Data Mining-based Ethereum Fraud Detection

Eunjin (EJ) Jung
Computer Science Department
University of San Francisco
San Francisco, U.S.A.
Email: ejung@cs.usfca.edu

Marion Le Tilly
Msc Applied Mathematics
École Polytechnique
Palaiseau, France
Email: marion.le-tilly
@polytechnique.edu

Ashish Gehani
Computer Science Laboratory
SRI International
Menlo Park, USA
Email: ashish.gehani@sri.com

Yunjie Ge
Computer Science Department
University of San Francisco
San Francisco, U.S.A.
Email: yge7@usfca.edu

***Abstract*—The popularity of blockchain-based currencies, such as Bitcoin and Ethereum, has grown among enthusiasts since 2009. Relying on the anonymity provided by the blockchain, hustlers have adapted offline scams to this new ecosystem. As a result, Ponzi schemes are proliferating on Ethereum, dressed up as secure investment schemes. They reward early investors with funds from the later ones before collapsing, leaving the last investors empty handed. Illegal in the offline world, they are creating thousands of victims on Ethereum, while stealing millions of dollars worth of ether. We use data mining to provide a detection model for Ponzi schemes on Ethereum, improving over prior work. We built a dataset of likely benign Ethereum smart contracts, in addition to known Ponzi scheme smart contracts, and designed features based on their compiled code and transactions. Using Weka to benchmark several classification algorithms, we obtained models that achieve both high precision and high recall. Our 0-day model can be used as soon as a smart contract is uploaded on the blockchain. The full-feature model continued to show high performance for almost 250 days. A detailed analysis on top-strength features provides novel perspectives on Ponzi scheme behavior.**

***Index Terms*—Blockchain, Smart Contract, Ponzi Schemes, Ethereum**

I. Introduction

In 2009, Satoshi Nakamoto came up with a protocol enabling value transfer among non-trusting individuals: the Bitcoin [1]. A peer-to-peer network enabling financial exchanges between users, based on economic incentives rather than trusted authorities. It operates on the blockchain: a public ledger keeping track of all the transactions in an immutable way using novel ideas such as Nakamoto consensus using *proof of work*. While all transactions are recorded on the blockchain and retrievable by any participant, Bitcoin still ensures the anonymity of its users who identify themselves solely with public keys, signing transactions with associated private keys.

Benefiting from the anonymity, criminals quickly took advantage of its popularity to rip off other users by creating multiple scams. An empirical analysis of Bitcoin-based scams in 2015 showed that at least $11 million was stolen from 13,000 victims [2]. Unlike scams before Bitcoin, cryptocurrencies are not yet subject to any government regulations, making it difficult to seek recompense for damages caused by fraud. Due to the immutability of the blockchain and the anonymity of users, it is nearly impossible to revert a fraudulent transaction.

Most scams are inspired by offline precedents such as Ponzi schemes: a 150-year-old deceit emulating an investment program with high returns, of which one can cash out only if there are enough new users to pay its profit with their investment. As a result, early investors cash in on the backs of the most recent ones, who end up empty-handed. These scams damage the reputation of the whole cryptocurrency ecosystem, including Ethereum. While Bitcoin also has a simple scripting language, Ethereum's Solidity language and the Ethereum Virtual Machine (EVM) provided Turing-completeness for the first time on blockchains. Now users can write *smart contracts*: pieces of code stored on the blockchain that implement applications capitalizing on distributed consensus as well as anonymity, such as escrow services. The combination of the blockchain's immutability and smart contracts opens a new chapter of applications. Now users do not have to place trust in institutions or authorities to behave correctly, but trust the code that is publicly viewable and potentially auditable. A number of projects were released as smart contract auditors and verifiers, such as QuantStamp [3] and CertiK [4]. In fact, "The code is the law" was the motto of the Distributed Autonomous Organization (DAO) [5]. Ethereum's smart contracts opened another level of potential use of blockchain and cryptocurrencies, but unfortunately they also attracted ill-intentioned users.

Ponzi schemes are profitable on Ethereum thanks to the blockchain's popularity. An empirical analysis in 2015 [2] showed that fraudulent transactions already represents 60% of the transaction volume in USD and may increase even more. Successful Ponzis impact few users who make high contributions, creating particularly unfortunate victims. *Smart Ponzi*, the Ponzi scheme smart contract, preys upon the transparency provided by the blockchain as a show of faith to draw in inexperienced customers who cannot identify the the deceit and flaws in the implementation.

