A Deep Learning Approach for Detecting Security Attacks on Blockchain*

Francesco Scicchitano[1], Angelica Liguori[1], Massimo Guarascio[1], Ettore Ritacco[1], and Giuseppe Manco[1]

ICAR-CNR, Via P. Bucci, 8/9c, Rende, Italy
{name.surname}@icar.cnr.it

Abstract

In these last years, Blockchain technologies have been widely used in several application fields to improve data privacy and trustworthiness and security of systems. Although the blockchain is a powerful tool, it is not immune to cyber attacks: for instance, recently (January 2019) a successful 51% attack on Ethereum Classic has revealed security vulnerabilities of its platform. Under a statistical perspective, attacks can be seen as an anomalous observation, with a strong deviation from the regular behavior. Machine Learning is a science whose goal is to learn insights, patterns and outliers within large data repositories; hence, it can be exploit for blockchain attack detection.

In this work, we define an anomaly detection system based on a encoder-decoder deep learning model, that is trained exploiting aggregate information extracted by monitoring blockchain activities. Experiments on complete historical logs of Ethereum Classic network prove the capability of the our model to effectively detect the publicly reported attacks. To the best of our knowledge, our approach is the first one that provides a comprehensive and feasible solution to monitor the security of blockchain transactions.

Keywords – *Blockchain; Anomaly detection; Attack detection; Autoencoders; Sequence to sequence models; Encoder-decoder models;*

1 Introduction

The *Blockchain* is largely considered an effective solution able to ensure security and trustworthiness. Nevertheless, it can be prone to attacks and security threats, as discussed in [21]. In particular, *Ethereum Classic* (ETC), a permissionless (public) blockchain-based decentralized platform for *smart contracts* [5], has recently experienced two significant attacks which compromised the functionality of the network [1].

Roughly, the blockchain is characterized by a global ledger that can record transactions efficiently and permanently through a timed sequence of *blocks*, called *chain*. Blocks contain information about the transactions, and each block is added to the chain after a validation process, based on a distributed consensus mechanism. Consensus is reached whenever a sufficient number of nodes validates the block, that is then considered trustworthy. This whole process is logged and it is possible to collect information characterizing the activities within the underlying ledger. It is natural to ask whether such information can be exploited to monitor the process and provide early analysis and detection mechanisms capable of reporting anomalous situations and prospective attacks.

Historically, data analysis techniques have been extensively exploited in the cyber security domain [4], and the recent diffusion of advanced machine learning techniques has allowed to

accurately identify cyber-attacks and detect threats, both real-time and in post-incident analysis [20, 15]. Notably, both supervised and unsupervised machine learning algorithms have been successfully employed to support intrusion detection and prevention systems, as well as to detect system misuses and security breaches. The scenarios of interest are usually characterized by a continuous stream of data (such as packet-level or application-level data) summarizing the behavior of the underlying network or system. The role of the machine learning algorithms consists either in identifying known attacks (supervised approach), or anomalous behavior (unsupervised approach). Anomaly-based techniques are able to fit normal system operating status, isolating and identifying anomalies as unexpected behavioral deviations. For this reason, anomaly detection approaches result to be appealing for their ability to detect zero-day attacks, i.e., attacks exploiting unknown vulnerabilities.

In this paper, we propose an encoder-decoder model that exploits the information collected by the activities within a blockchain, to highlight anomalies in the underlying network activity, that can represent a potential symptom of a forthcoming or current attack on the system integrity. The contribution of the paper is twofold:

- We identify a set of properties to be computed from the blockchain logs, which describe the status of the system at every time step.
- Next, we devise an unsupervised neural architecture, which takes as input a time series representing the status of the blockchain network for a time window, able to compute a score representing the degree of anomaly exhibited by the time series.

Experiments on the historical logs of the ETC network show that the system is effective in detecting the reported attacks. To the best of our knowledge, our approach is the first one that provides a comprehensive and feasible solution to monitor the security of blockchain transactions.

