*Proc. Blockchain Kaigi 2023 (BCK23)*
JPS Conf. Proc. **43**, 011002 (2024)
https://doi.org/10.7566/JPSCP.43.011002

# Machine Learning Approach to Identify Malicious Smart Contract Opcodes: A Preliminary Study

Derek LIU[1*], Francesco PICCOLI[2*], and Victor FANG[3*]

[1] *derek.liu@anchain.ai*
[2] *francesco.piccoli@anchain.ai*
[3] *victor.fang@anchain.ai*
** Anchain.ai, San Jose, California, U.S.A*

(Received February 15, 2024)

The fast growth of the Blockchain and Web3 ecosystem in the past few years has been met with a surge in scams, hacks, ransomware, and rug pulls, leading to a 2.3 Billion dollars financial losses. This paper presents a preliminary and comprehensive analysis aimed at identifying and flagging potential malicious smart contracts deployed on the Ethereum blockchain, and potentially all Ethereum Virtual Machine (EVM) compatible chains. We focus on smart contract opcodes, 100% universally available and executable on any Ethereum Virtual Machine stacks. This research highlights the potential for the use of fundamental data science techniques to automatically detect and limit the reach of malicious smart contracts. We demonstrate that our machine learning can achieve 85.17% of accuracy, 86.36% precision, and 80.00% recall, which manifests the feasibility towards building an AI system to automatically safeguard the Web3 blockchain system.

## 1. Introduction

Towards a more secure smart contract blockchain ecosystem, this paper aims to take an initial assessment of the feasibility and reliability of building a predictive, real time machine learning model to identify threats in the Web3 decentralized blockchain ecosystem. Using smart contract Opcodes[1], and contextual metadata, collected from a diverse set of sources, we compare various traditional supervised machine learning techniques in a holistic evaluation.

### *1.1 Background*

The threat landscape in the cryptocurrency realm has proven to be highly versatile and the ability for bad actors to target users in the space continues to test the boundaries of trust and security. Smart contracts are self-executing agreements written in code on blockchain networks like Ethereum. They automate contract execution, eliminating the need for intermediaries, ensuring transparency, and enabling decentralized applications (dApps) in Web3. They play a vital role in decentralized finance (DeFi), decentralized

autonomous organizations (DAOs), and other Web3 innovations by enabling trustless interactions and automated processes.

According to the 2023 Annual Risk Report (AnChain.ai)[2], in 2023 alone, nearly 500 million smart contracts were deployed across EVM chains (Celo, Avalanche, Arbitrum, Fanton, BSC, Ethereum, Optimism), bringing the total to nearly $2.3 billion. A notable shift in the crypto landscape stems from the movement of bad actors from the historically dominant Bitcoin blockchain to smart contract enabled chains like Ethereum, introducing novel threats. The most significant smart contract attack of 2023 occurred on March 13, in which a single attacker embezzled approximately $197 million from Euler Finance, a prominent DeFi lending/borrowing dApp on Ethereum, in about 15 minutes.

### *1.2 Motivation*

As today's dominant smart contract platform, the Ethereum blockchain ecosystem needs tools for securing smart contracts. Cryptocurrency fraud and scams, ranging from investment fraud to phishing attacks, emphasize the risks faced by users. The popularity of deceptive smart contracts, demonstrated by wallet drainers and spoof tokens, poses a significant threat.

Ethereum wallet drainers are malicious smart contracts or applications designed to exploit vulnerabilities in Ethereum wallets or DeFi protocols, often resulting in the theft of funds from unsuspecting users. They can employ various tactics such as phishing attacks, rug pulls, or exploiting vulnerabilities in smart contracts to drain funds from wallets.[3]

Spoof tokens are fraudulent tokens created to deceive investors by mimicking the branding or appearance of legitimate cryptocurrencies or tokens. They are often deployed on decentralized exchanges (DEXs) and promote false promises or exaggerated returns to attract investors. However, once investors purchase these spoof tokens, their value may plummet, and the creators may disappear with the funds, leading to financial losses for investors.