Prior work [6], [7], [8] showed that data mining-based Ponzi scheme detection is feasible. There are two types of features: *0-day* features and *behavior-based* features. 0-day features are available as soon as the smart contract is uploaded to the blockchain, such as the bytecode and the size of the smart contract. Behavior-based features are based on the "behavior" of the contract, such as paying the early investors from the payments later investors make. We used a set of smart contracts

with verified source code and present new features. We also used them to build new behavior-based, 0-day and combined classification models with improved performances. Our 0-day model shows 0.98 precision and 0.89 recall, compared to 0.90 and 0.80, respectively, by Chen *et al.* [8]. Our full-feature model shows 1.00 precision and 0.90 recall, compared to 0.94 and 0.81, respectively, in prior work [8]. While these results are strong, active attackers may get hold of the model and its features, and manipulate their smart contracts to evade the detection by the classifiers. We considered a model with selective features that are less controllable and predictable by active attackers and showed results of 0.97 precision and 0.78 recall.

The rest of the paper is organized as follows. Section II summarizes prior work using classification to detect Ponzi scheme smart contracts on Ethereum. Section III provides a brief explanation of the Ethereum platform including smart contracts. Section IV describes how the data sets are collected, how the features are defined, and the classification models are trained. Section VI briefly discusses directions for future work and summarizes the paper.

II. Related Work

Financial scams in cryptocurrencies are different from financial scams outside blockchain platforms in the sense that most transactions relevant to the scams are publicly viewable, with some exceptions such as Zcash [9] and Monero [10]. Kamps and Kleinberg took advantage of this public transaction data in detecting a classic financial fraud, *pump and dump* price manipulation in cryptocurrencies and showed posthumous detection is possible [11].

Nikolic et al. surveyed smart contracts in Ethereum with vulnerabilities inspired by Parity hack [12]. In particular, they found 1,423 suicidal contracts out of 34,200 they collected. We found that some Ponzi scheme smart contracts from the public dataset [13] were suicidal. Nikolic et al. conjectured that the smart contracts with the suicide codes are likely to have vulnerabilities. They also found that out of 1,495 contracts classified as suicidal, there were only 403 distinct contracts, which suggests that the contracts with vulnerabilities are often reused.

Bian et al. analyzed the smart contracts written for Initial Coin Offerings (ICOs) and related information such as white papers to give scores to ICO offers [14]. Our approach only uses the information in the Ethereum blockchain and considers all Ponzi schemes, not just fraudulent ICOs.

Vasek and Moore studied 1,780 Bitcoin-based Ponzi schemes from `http://bitcointalks.org` and showed the social interaction between scammers and victims affect the lifetime of the scams [15].

In [6], Bartoletti et al. analyzed Ponzi scheme smart contracts' behavior on Ethereum, using similarities between contracts bytecode to locate 191 of them. In total, the contracts collected almost half a million USD from more than 2000 distinct users. [6] highlights characteristics of Ponzi schemes behavior such as a high Gini coefficient, a measure of inequality in the distribution of money made by contracts to investors.

The approach in [7] reuses this characteristics to apply data mining techniques to Ponzi schemes implemented on Bitcoin. Their best classifier manages to detect 31 Ponzi schemes out of 32 with 1% of false positives. It detects Ponzi schemes using features computed over a contract's lifetime, narrowing the range of countermeasures that can be implemented against such schemes because tokens can not be refunded nor flagged as stolen. Rahouti et al. [16] surveyed machine learning and data mining-based approaches in anomaly detection in Bitcoin, including [7].

Nevertheless, its approach has inspired [8] which describes a similar approach using Account features but also added Opcode features based on the contract's code stored on the blockchain. They built three classification models, Account, Opcode, and Account+Opcode using XGBoost, and the best performance comes from Account+Opcode model, precision at 94% and recall at 81%.

While Chen et al.'s [8] approach is the most similar to ours, we took three different steps in building our classification models. First, we only used smart contracts with verified source codes on Etherscan [17] that has at least 10 incoming or outgoing transactions as our non-Ponzi data set. The creators of these contracts submit the source code to Etherscan and Etherscan confirms that the source code compiles to the same code as what is uploaded to the blockchain. Many Ethereum users see this as an indicator that it is safe to use these contracts. Etherscan started this service in March 2016, so our non-Ponzi data set only contains contracts posted on Ethereum blockchain after March 2016. This resulted in a smaller non-Ponzi data set than what was used in [8], but this way it is less likely to have wrongly-labeled data in our training data set. Second, we added more Account features, which we call behavior-based features. Third, we used different classification models and achieved better performance. The details of these features and classification models are in Section IV.

III. Ethereum platform

This section elaborates on Ethereum smart contracts and how Ponzi schemes exploit their characteristics. To enable the implementation of blockchain applications, Ethereum [18] integrates an Ethereum Virtual Machine (EVM), a run-time environment for EVM bytecode, to its platform. Thus, creating an Ethereum application boils down to writing its source code in a high-level language such as Solidity[1], compiling it into EVM bytecode and eventually uploading it on the Ethereum blockchain. Once the contract has been uploaded, it is active: it can fired and receive transactions but most importantly, its bytecode is immutable.