The paper is organized as follows. In section 2 we review the architecture of the ETC network and describe the attacks that exposed its potential vulnerabilities. We also review how machine learning and blockchain technology have been combined in the current literature. Section 3 provides two contributions: first, it gives details about the structure of the ETC logs and the preprocessing steps applied to engineer the relevant features describing the blockchain status. Next, we discuss about the details of the Encoder/Decoder architecture. Section 4 describes the experiments that demonstrate the effectiveness of the proposed architecture, and finally in section 5 we discuss open problems and pointers to future directions.

2 Background and Related Work

The blockchain is a *Distributed Ledger Technology* [9], i.e. a system that allows to store exchanges and information in a secure and permanent way. Given the structure and protocol, blockchain removes the need to have intermediaries, who were previously required to act as trusted third parties to verify, register and coordinate data.

From a technical perspective, a blockchain is a distributed database shared into a peer-to-peer network: data are stored under the form of transactions within specific structures named blocks. Each block is the collection of all transactions performed in a time window and different blocks may contain a different number of transactions. Blocks are connected by a cryptographic bond in the chain: a hash function is computed on the concatenation of the *signature* of the current block with the hash computed on the previous one. By repeating this step for each block the chain is created and the references to the previous block are maintained, as shown in

Figure 1. A not injective (non-invertible) mathematical function is employed to generate the hash. The chain is immutable and can only be extended by adding new blocks: any modification on a block generates a different hash, thus making all subsequent blocks inconsistent.

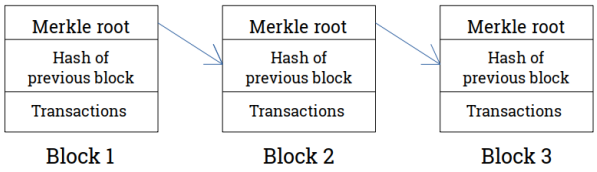


Figure 1: The blockchain structure.

The security is guaranteed by a consensus mechanism that allows to confirm transactions and produce new blocks. The most popular consensus algorithm is named "Proof-of-Work" (PoW), first adopted in the Bitcoin blockchain [16]: special nodes in the network, called *miners*, compete in validating the blocks, and only one will produce the final validation.

PoW is considered a robust method, able to reach the Byzantine-fault-tolerant consensus [14] and to minimize the risk for a Double Spending attack [18]. In fact, any modification to the ledger would require to re-validate all the blocks between the altered block and the current one, before any other validation: the confirmation is achieved when at least 51% miners agree about the validation.

Originally, blockchain was proposed as a technology to support cryptocurrencies. It later evolved with smart contracts, i.e. computer programs performed in decentralized way on Blockchain so that they can be executed and verified automatically. ETC implements such a technology, by providing a platform for decentralized applications that run and are regulated on a distributed ledger. Each contract is composed of a sequence of operations whose cost is expressed as *gas*, i.e., the amount of cryptocurrency that is deemed necessary to execute them. Actually, ETC implements a priority mechanism of the transaction execution. The priority depends on the *provided gas*, that corresponds to the fee received by the first miner able to compute the solution for the puzzle associated to a block: the higher the provided gas, the higher the priority.

Blockchain is not immune to attacks. The ETC network exhibits some vulnerabilities [7] and has experienced several attacks in the recent years. In particular, [1] reports an attack on 18 June 2016 (referred as DAO from now on), where a vulnerability of the transaction protocol was exploited. More recently, in January 2019 researchers confirmed a successful 51% attack [1] on the ETC blockchain, where hackers were able to roll back transactions. Besides a detailed analysis of the vulnerabilities and countermeasures to improve the robustness of the overall architecture, it is natural to ask whether it is possible to devise an Intrusion or Anomaly detection systems capable of detecting signals of malicious behaviors and issue early warnings. However, to the best of our knowledge, only a limited number of approaches have been proposed in the current literature.

Signorini et al.[19] propose BAD, a blockchain anomaly detection system that collects information from blocks in the main chain as well as orphan blocks (i.e., blocks contained within pruned branches). There are two issues with this approach: first of all the amount of considered information could be infeasible to handle, as they aim at storing data about each branch occurring on the blockchain; second, it would require nevertheless a modification of the protocol, as

[1] See http://cointelegraph.com/news/ethereum-classic-51-attack-the-reality-of-proof-of-work

information concerning the orphan blocks is not currently stored according to the blockchain protocol specifics.