If widespread adoption of smart contracts is desired, it is crucial to develop security models capable of swiftly identifying bad actors. An example of one fundamental model would flag potential malware and ransomware smart contracts before an individual unknowingly tries to interact with one.

### *1.3 Objectives*

The primary objective of this project is to leverage fundamental data science techniques to conduct a comprehensive analysis of smart contracts deployed on the Ethereum blockchain.

The analysis in this project aims to compare a variety of different algorithms to discern between malicious and safe smart contracts. Each model will be evaluated against a set of metrics with an eventual goal of building a proactive defense mechanism that can preemptively identify and mitigate potential threats. This study includes novel machine learning methods and training datasets that have not been used in other publications with similar objectives discussed in literature review.

Looking forward, the selected model holds the potential for practical implementation in the form of user-friendly tools. One envisioned avenue is the development of an add-on or browser plugin, akin to existing antivirus tools, providing users with an additional layer of security against malicious smart contracts.

## 2. Literature Review

Smart contracts, integral components of blockchain systems, have garnered significant attention due to their potential to automate peer-to-peer transactions and decentralized services. However, this innovation also brings forth substantial challenges, particularly in terms of security vulnerabilities and fraudulent activities. This literature review aims to provide an overview of existing research efforts in detecting and mitigating security risks associated with Ethereum smart contracts.

### *2.1 Scam Detection Frameworks*

Hu et al. (2022)[4] proposed SCSGuard, a deep learning-based scam detection framework that utilizes bytecode features extracted from smart contracts. By employing a GRU network with attention mechanism, SCSGuard demonstrates high accuracy, precision, and recall in identifying various scam genres, including Ponzi and Honeypot schemes.

Huang et al. (2021)[5] introduced a method for hunting vulnerable smart contracts by leveraging graph embedding techniques. Through bytecode normalization and slicing,

coupled with unsupervised graph embedding, the proposed approach effectively identifies potential vulnerabilities, showcasing efficiency and high precision.

*2.2 Static Analysis Frameworks*

Feist et al. (2019)[6] presented Slither, a static analysis framework designed to detect vulnerabilities and optimize Ethereum smart contracts. By converting Solidity code into an intermediate representation, SlithIR, and applying program analysis techniques such as dataflow and taint tracking, Slither offers automated vulnerability detection and code optimization opportunities.

Brent (2018)[7] introduced Vandal, another security analysis framework for Ethereum smart contracts. Vandal employs an analysis pipeline that converts low-level EVM bytecode into semantic logic relations, enabling declarative expression of security analyses. The framework demonstrates high effectiveness and efficiency in detecting vulnerabilities across a large number of contracts.

*2.3 Defect Detection Methods*

Hu et al. (2023)[8] proposed SoliDetector, a defect detection tool based on a knowledge graph of the Solidity language. By defining ontology layers and constructing instance layers capturing syntactic and logical relationships, SoliDetector achieves comprehensive defect localization with high accuracy and speed.

Chen et al. (2021)[9] introduced DefectChecker, a symbolic execution-based approach for detecting contract defects in Ethereum smart contracts. Focusing on defects that can cause unwanted behaviors, DefectChecker analyzes bytecode to identify vulnerabilities, demonstrating high F-score and efficiency in evaluation.

*2.4 Systematic Reviews and Comparative Studies*

Kushwaha et al. (2022)[10] conducted a systematic review of security vulnerabilities in Ethereum smart contracts. The study provides insights into various vulnerabilities, detection tools, real-life attacks, and preventive mechanisms, offering a comprehensive overview of the security landscape in Ethereum-based systems.

Rameder et al. (2022)[11] performed a literature review focusing on automated vulnerability analysis of smart contracts on Ethereum. The review assesses classifications of vulnerabilities, detection methods, security analysis tools, and

benchmarks, providing a structured overview of the state of the art in Ethereum smart contract security analysis.

The literature review highlights diverse approaches and tools developed to address security challenges in Ethereum smart contracts, ranging from deep learning-based scam detection to static analysis frameworks and defect detection methods. The research in this paper intends to build upon the previous work by leveraging their insights in a real time detection framework that would be more feasible to develop for mass adoption.