Ponzi smart contracts capitalize on this immutability to attract users, claiming that because the contract will run forever without being interfered with, investors are assured to make a profit. Yet, they fail to mention that smart contract creators

[1] http://solidity.readthedocs.io/en/develop/

may modify its parameters such as the fee required for each investment or that, as explained in [6], smart contracts are not proven to be without flaws. In other words, the immutability of the blockchain only ensure that the code execution is automatically enforced.

Ethereum has two types of accounts: contract and externally owned. Each account has a unique 20-bytes address pointing at four fields used to guarantee the uniqueness of each of its state:

- the **nonce**: incremented at each transaction to ensure it is processed only once;
- the account **balance**: current amount of ether of the account;
- the **contract bytecode**: immutable because it is stored on the blockchain;
- the account **storage**.

An externally owned account is controlled by a private key (accordingly by its owner) as opposed to contract accounts controlled by their contract code. Each time a contract account receives a transaction, it activates itself using the payload as input: reading and writing to internal storage, sending transactions or creating contracts in turn. For instance, when a Ponzi smart contract receives a transaction containing a sufficient amount of *Ether*, Ethereum's cryptocurrency, it triggers payment transactions to previous participants. In this context, the transaction received by the smart Ponzi is called a normal transaction as opposed to the one it fires called internal transactions. We use features from both transactions in smart contract classification.

To broadcast transactions to the whole network, they are packaged into blocks which are executed and verified by *miners*. Ethereum miners solve a hash puzzle to create the next block and earn the block reward. They are encouraged to include transactions in the block they create as they can collect *transaction fees* from included transactions. For transactions that need computation, i.e. transactions that invoke smart contracts, fees are proportional to the amount of computation required for the execution of the transaction. This computation fee is calculated as the multiplication of *gas limit* and *gas price*. The higher the price, the more eager miners will be to execute the transaction. If there is not enough gas, the execution fails, it *runs out of gas*: all side effects are reversed but the fee is lost. After the execution, transactions are validated block by block using a consensus protocol similar to Bitcoin, ensuring the safety of the system unless an attacker manages to own more than 51% of the computing power of the whole network.

IV. Classification model

A. Data collection

First of all, we need a training dataset of both Ponzi and non-Ponzi smart contracts in Ethereum. For fraud detection projects, the data collection is often the most laborious step, requiring strenuous manual verification and research to distinguish fraud and non-fraud instances. Thanks to [6], an open-source dataset containing 184 Ponzi scheme addresses is available at [13].[2] We were only able to retrieve the bytecode of 172 of them, as 8 of them executed *self-destruct* code. Each address has been manually inspected to confirm whether it is a Ponzi scheme by both [6], [8], thus we consider these contracts as ground truth. The oldest Ponzi smart contract was posted in Ethereum blockchain on August 7th, 2015 and the latest one was posted on August 27th, 2017.

We used a web scrapper to collect 3,203 addresses of smart contracts with source codes *verified* by Etherscan, involved in at least 10 transactions, while trying to match the time frame of Ponzi scheme. When a developer creates a new smart contract on Ethereum, she has an option of submitting the source code to Etherscan. Then Etherscan verifies the source code to prove the code has deterministic and verifiable builds. Many Ethereum users take this verification as a vote of confidence on the smart contract code. Since Etherscan started to provide this service in March 2016, the oldest non-Ponzi smart contract with verified source code was posted on February 11th, 2016. The latest non-Ponzi smart contract we used was posted on June 7th, 2018, based on the date of the last transaction from the Ponzi smart contracts.

We also ruled out the smart contracts with fewer than 10 transactions so that we will capture the behavior of non-Ponzi smart contracts that have substantial interactions with other accounts. However, it does not prevent the code from having flaws. We were not able to manually confirm all 3,203 smart contracts but performed a spot check and did not encounter any smart contract that was a Ponzi scheme.

We also collected all the transactions interacted with these contract addresses using Etherscan APIs. These transactions contain information such as which account paid how much Ether to the smart contract in question when. This information is later used to monitor the behavior of smart contract and turn them into features for classification. Then through Web3 python interface [3], the EVM bytecode was recovered from the blockchain.

B. Classification models

In order to come up with a classification model as effective as possible, several well-known classification algorithms competed representing different learning strategies:

- **J48**: a decision tree which enables visualization of the decision process exhibiting which features are used and how;
- **Random Forest** (RF): a decision forest, it aggregates the prediction of individual trees. It has proven to be an excellent classifier for other fraud detection [19];
- **Stochastic Gradient Descent** (SGD): iterative method for optimizing a differentiable objective function.

Since we aim to detect Ponzi schemes, models will be evaluated using three metrics below:

[2]The original data set used in [6] contained 152 Ponzi contracts, and the authors added more to the dataset in January 2019.