More similar to our approach is the study proposed in [3]. Here, the authors propose a visual analytical approach where statistics are collected from Ethereum public blockchain and visualized through a chronological dashboard. The authors show that the collected statistics allow to detect the DAO attack, by highlighting an anomalous peak close to the corresponding date. Our method extends this approach by proposing a fully automatic and accurate architecture based on deep neural networks.

3 Methodology

In this section we devise a machine learning approach to identify anomalies in the usage of Blockchain-based systems. As discussed before, the approach we propose is essentially unsupervised, due to the lack of labeled data: successful attacks represent very rare events and typically they don't share common patterns. As a consequence, supervised models exhibit poor detection performances.

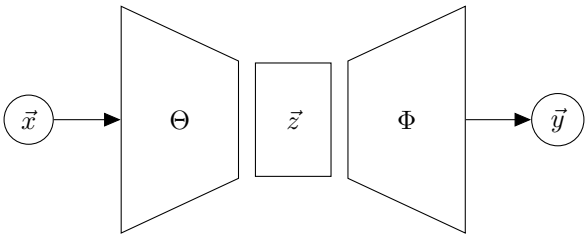
The proposed anomaly detection system builds a neural encoder-decoder model, capable of summarizing the information about the status of the ledger within a latent space and then to rebuild the original information from this space. The underlying intuition is that, whenever the current status is consistent, the encoding/decoding operation preserves the basic properties of the data. By contrast, anomalous situations exhibit inconsistent values that ultimately result in a failing reconstruction. This happens, for example, when the amount of *ether* (the cryptocurrency of the ETC blockchain) is anomalously high compared to all the remaining parameters of the system. An encoder-decoder would interpret this quantity as noise and consequently would ignore such a value in the reconstruction. As a consequence, the difference between the original and the reconstructed values would highlight the anomalous situation and consequently trigger the alert.

The model works on sequences of temporally-sorted events. Formally, we assume that the data is organized as a sequence $X = \{\vec{x}_1, \ldots, \vec{x}_N\}$ relative to a period of observation, where $\vec{x}_t$ is a vector of features describing the t-th event in chronological order within X. An anomaly is an unexpected event in X, i.e. a vector $\vec{x}_t$ significantly different from its neighbors $\vec{x}_{t-w}, \ldots, \vec{x}_{t-1}, \vec{x}_{t+1}, \ldots, \vec{x}_{t+v}$ (for a given window $[t-w, t+v]$ to be determined). Thus, the model should be capable of summarizing all the regularities in the data that characterize the sequence $\vec{x}_{t-w}, \ldots, \vec{x}_{t+v}$.

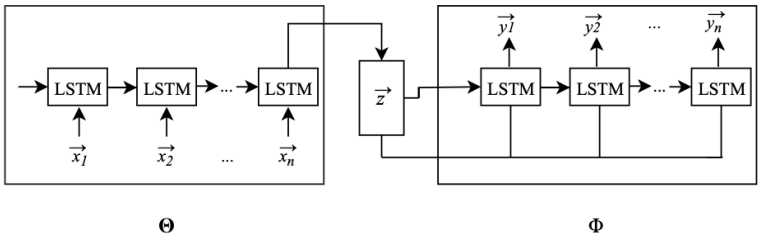
Autoencoders [11, 2] represent an effective solution to the unsupervised task of learning a compact representation of the latent features characterizing a piece of information. An autoencoder is a neural architecture trained to reproduce as output a duplicate of its input. It is composed by two elements, as shown in Figure 2a.

- The encoder Θ, a neural network whose goal is to map an input $\vec{x}$ to a latent compact representation $\Theta(\vec{x}) = \vec{z} \in \mathbb{R}^K$. This mapping produces an embedding of the original input into a latent vector of size K.
- A decoder Φ, another neural networks that, given a K-dimensional vector $\vec{z}$, aims at producing an output $\Phi(\vec{z}) = \vec{y}$ that is as close as possible to the original input.

