the dynamics of the simulation, given the data flow between two components.

The first step of the simulation occurs through the database component, which provides the transactions to be processed in the network. The database can provide real Ethereum transactions extracted from the service APIs as well as synthetic transactions, which features such as fee, value, source and destination are defined in a personalized way.

The second step uses the Ganache tool which is responsible for providing the Ethereum service. We set up a network

[3]https://www.getmonero.org

[4]https://www.trufflesuite.com/ganache

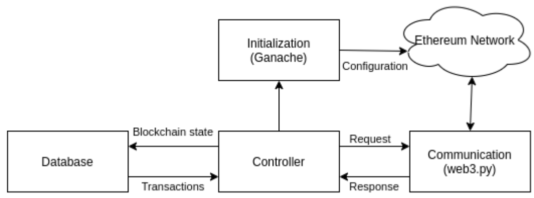

Figure 1: Overview of the architecture for our controlled environment to run simulations.

with the time between blocks of 13 seconds, which is the Ethereum's last year average inter block time[5]. We set up 2000 accounts with a balance in the network, so each address in the database can be mapped into a unique address in the simulation. Given these settings, the control triggers a command for Ganache to generate a new instance of the private Ethereum network. As shown in Figure 1, Controller is the main component of the proposed architecture and allows us to monitor all stages of the simulations. In particular, the control maps users extracted from real transactions to users of the network initialized in the simulation. Additionally, the controller defines the number of transaction injection per period, monitors the transaction queue (*mempool*), and requests the creation of new transactions and smart contracts. The communication component, in turn, uses the API *web3.py* to request the execution of transactions to the network as well as provide the response of these executions.

We carry out simulation rounds with a period of 15 minutes, where the attack occurs between the 5th and 10th minutes. We have removed the five minutes of warmup and five minutes of cool-down periods from our simulations. We reproduced three scenarios in our network simulations: (i) a normal flow with 2000 Ethereum transactions, (ii) *under-priced DoS attack* with the same transactions as the normal flow, but adding 10% of the malicious transaction flow, and (iii) the *Ethereum Boom* with transactions of December 20, 2017, where a surge of transactions was observed. For the first and second scenarios, we use the transactions of May 5, 2019, between 08:22 and 08:37. The gas values, gas prices, and values of the actual transactions are replicated in the three scenarios of the simulation.

Table 1: ETH transactions attributes.

Attribute	Description
hash	Unique, fixed-length text using all transaction data.
sender	Transaction sender.
recipient	Transaction recipient.
value	Amount of Ether transferred from the sender to the recipient.
gas	Unit of computational effort required to successfully execute a transaction.
gas limit	Computational effort unit offered to execute a transaction.
gas price	Amount in Ether paid per gas.
pending time	Interval between the observation of a transaction in the mempool and its inclusion in the blockchain.

We evaluate the Ethereum network under attack using actual blockchain data to mimic genuine Ethereum clients. These data have been collected in [9], and they are available publicly. Moreover, we use synthetic transactions as a way to reproduce consistent attack behavior to the network. We use the transactions features shown in Table 1 to reproduce the behavior of the network in our simulations. In the case of an attack (scenario ii), we set up the gas price to 1 GWei

[5]https://etherscan.io/chart/blocktime

(10^{-9} Ether), which is a low gas price value, as observed in real transactions. Note that, in the simulation, we include transactions in the mempool according to the time observed in the dataset, whereas the pending time feature as defined in Table 1 is driven by the dynamic of the simulation.

4. DETECTING THE UNDER-PRICED DOS ATTACK

In this work, we propose the use of machine learning techniques and ensemble methods to detect the *under-priced DoS attack*. The dataset is composed of transactions of our simulation during the attack period and the following features are used: gas, gas price, value, and pending time. These features are described in Table 1.

We adopt machine learning approaches commonly found in the literature: Decision Tree, Random Forest, KNN, SVM, and Naïve Bayes classifiers. In addition, we consider the use of committees, i.e., sets of classifiers which jointly define the classification, based on community decision. The joint decision of a committee usually allows the combination of the advantages of different classifiers, eliminating their weaknesses and then leading to more robust solution. In a committee, the predictions for each instance are made based on a voting criterion, which can be defined as (i) majority vote (hard) and (ii) average confidence (soft). The classification in the first model is given by the most voted class among the participating committee models. The second considers the certainty that the models have that a transaction belongs to a class and, thus, presents an answer based on an average.

The transactions of the dataset are imbalanced, as only 10% of the instances represent attack situations. Thus, we consider a *positive* class as an attack transaction (minority class) and the *negative* class as a non-malicious transaction (majority class). We deal with the imbalanced instances using: (i) undersampling, which stands for reducing the number of majority class (non-malicious transactions), (ii) oversampling, generating more instances of the minority class by replicating its values, and (iii) SMOTE [2], an oversampling technique that generates synthetic data.