[3]github repository: https://github.com/ethereum/web3.py

- **precision**: the ratio of actual Ponzi instances to those classified as Ponzi.

$$precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

- **recall**: the ratio of correctly classified Ponzi instances to all Ponzi smart contracts.

$$recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

- **F-score**

$$F - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Note that we focus on the precision and recall of Ponzi instances. The training dataset is highly unbalanced with 3,203 non-Ponzi instances and only 172 Ponzi instances. As a result, the $ZeroR$ algorithm, which simply classifies every instance as non-Ponzi, correctly classifies 94.9% of the all instances. However the precision of ZeroR algorithm would be impossible to compute (divided by 0) and its recall would be 0. Since our goal is to protect Ethereum users from Ponzi schemes, we optimize for precision and recall of Ponzi instances.

C. Features and their robustness

Feature selection is a crucial upstream work to have an efficient classification model. The following approach is motivated by previous works [6], [8], [20], [21] which shed light on two types of features. Both [6], [8] granted a significant share of their work to the contracts' bytecode to create features or to find similar contracts, while [8], [7], [20], [21] characterized contracts by their interactions with their environment. We call the former as code features and the latter behavior-based features. Since the code features are available as soon as the contracts are uploaded to the blockchain, we use code features to build our 0-day classification model that is usable as early as day 0, when the contract is uploaded.

1) Code (0-day) features: Using `evmdis`[4], an open source disassembler, each contract bytecode has been turned into its equivalent in opcodes. Opcode is the abbreviation of operation code in Ethereum Virtual Machine. Figure 1 is a portion of disassembled bytecode from a smart Ponzi.

Then, the frequency of each opcode in the contract code is computed and stored in the database. We merged the frequencies of some opcodes in the same category into one, such as DUP1 and DUP2 into one DUP12 feature. The size of the bytecode of each contract in bytes is stored too to create the **size_info** feature.

Top 10 code features Here we explain the top-10 strong code features in our classification models. The full list of the EVM opcodes is available at [22].

- SSTORE writes a (u)int256 value to storage according to its key.
- POP pops a (u)int256 off the stack and discards it.
- MSTORE writes a (u)int256 to memory at its offset.
- SWAP1 swaps the last two values on the stack.

[4]github repository: https://github.com/Arachnid/evmdis

```
# Stack: []
0x4     MSTORE(0x40, 0x60)
0xA     JUMPI(:label0, !CALLDATASIZE())

# Stack: []
0x13    PUSH(CALLDATALOAD(0x0) / 0x2 ** 0xE0)
0x19    DUP1
0x1E    JUMPI(:label1, POP(@0x13) == 0x72EA4B8C)

:label0
# Stack: []
0x20    PUSH(:label2)
0x23    PUSH(CALLVALUE())
0x24    PUSH(0x0)
0x26    DUP1
0x27    DUP1
0x31    DUP4
0x36    JUMPI(:label6, POP(@0x23) < 0xDE0B6B3A7640000)

# Stack: [0x0 0x0 0x0 @0x23 :label2]
0x37    PUSH(0x1)
0x39    DUP1
0x3B    DUP1
0x3C    PUSH(POP(0x1) + SLOAD(POP(0x1)))
0x3D    SWAP1
0x3E    DUP2
0x3F    SWAP1
0x40    SSTORE(POP(0x1), POP(@0x3C))
0x41    PUSH(0x3)
0x43    DUP1
0x44    PUSH(SLOAD(POP(0x3)))
0x45    PUSH(CALLER())
0x46    SWAP2
0x47    SWAP1
0x48    DUP2
0x4E    JUMPI(0x2, !(POP(@0x3C) < POP(@0x44)))
```

Fig. 1: Disassembled bytecode for contract 0x109c4f2ccc82c4d77bde15f306707320294aea3f

TABLE I: Statistics of Top Code Feature Frequency

Opcode	Ponzi		non-Ponzi	
	avg	stdev	avg	stdev
SSTORE	15.25	12.05	0.36	3.50
POP	57.35	61.67	3.42	29.20
MSTORE	32.50	48.74	2.33	18.16
SWAP1	95.88	84.86	4.24	34.76
STOP	1.08	0.70	0.03	0.38
DUP9	4.03	3.46	0.10	0.95
RETURN	2.92	1.37	0.13	1.02
SWAP2	35.86	35.88	1.65	13.63
DUP12	135.39	129.22	6.64	52.47
JUMP	19.45	15.00	1.41	11.41

- STOP halts execution of the contract
- DUP9 duplicates the 10th value in the stack.
- RETURN returns from this contract call
- SWAP2 swaps the top of the stack with the 3rd last element
- DUP12 DUP1 clones the last value on the stack and DUP2 clones the 2nd last value on the stack
- JUMP unconditional jump

Table I shows the average and standard deviation of their frequencies in Ponzi and non-Ponzi smart contracts.