In the above description, the components Θ, Φ, $\vec{x}$ and $\vec{y}$ are unspecified. In our framework, we consider the input and output $\vec{x}, \vec{y}$ of the autoencoder to be temporally marked events. As a consequence Θ and Φ can be specified as recurrent networks (RNNs) [10]. An RNN naturally fits



(a) General autoencoder structure.



(b) Recurrent Autoencoder (RAE).

sequential data, since, iterating over the sequence, it saves (partial) memory of each event. In our implementation we decided to instantiate the RNNs as Long short-term memory networks (LSTM, see [13] for a detailed description).

The resulting architecture is a sequence to sequence recurrent autoencoder (RAE) model [8] shown in Figure 2b. In short, the objective of RAE is to: (i) read all the events in the sequence, (ii) extract a compact representation of all the events within the sequence; (iii) generate a new sequence that is a copy of (or close to) the input one, by exploiting the representation. Mathematically, this can be specified as follows: given an input sequence $\mathcal{I} = \{\vec{x}_1, \ldots, \vec{x}_n\}$, we compute an output sequence $\mathcal{O} = \{\vec{y}_1, \ldots, \vec{y}_n\}$ where:

$$\begin{aligned} \vec{h}_t^{(e)} &= \texttt{LSTM}_\theta(\vec{x}_t, \vec{h}_{t-1}^{(e)}) \\ \vec{z} &= \texttt{mlp}_\vartheta(\vec{h}_n^{(e)}) \\ \vec{h}_t^{(d)} &= \texttt{LSTM}_\phi(\vec{z}, \vec{h}_{t-1}^{(d)}) \\ \vec{y}_t &= \texttt{mlp}_\varphi(\vec{h}_t^{(d)}) \end{aligned} \tag{1}$$

where $\texttt{LSTM}_\theta$ and $\texttt{mlp}_\vartheta$ represent the encoder, with internal state $\vec{h}_t^{(e)}$ given the t-th event; symmetrically, $\texttt{LSTM}_\phi$ and $\texttt{mlp}_\varphi$ represent the decoder, with inner state $\vec{h}_t^{(d)}$. Further, $\texttt{mlp}_\vartheta$ and $\texttt{mlp}_\varphi$ represent multilayer networks parameterized by θ and ϕ, respectively.

Since the main purpose of RAE is to reconstruct the input from a compact representation, the model can be trained by considering a reconstruction loss:

$$\ell(\mathcal{I}, \mathcal{O}) = \frac{1}{n} \sum_{t=1}^{n} \|\vec{x}_t - \vec{y}_t\|_2 \tag{2}$$

Input subsequences are obtained from X through a sliding window mechanism. Each timestep within X is associated with a subsequence $W_t = \{\vec{x}_{t-m+1}, \ldots, \ldots, \vec{x}_t\}$, where m is the window size. Thus, RAE is trained on the set $\{W_m, \ldots, W_N\}$ of subsequences that can

obtained from X, as illustrated in Figure 2, by learning to reconstruct them in a way that minimizes the specified loss.

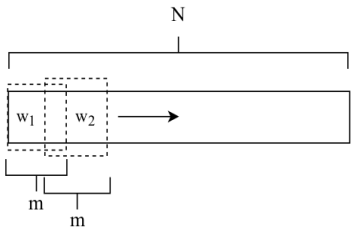


Figure 2: Sliding window mechanism.

RAE will learn to model normal behavior, by mapping input data to compact representations that can capture all the relevant features characterizing it. An anomalous event will compromise this reconstruction capability. Thus, the distance between the input and the output can be exploited as a score to measure the outlierness of the analyzed sequence. Since each event can be included within several overlapping windows, its final score is computed as the average of the outlierness scores of all the involved windows:

$$o(\vec{x}_t) = \frac{1}{m} \sum_{i=1}^{m} \|W_i - \tilde{W}_i\|_2,$$

where $\tilde{W}_i = dec(enc(W_i))$ is the output of the RAE with input W_i.