The data is split into two groups: training and test datasets. The training dataset (70% of the transactions) is used to create the classifier models. On the other hand, the generated models are evaluated with respect to the test dataset (30% of the transactions). We evaluate these models according to accuracy, precision, recall, F_1-score, F_β-score with $\beta = 2$, number of true positives (TP) and negatives (TN), and area under the ROC curve (AUC-ROC). It is worth noting that accuracy is not a proper performance metric when data is imbalanced. On the other hand, F_β-score and AUC-ROC are proper performance metrics when data is imbalanced.

5. EVALUATION

5.1 Analysis of results during under-priced DoS attack

In this section, we evaluate the impact of *under-priced DoS attack* in the Ethereum network. We compare the pending time for Ethereum in the three previous defined scenarios. Figure 2 shows the cumulative probability distribution of the pending time for each scenario. At a glance, the Ethereum Boom scenario presents higher pending time, when compared to the other scenarios. Clearly, on the other hand, the Ethereum network under regular flow presents the lower pending times. For instance, 60% of all transactions of a regular Ethereum system (where we do not notice an attack) do not experience pending times higher than 19.5 seconds at the 50th percentile. When the system is under attack, practically 50% of all transactions experience pending times higher than 50 seconds. In this same situation, in the Boom scenario, the pending times are practically 4 times larger and practically 50% of all transactions experience pending times higher than 200 seconds.

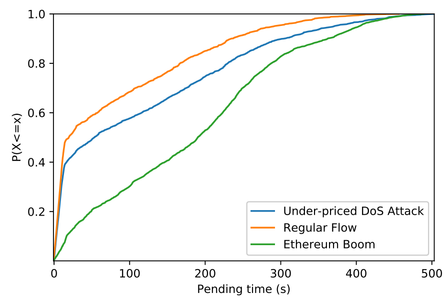

Figure 2: CDF of the pending time.

As shown in Table 2, the *under-priced DoS attack* increases the mean pending time by 42.16%, while the period of the greatest use of the network (the Ethereum Boom) increases this value by 131.90%. The transactions during a normal flow are statistically different from transactions during a *under-priced DoS attack* (i.e., conf. intervals do not interleave, for a 95% confidence level). The Kruskal-Wallis test confirms this difference between the normal transactions and transactions during an attack. In this case, we obtain the Kruskal-Wallis test of 60.7 and a p-value close to zero.

Table 2: Pending time details in seconds.

	Mean	Confidence Interval	STD	Mode
Regular flow	79.56	75.15 - 83.96	100.37	19.55
Under-priced DoS attack	113.11	107.96 - 118.27	126.06	53.54
Ethereum boom	184.50	180.69 - 188.32	122.98	190.09

5.2 Analysis of Machine Learning Models

Here, we evaluate the efficiency of the machine learning techniques to detect the *under-priced DoS attack*. Due to space constraints, we only consider the second scenario defined. Scikit-learn [7] is a machine learning library and is adopted here for the experiments. For the Decision Tree and Random Forest models, which are tree-based classifiers, we predefined the maximum tree depth equals to three and the optimization criterion as "gini." Particularly for the Random Forest, we defined the number of trees equals to 100. Default values were used for the other parameters.

Table 3 presents the performance of the machine learning models and balancing strategies considered in this work.

The Random Forest performed better than the other techniques for all tested balancing strategies with respect to accuracy, precision, F_1-score, and TN values. For example, when the oversampling technique is used, Random Forest reaches accuracy, precision, F_1-score, and TN values larger than 0.95, 0.75, 0.78, and 1949, respectively.

Table 3: Machine learning methods results using different balancing techniques.