Our top-10 code features are very different from top-10 important features in [8]: with the exception of MSTORE, which is ranked at 10th important feature in [8], our top-10 do not overlap with their top-10 important features. As our non-Ponzi data set only consists of contracts with verified source code with minimum 10 transactions from the same timeframe as Ponzi data set, our non-Ponzi data set is much smaller than theirs and the Opcode distributions are very different, which

results in different features in top 10.

2) Behavior-based features: Behavior-based features use the interactions of the smart Ponzi with its users, i.e. transactions. These features capture logical and relevant ideas reflecting how Ponzi schemes behave.

Previously used features

The literature [8], [7], [20], [21] provides a range of behavior-based features which characterize Ponzi smart contracts behavior.

- **nbr_tx_in** (resp. **nbr_tx_out**): the number of transactions which transferred ether to (resp. from) the smart contract;
- **Tot_in** (resp. **Tot_out**): the total amount of Ether transferred to (resp. from) the smart contract;
- **mean_in** (resp. **mean_out**): the average value of Ether transferred to (resp. from) the smart contract;
- **sdev_in** (resp. **sdev_out**): the standard deviation of Ether transferred to (resp. from) the smart contract;

Strength and robustness: We found that most of these features were not very strong, with the exception of the `nbr_tx_in`. This observation concurs with the findings in [8]. We speculate that an experienced Ponzi smart contract creators use several Ethereum wallet addresses to mask the number and amount of transactions in and out of the Ponzi smart contract to camouflage its fraudulent nature.

New behavior-based features

Ponzi schemes are considered as fraud mainly because the sooner a user invests, the bigger the reward, to the extent that some investors never see their money back. Early investors receive huge amount of money while later investors receive very limited ones. This distribution of money is unequal and therefore characterized by a high Gini coefficient.

The Gini coefficient, ranging from 0 to 1, characterizes inequality among a given distribution: the closer it is to 1 the more unequal the distribution is. For a given distribution a among n participant, it is computed through the following formula:

$$G(a) = \frac{\sum_{i=1}^{n} (2i - n - 1)a(i)}{n \sum_{i=1}^{n} a(i)}$$

Furthermore, Ponzi smart contracts tend to slow down in the transaction frequency as shown in [6], hence the *Gini_time* features.

- **avg_time_btw_tx**: average amount of time between two transactions;
- **lifetime**: time between the first and the last transactions;
- **Gini_amt_in** (resp. **Gini_amt_out**): the Gini coefficient computed over the Ether amount of transactions to (resp. from) the smart contract;
- **overlap_addr**: the number of addresses that paid to the contract and also were paid by it.
- **Gini_time_in** (resp. **Gini_time_out**): the Gini coefficient computed over the time of the transactions paid (resp. was paid by) the smart contracts;
- **num_addr_in** (resp. **num_addr_out**): the number of addresses (accounts) that paid to (resp. were paid by) this contract;

TABLE II: Statistics of Top 8 Behavior-based Feature Frequency

Feature	Ponzi		non-Ponzi	
	avg	stdev	avg	stdev
size_info (bytes)	3848.47	3619.51	12514.90	7560.74
nbr_tx_in*	107.60	186.42	9594.61	40253.79
lifetime (sec)	23M	31M	21M	15M
num_addr_in	107.60	186.42	8349.56	38549.43
gini_amt_in	0.53	0.34	0.60	0.43
overlap_addr	6.98	16.39	13.83	220.48
gini_time_out	0.0003	0.0013	0.0002	0.0005
avg_time_btw_tx	526079.02	1612850.48	79009.68	153799.81

TABLE III: 0-day model performance comparison

Model	Precision	Recall	F-score
J48	0.97	0.93	0.95
Random Forest	0.96	**0.96**	**0.96**
SGD	**0.98**	0.94	**0.96**
[8]	0.90	0.80	0.84

Table II shows the statistics of top 8 behavior-based features frequency in Ponzi and non-Ponzi smart contracts. Except for *nbr_tx_in**, the rest of the top behavior-based features are new features.

V. Performance evaluation

This section presents our behavior-based, 0-day, and full-feature model's performances compared to corresponding models in [8]. We also show that our full-feature model maintains high precision and recall from day 0 to day 248 of a smart contract's lifetime. Each model was trained and tested using 10-fold cross-validation. Metrics may vary with the random seed used for the cross-validation. Therefore, averages of precision and recall were computed. All performance evaluations were done with Weka 3.8.3 [23].

A. 0-day model for early detection

Our 0-day model only uses features that are available on 0^{th} day – i.e. as soon as the smart contract is uploaded to the blockchain. Table III shows how accuracy and recall of all three classification models compares to the opcode-only model in prior work [8]. The best performance numbers are shown in bold. The strong performance of our 0-day model continues not only on day 0 but also on the following days as all the features rely on immutable properties of the smart contract. In other words, the 0-day model is effective not only for early detection but also Ponzi-scheme detection during the entire lifetime of a smart contract.

Table I above shows the top-10 strong features in 0-day model using information gain[5].