4 Analysis of the Ethereum Classic Network

The model proposed in the previous section represents a general approach to anomaly detection on sequential data. In this section, we apply our model to the ETC network to identify abnormal situations that can be symptoms of incoming attacks. As discussed in section 2, ETC has experienced two (known) successful attacks: the DAO attack (18 June 2016) and the 51% attack. The latter is scarcely documented and the reports state that it happened in a period within the interval 5-8 January 2019. Hence, our objective is to adopt the RAE model to check whether these events exhibit an anomaly score significantly higher than the other ones.

4.1 Data source and preprocessing

Our experiments have been conducted on a sample of ETC blockchain spanning over a period of four years (July 2015 - July 2019). The dataset is available on Kaggle[2] and consists of seven tables, namely *blocks*, *transactions*, *contracts*, *logs*, *token_transfers*, *tokens* and *traces*. These tables provide information about the usage of the network, the working status and the blocks.

A preliminary analysis of these tables highlighted that only two of them stored relevant information for our purposes, respectively *blocks* and *transactions*.

The raw data have been preprocessed as follows: (i) feature engineering, i.e. selection of the most relevant attribute, correlation analysis, filtering and aggregation; (ii) normalization; (iii) data view generation according to the sliding window mechanism.

[2] https://www.kaggle.com/bigquery/crypto-ethereum-classic

The first two steps result in the following subset of relevant features, computed on a daily basis:

(i) `block_size_average`, the average size (in bytes) of a block; (ii) `provided_gas_average`, referring to the average *provided gas* needed to perform the transaction; (iii) `block_difficulty_average`, the average difficulty (i.e. the effort) necessary to validate a block; (iv) `transaction_average_per_block`, the average number of transactions contained in a block; (v) `gas_used_sum`, the total amount of employed gas; (vi) `transactions_number`, the total number of transactions in all blocks.

The normalization steps is meant to re-scale values to avoid instability effects with the neural modeling, as well as to regularize values by removing seasonal, cyclic and trending fluctuations in the data [6] and it is performed via moving quantile ratio [12]. Figure 2 shows both the original and the normalized data, plotted along the time window. Within the plots, the red lines highlight the periods when the attacks took place.

We partitioned the whole dataset in four periods as shown in Figure 3. Data prior to 7 August 2015 were ignored: they correspond to the first week of the chain genesis and contain several missing values. The time series were finally produced for each block, to feed the RAE according to the learning/evaluation strategy outlined as follows. In practice, we consider two different training set, namely D_1 (relative to the interval August 2015 - April 2016) and D_2 (covering the interval July 2016 - November 2018). T_1 and T_2 (the periods within a range of about two months from the attacks, exact dates are listed in Figure 3) are used as test sets, for evaluating the outlierness scores.

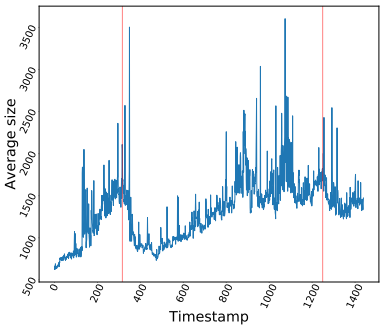
4.2 Experimental Results

We performed three different tests by training two instances of RAE. The first two exploited D_1 as training set and scored all the events within T_1 and T_2, respectively. In the third experiment RAE is trained on $D_1 \cup D_2$ and scores events in T_2.

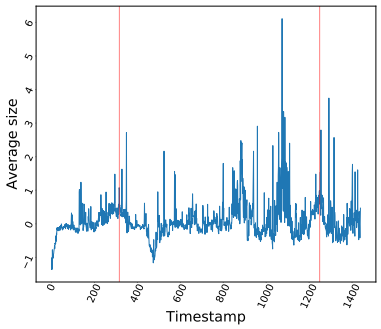
Figures 4a and 4b plot the outlierness scores computed by RAE on both T_1 and T_2. We can see that the DAO attack is perfectly detected by RAE, as shown in Figure 4a. The network provides no warning until day 295 where we can see a small peak. Instead, the outlierness score exhibits a difference of two orders of magnitude on day 316, corresponding to the DAO attack. The result is consistent with the findings of [3], but in our case RAE is capable of perfectly detect the exact day of the attack.