BT	Classifier	Accuracy	Precision	Recall	F_1-score	F_β-score	TP	TN	AUC-ROC
Undersampling	Decision Tree	0.891818	0.454756	**0.984925**	0.622222	0.798696	**196**	1766	0.933742
	Random Forest	**0.915000**	**0.515957**	0.974874	**0.674783**	**0.827645**	194	**1819**	**0.941960**
	KNN	0.686364	0.215527	0.934673	0.350282	0.560579	186	1324	0.798171
	Gaussian SVM	0.614091	0.178854	0.909548	0.298927	0.500553	181	1170	0.747128
	Naive Bayes	0.871364	0.411392	0.979899	0.579495	0.767717	195	1722	0.920235
	Soft Committee	0.907273	0.493606	0.969849	0.654237	0.812974	193	1803	0.935449
	Hard Committee	0.910000	0.501292	0.974874	0.662116	0.819949	194	1808	0.939211
Oversampling	Decision Tree	0.911818	0.506460	**0.984925**	0.668942	**0.828402**	**196**	1810	**0.944736**
	Random Forest	**0.959545**	**0.757009**	0.814070	**0.784504**	0.801980	162	**1949**	0.894042
	KNN	0.872727	0.398496	0.798995	0.531773	0.665272	159	1761	0.839527
	Gaussian SVM	0.685909	0.216590	0.944724	0.352390	0.564904	188	1321	0.802447
	Naive Bayes	0.860909	0.392354	0.979899	0.560345	0.754060	195	1699	0.914487
	Soft Committee	0.915909	0.518919	0.964824	0.674868	0.823328	192	1823	0.937934
	Hard Committee	0.913182	0.510582	0.969849	0.668977	0.821976	193	1816	0.938698
SMOTE	Decision Tree	0.912727	0.509091	**0.984925**	0.671233	**0.829805**	**196**	1812	**0.945236**
	Random Forest	**0.953182**	**0.703390**	0.834171	**0.763218**	0.804264	166	**1931**	0.899594
	KNN	0.868182	0.388206	0.793970	0.521452	0.656692	158	1752	0.834766
	Gaussian SVM	0.690455	0.218458	0.939698	0.354502	0.565981	187	1332	0.802683
	Naive Bayes	0.860909	0.391039	0.964824	0.556522	0.745921	192	1702	0.907699
	Soft Committee	0.913636	0.512064	0.959799	0.667832	0.816938	191	1819	0.934422
	Hard Committee	0.913182	0.510695	0.959799	0.666667	0.816239	191	1818	0.934172

On the other hand, Decision Tree presents the best performance among strategies with respect to recall, F_β-score, TP, and AUC-ROC. The exceptions occur for F_β-score and AUC-ROC when using undersampling, where Random Forest achieved the best values. The best values of recall, F_β-score, TP, and AUC-ROC were obtained by Decision Tree and are larger than 0.98, 0.82, 196, and 0.94, respectively. In general, one can conclude that Decision Tree can identify the malicious transactions better than the other approaches.

In general, both the soft and hard committees found good results when compared to the other classifiers. Despite this good performance, they did not present the best values in any case. SVM achieved the worst results in general. The oversampling and SMOTE generated the best results in most of the cases. Oversampling provided the best accuracy, precision, F_1-score, and TN, while the best F_β-score and AUC-ROC were obtained when using SMOTE.

6. CONCLUSIONS

In this work, we analyze the Ethereum network behavior during a *under-priced DoS attack* and propose the application of machine learning techniques to detect this attack. From simulations driven by real data, we show that the *under-priced DoS attack* has the power to increase the pending time of the transactions in Ethereum. Additionally, we note that the high demand for transactions per second (e.g., the Ethereum Boom) can generate a transaction rate slower than a DoS attack. Ultimately, we show that the application of machine learning techniques is able to detect *under-priced DoS attack* with considerable performance. Decision Tree and Random Forest, in particular, have reached the highest performance for the majority of the evaluated measures. As for future work, we intend to jointly validate our proposal with existing counter measurements to mitigate *under-priced DoS attack*.

Acknowledgements: We acknowledge the partial support of CAPES, CNPq, FAPEMIG.

7. REFERENCES

[1] K. Baqer, D. Y. Huang, D. McCoy, and N. Weaver. Stressing out: Bitcoin "stress testing". In *International Conference on Financial Cryptography and Data Security*, pages 3–18. Springer, 2016.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-Sampling Technique. *Int. J. Artif. Intell. Res.*, 16:321–357, 2002.

[3] H. Chen, M. Pendleton, L. Njilla, and S. Xu. A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defenses. *ACM CSUR*, 53(3):1–43, 2020.

[4] T. Chen, X. Li, Y. Wang, J. Chen, Z. Li, X. Luo, M. H. Au, and X. Zhang. An Adaptive Gas cost Mechanism for Ethereum to Defend Against Under-Priced DoS Attacks. In *ISPEC*, 2017.

[5] J. O. M. Chervinski, D. Kreutz, and J. Yu. FloodXMR: Low-Cost Transaction Flooding Attack with Monero's Bulletproof Protocol. *IACR Cryptology ePrint Archive*, page 455, 2019.

[6] D. Gerard. *Attack of the 50 foot blockchain: Bitcoin, blockchain, Ethereum & smart contracts.* 2017.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *The Journal of machine Learning research*, 12:2825–2830, 2011.

[8] M. Saad, L. Njilla, C. Kamhoua, J. Kim, D. Nyang, and A. Mohaisen. Mempool Optimization for Defending Against DDoS Attacks in PoW-based Blockchain Systems. In *IEEE ICBC*, 2019.

[9] J. E. d. A. Sousa, V. Oliveira, J. Valadares, A. B. Vieira, S. M. Villela, H. S. Bernardino, and G. Dias. An Analysis of the Fees and Pending Time Correlation in Ethereum. *IJNM nem.2113*, 2020.