B. Behavior-based model

Table IV shows our behavior-based models' performance compared to the behavior-based model in prior work [8]. J48 shows the best recall of 0.872 among three classifier models. Random Forest shows precision greater than Chen *et al.* [8].

[5]https://en.wikipedia.org/wiki/Information_gain_ratio

TABLE IV: Behavior-based model performance comparison

Model	Precision	Recall	F-score
J48	0.93	**0.87**	**0.90**
Random Forest	**0.98**	0.84	**0.91**
SGD	1.00	0.08	0.15
[8]	0.74	0.32	0.44

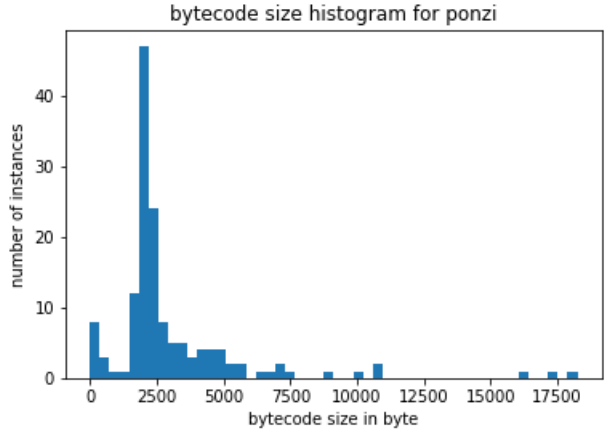
While SGD resulted in better precision, the recall renders this model unusable, as shown by its F-score. Overall, the model using only behavior-based features did not perform as well as our own 0-day model but our new behavior-based features improved the classification effectiveness.

Table II above shows top-8 behavior-based features according to their information gain. Note that `size_info` is the only feature included in both behavior-based and 0-day models, as this feature is available as soon as the contract is uploaded but also reflects the way the contract is written. The statistics in Table II provide insight into why these features help a model discriminate. We discuss two of our new features and their strengths in detail.

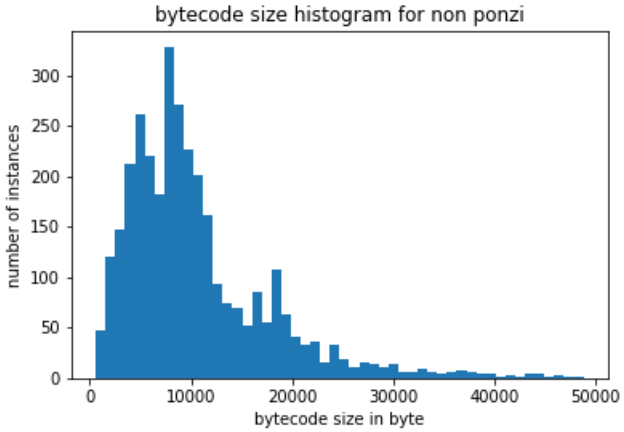
1) size_info: Figure 2 illustrates the Ponzi smart contracts with a standard deviation less than half of the standard deviation of the non-Ponzi smart contracts. Our conjecture is that Ponzi smart contracts are often reused by different attackers as any smart contract can be downloaded from the blockchain and re-uploaded by a different user, which results in the significantly smaller standard deviation. If the classification model is known to the attackers, they may add unreachable or no-op codes to their smart contracts to change the smart contract sizes. However, such change is only feasible if the attackers have the source code to begin with. So the attackers who are copying and re-uploading existing Ponzi smart contracts are less likely to be able to defeat the classification model.

2) lifetime: Figure 3 shows lifetime histograms for both Ponzi and non-Ponzi instances. Table II also shows that the standard deviation of non-Ponzi smart contracts is less than half of that of Ponzi ones. Lifetime is relatively easy to manipulate for attackers as they can pay the smart contract or get paid by the smart contract at a later date. However, all transactions require a transaction fee to be added to the canonical chain. The attackers would have to pay for lifetime change. This may discourage them from issuing transactions and changing the lifetime too often.

3) avg_time_btw_tx: Table II shows a rather different result from the intense activity of the Ponzi schemes described in earlier work [6], showing much longer average elapsed time between transactions associated with Ponzi smart contracts than non-Ponzi – i.e., 526K versus 79K. This is due to some Ponzi schemes firing transactions a year after their last sign of activity. The first quartile of smart Ponzi is smaller than 6 minutes compared to 70 minutes for non-Ponzi, an observation that supports prior analysis [6]. Even when the classification model is known to the attackers, it will be difficult for an attacker to manipulate this feature as the attacker with many addresses can fire transactions to shorten the elapsed time between transactions, but cannot prevent transactions from happening and elongate the time between transactions.