Within Figure 4b, the same model is used to score events in T_2. We can see that the outlierness score changes its pattern as the attack period approaches. However, the peak is translated of a few days. The situation represented here needs some interpretation. The network highlights a strong anomaly on day 1255, a few days after the actual attack. A close inspection of the events occurring during the 51% attack witnesses that a significant number of companies, working on ETC, detected the attack and decided to freeze their activities[3] [4] [5]. Then, on the basis of our analysis, block size and related features seems not fully sufficient to early detect some types of attacks occurring on blockchain, but we figure out that integrating data from other sources (e.g. the server operating system and the application [3]) could improve the detection performances of our approach. As a result, the blockchain didn't register a core amount of transactions which were instead restarted in the forthcoming days. In practice, day 1255 (where the anomaly score is the highest) appears to be the day where all the frozen

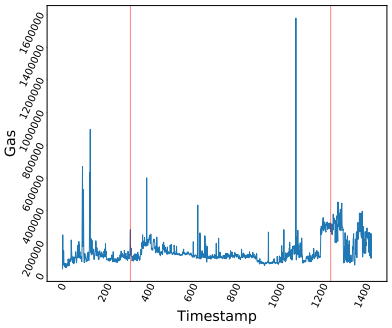
[3] http://shorturl.at/giz14
[4] http://blog.coinbase.com/ethereum-classic-etc-is-currently-being-51-attacked-33be13ce32de
[5] http://cryptonomist.ch/2019/01/07/ethereum-classic-attacco-del-51/