(a) Ponzi smart contracts



(b) Non-Ponzi smart contracts

Fig. 2: Bytecode size histograms

TABLE V: Full-feature model performance comparison

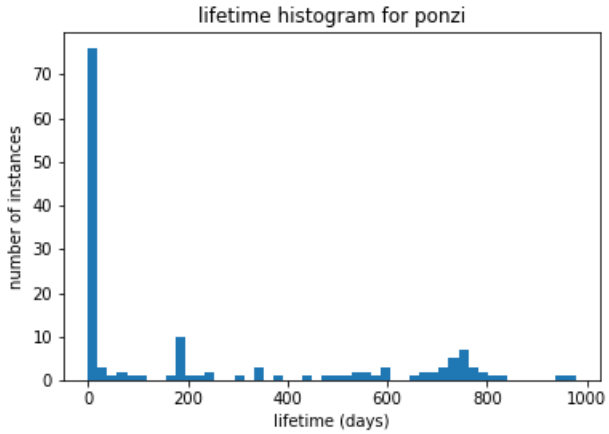
Model	Precision	Recall	F-score
J48	0.98	**0.97**	**0.97**
Random Forest	0.93	0.92	0.93
SGD	**0.99**	0.94	0.96
[8]	0.94	0.81	0.86

C. Full-feature model

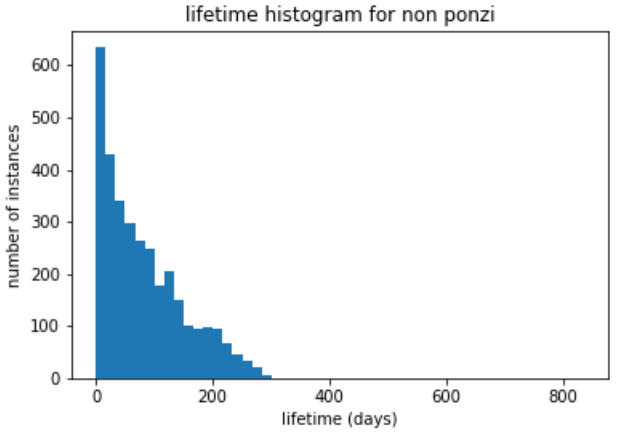
Table V shows our full-feature model's performance, combining both code and behavior-based features. The best performance numbers are shown in bold.

We tested the full-feature model on days 0 to 248 from when the smart contract had been posted on the blockchain. Both precision and recall (and thus F-score as well) continued to show similar performance of over 0.98 precision and 0.94 recall. The performance over the first 40 days from when the smart contract was uploaded is shown in Figure 4.

Overall, the full-feature model shows little improvement over the best performance of 0-day models. In fact, the



(a) Ponzi smart contracts



(b) Non-Ponzi smart contracts

Fig. 3: Lifetime histograms

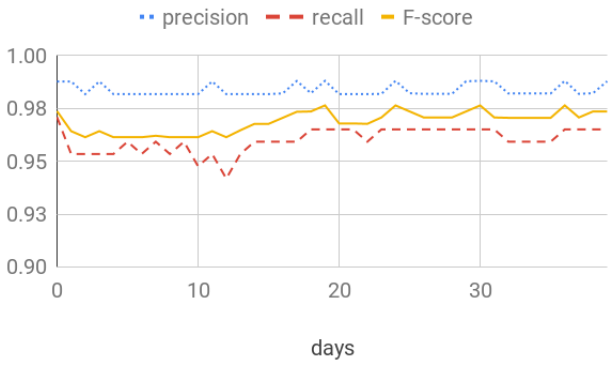


Fig. 4: Full-feature model's perf. over the first 40 days

TABLE VI: Rankings of top 20-30 features on Days 0 and 50

day 0	day 50
size_info	size_info
SWAP7	SWAP7
Gini_amt_in	overlap_addr
SWAP5	Gini_amt_in
overlap_addr	SWAP5
lifetime	mean_in
SWAP6	SWAP6
nbr_tx_in	gini_time_out
mean_in	DUP7
gini_time_out	sdev_in
DUP7	Tot_in

size_info feature is only listed at the 20^{th} position in information gain ranking, while the top 19 features are all code features. Table VI shows how the top 20-30 features' ranking changed from day 0 to day 50.

Interestingly, *avg_time_btw_tx* is not listed as a top behavior-feature in the full-feature model. Instead, the average amount of investment (*mean_in*) is useful on the first day. Our conjecture is that non-Ponzi smart contracts are more likely to have investments starting on day 0 than Ponzi smart contracts. While this feature remains strong on day 50, *lifetime* becomes less important on day 50. Note that our *lifetime* feature is in seconds, and may differ greatly even in one day. By day 50, the discrimination power of *lifetime* is not as strong as *sdev_in* or *Tot_in*, which provide more information on investment patterns.