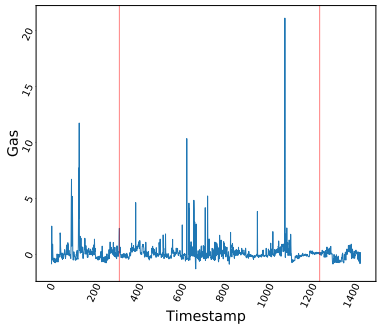
(a) `block_size_average`



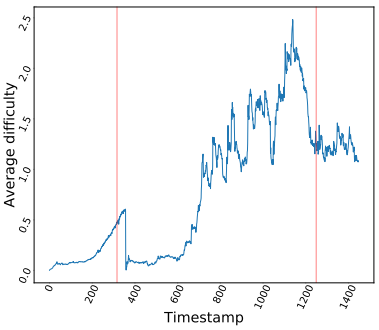
(b) `block_size_average_norm`



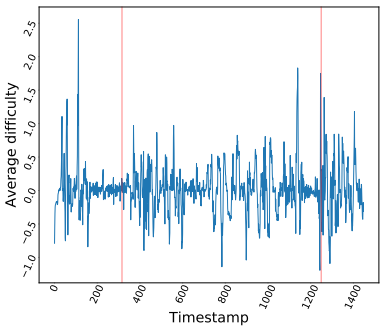
(c) `provided_gas_average`



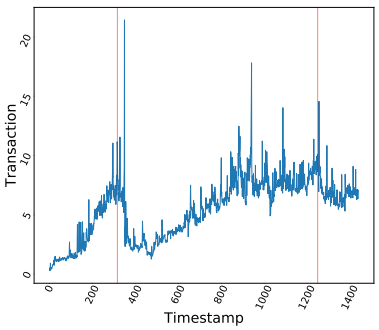
(d) `provided_gas_average_norm`



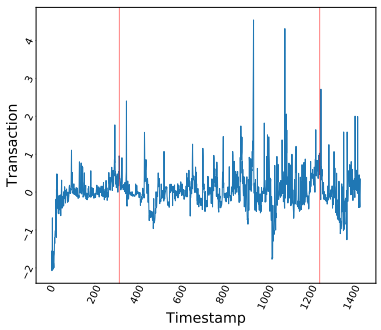
(e) `block_difficulty_average`



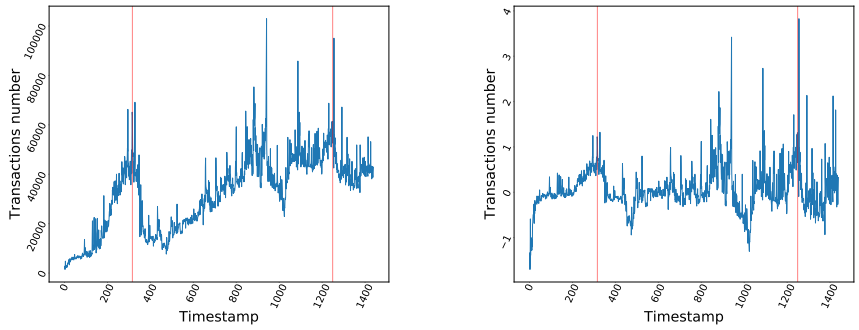
(f) `block_difficulty_average_norm`



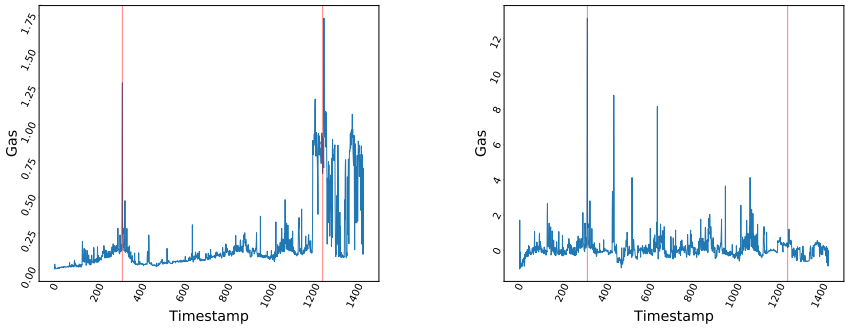
(g) `transaction_average_per_block`



(h) `transaction_average_per_block_norm`



(i) `transactions_number` (j) `transactions_number_norm`



(k) `gas_used_sum` (l) `gas_used_sum_norm`

Figure 2: chronological distribution of the original and normalized data.

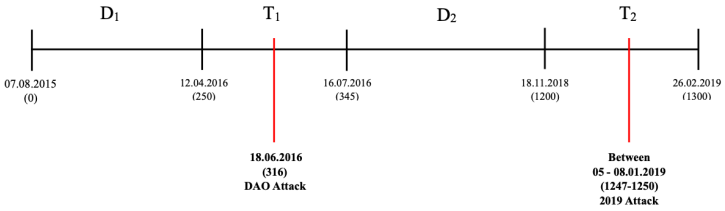
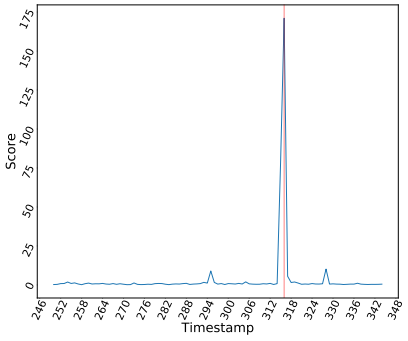
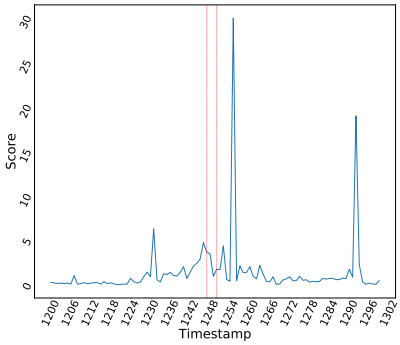


Figure 3: Data split according to the attacks.

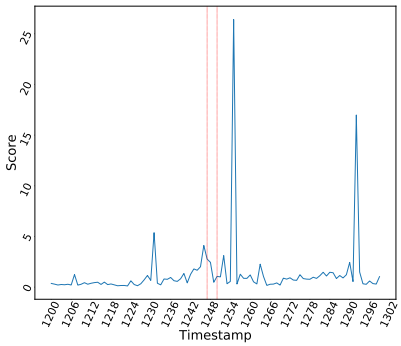
activities were recorded on the network, thus representing the actual moment where the anomaly was injected within the blockchain. This finding is consistent with the findings of the third experiment, where RAE is trained both on D_1 and D_2: Figure 4c shows a similar pattern, thus demonstrating that the preprocessing steps make the RAE findings consistent even on different time periods. The only false positive seems to be represented by the peak on day 1291, which appears in both models. Although there is no reporting of attacks in those periods, it would be interesting to provide a close inspection to the activities during that period and try to explain such anomaly based on those.