VI. Conclusion

While the blockchain is considered the next digital revolution, it is also perceived as an opportunity for exploitation by scammers. Ponzi smart contracts are presented as high-return investment opportunities, while the risks involved are barely mentioned. We built data mining-based Ponzi detection models. The 0-day model can be used to flag Ponzi smart contracts as soon as they are uploaded to the blockchain. It has a precision of 0.98 and recall of 0.96, improved from 0.90 and 0.80 in prior work [8]. Our behavior-based model has precision of 0.98 and 0.87, greatly improved from the earlier work's 0.74 and 0.32, respectively. We also discussed behavior-based features in the case of active attackers who manipulate the smart contracts to evade the classifier-based detection.

Our full-feature model exhibits a precision of 0.99 and recall of 0.97, improving over the prior work's 0.94 and 0.81. In practice, this model can be used to flag suspicious contracts so potential investors know about their risk. By testing on the thousands of contracts already on the blockchain and doing so over almost 250 days, we validated the usefulness of the models. Both our 0-day model and the full-feature model maintain high precision and recall (over 0.90) from the first day the smart contract is uploaded (day 0) to day 248. Data mining-based Ponzi detection can server as the first line of defense against fraudulent smart contracts.

ACKNOWLEDGEMENTS

We thank Bartoletti *et al.* for sharing the Ponzi data set and Howon Song for the Etherscan scraper used to extract (non-Ponzi) smart contract addresses.

This work was funded in part by the National Science Foundation (NSF) under Grant ACI-1547467 and Department of Homeland Security (DHS) Science and Technology (S&T) Directorate under Contracts HSHQDC-16-C-00034 and HSHQDC-16-C-00024. The views and conclusions contained herein are the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, DHS or the US government.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf," 2009.
[2] M. Vasek and T. Moore, "There's no free lunch, even using bitcoin: Tracking the popularity and profits of virtual currency scams," in *Financial Cryptography and Data Security*, 2015.
[3] "Quantstamp," https://quantstamp.com/, 2017.
[4] "Certik," https://certik.org/, 2017.
[5] "Distributed autonomous organization (dao)," https://en.wikipedia.org/wiki/The_DAO_(organization), 2016.
[6] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia, "Dissecting ponzi schemes on ethereum: identification, analysis, and impact," http://arxiv.org/abs/1703.03779, 2017.
[7] M. Bartoletti, B. Pes, and S. Serusi, "Data mining for detecting bitcoin ponzi schemes," http://arxiv.org/abs/1803.00646, 2018.
[8] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng, and Y. Zhou, "Detecting ponzi schemes on ethereum: Towards healthier blockchain technology," in *Proceedings of the WWW '18*.
[9] "Zcash," https://z.cash/, 2018.
[10] "Monero," https://ww.getmonero.org/, 2014.
[11] J. Kamps and B. Kleinberg, "To the moon: defining and detecting cryptocurrency pump-and-dumps," *Crime Science*, vol. 7, no. 1, p. 18, Nov 2018.
[12] I. Nikolić, A. Kolluri, I. Sergey, P. Saxena, and A. Hobor, "Finding the greedy, prodigal, and suicidal contracts at scale," in *Proceedings of ACSAC '18*.
[13] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia, "Dataset of ponzi schemes," https://goo.gl/CvdxBp, 2017.
[14] S. Bian, Z. Deng, F. Li, W. Monroe, P. Shi, Z. Sun, W. Wu, S. Wang, W. Y. Wang, A. Yuan, T. Zhang, and J. Li, "Icorating: A deep-learning system for scam ICO identification," http://arxiv.org/abs/1803.03670, 2018.
[15] M. Vasek and T. Moore, "Analyzing the bitcoin ponzi scheme ecosystem," in *The 5th Workshop on Bitcoin and Blockchain Research*, 2018.
[16] M. Rahouti, K. Xiong, and N. Ghani, "Bitcoin concepts, threats, and machine-learning security solutions," *IEEE Access*, vol. 6, pp. 67 189–67 205, 2018.
[17] "Etherscan: the ethereum block explorer," https://etherscan.io/, 2017.
[18] V. Buterin, "A next generation smart contract and decentralized application platform," *Ethereum white paper*, 2014.
[19] S. Bhattacharyya, S. Jha, K. K. Tharakunnel, and J. C. Westland, "Data mining for credit card fraud: A comparative study," *Decision Support Systems*, vol. 50, pp. 602–613, 2011.
[20] D. Zambre and A. Shah, "Analysis of bitcoin network dataset for fraud," http://snap.stanford.edu/class/cs224w-2013/projects2013/cs224w-030-final.pdf, 2013.
[21] T. Pham and S. Lee, "Anomaly detection in bitcoin network using unsupervised learning methods," http://arxiv.org/abs/1611.03941, 2016.
[22] "Ethereum virtual machine opcodes," https://ethervm.io/#opcodes, 2018.
[23] E. Frank, M. A. Hall, and I. H. Witten, *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*, 4th ed. Morgan Kaufmann, 2016.