(a) Outlierness score of events within T_1.



(b) Outlierness score on T_2.



(c) Outlierness score on T_2. RAE is trained on $D_1 \cup D_2$.

5 Concluding Remarks

In this work, we define an encoder-decoder deep learning model to detect anomalies in the usage of blockchain systems. In this scenario, the main problem is the rarity of the events to identify (i.e. the attacks), therefore, an unsupervised solution is adopted to address this issue. Specifically, the contribution of the paper is twofold: (i) we identified a relevant set of features computed on blockchain logs describing network status in determined time steps; (ii) a sequence-to-sequence neural network model is used to recognize anomalous changes in the blockchain network. Experiments on the complete historical logs of Ethereum Classic prove our model capability to effectively detect security attacks.

As future works, we plan to study the usage of hybrid architectures based on the combination of RNNs with Convolutional Neural Networks [17] to obtain an automatic feature engineering and evaluate eventual improvements. On the other hand, we are interested in defining models able to predict attacks before they happen, improving the network security.

6 Acknowledgment

This work has been partially supported by MIUR - PON Research and Innovation 2014-2020 under project Secure Open Nets.

References

[1] N. Atzei, M. Bartoletti, and T. Cimoli. A survey of attacks on ethereum smart contracts (sok). In *Principles of Security and Trust"*, pages 164–186. Springer, 2017.

[2] Y. Bengio, L. Pascal, P. Dan, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, volume 19, pages 153–160. MIT Press, 2007.

[3] A. Bogner. Seeing is understanding: Anomaly detection in blockchains with visualized features. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, UbiComp '17, pages 5–8. ACM, 2017.

[4] A. L. Buczak and E. Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials*, 18(2):1153–1176, 2016.

[5] V Buterin. Ethereum white paper. a next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper, 2014.

[6] Chris Chatfield. *The analysis of time series: an introduction*. CRC Press, 6th edition, 2004.

[7] H. Chen, M. Pendleton, L. Njilla, and S. Xu. A survey on ethereum systems security: Vulnerabilities, attacks and defenses. *arXiv*, 1908.04507v1, 2019.

[8] K. Cho, Ç. B. v. Merrienboer, Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[9] Nabil El Ioini and Claus Pahl. A review of distributed ledger technologies. In Hervé Panetto, Christophe Debruyne, Henderik A. Proper, Claudio Agostino Ardagna, Dumitru Roman, and Robert Meersman, editors, *On the Move to Meaningful Internet Systems. OTM 2018 Conferences*, pages 277–288. Springer International Publishing, 2018.

[10] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer, 2012.

[11] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[12] David C. Hoaglin, Frederick Mosteller, and John W. Tukey (Editor). *Understanding Robust and Exploratory Data Analysis*. Wiley-Interscience, 1 edition, 2000.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[14] L. Lamport, R. Shostak, and Ma. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[15] S. Mahdavifar and A.A. Ghorbani. Application of deep learning to cybersecurity: A survey. *Neurocomputing*, 347:149 – 176, 2019.

[16] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.

[17] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[18] Meni Rosenfeld. Analysis of hashrate-based double spending. In *arXiv, eprint 1402.2009.*, 2014.

[19] M. Signorini, M. Pontecorvi, W. Kanoun, and Di Pietro R. BAD: blockchain anomaly detection. *CoRR*, abs/1807.03833, 2018.

[20] M. Usman, M.A. Jan, X. He, and J. Chen. A survey on representation learning efforts in cybersecurity domain. *ACM Comput. Surv.*, 52(6):111:1–111:28, 2019.

[21] C. Ye, G. Li, H. Cai, Y. Gu, and A. Fukuda. Analysis of security in blockchain: Case study in 51%-attack detecting. In *2018 5th International Conference on Dependable Systems and Their Applications (DSA)*, pages 15–24, 2018.