## 3. Methodology

The research conducted required the following steps to achieve a predictive model:

- Data collection
- Data preprocessing
- Model selection

### *3.1 Data Collection*

The data collection effort involved gathering a total sample of 835 smart contracts from the Ethereum blockchain. Among these, 455 contracts were categorized as safe, while 380 contracts were identified as malicious. Within the malicious category, 327 contracts were flagged as scams, and 53 contracts were recognized as vulnerable.

#### *3.1.1 Malicious Smart Contracts*

To build a smart contract detection model on the Ethereum blockchain, research first needs to define the criteria for malicious contracts, categorizing them into Hacker/Scammer and Vulnerable types. Collection methods, detailed in subsequent subsections, utilize sources such as Etherscan, forums, blogs, and Twitter for a comprehensive dataset. Challenges include verification and data integrity. Overall, within the malicious category, 327 contracts were flagged as scams, and 53 contracts were labeled as vulnerable.

To establish a reliable ground truth for identifying malicious smart contracts, the experiment defines a contract as "malicious" if there is sufficient evidence supporting its classification into either of the following categories:

Hacker/Scammer:

*Definition*: Contracts that falsely impersonate a different entity.

*Example*: Figure 1 shows an address on etherscan with a phishing warning. Most likely this contract was created by a bad actor to deceive careless or unassuming individuals to send funds to the wrong destination.

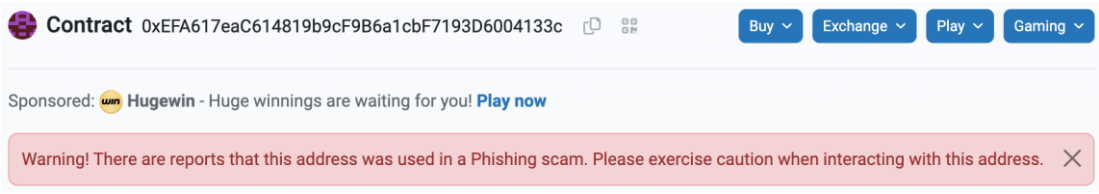


Figure 1: Address 0xEFA617eaC614819b9cF9B6a1cbF7193D6004133c with a phishing warning on Etherscan.

Vulnerable Contract:

*Definition*: Contracts known to have vulnerabilities that could potentially lead to exploitation.

*Example*: Figure 2 shows an example of a vulnerable contract on etherscan due to a MultiSig exploit vulnerability (Breidenbach)[12] which would allow attackers to drain victims' funds.

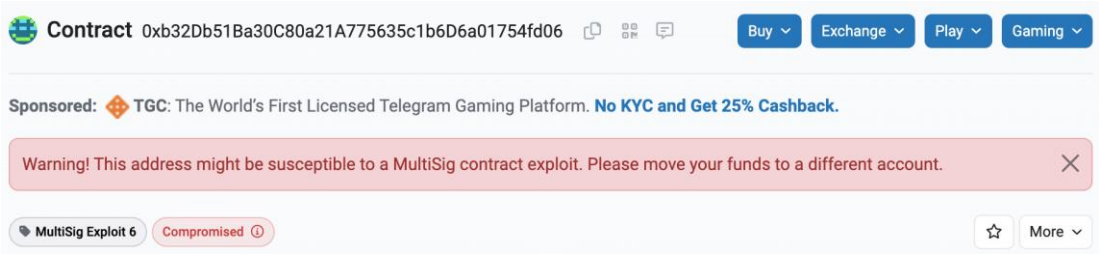


Figure 2: Address 0xb32Db51Ba30C80a21A775635c1b6D6a01754fd06 with MultiSig exploit vulnerability warning on Etherscan.

Malicious smart contract addresses were collected from multiple data sources with the goal of producing a comprehensive dataset given the time constraint of the project.

The following sources were utilized in the collection of known malicious smart contracts in order of volume gathered:

- Etherscan[13] and Etherscan Label Cloud[14]:

1. Etherscan is a popular Ethereum blockchain explorer. It was used in this research to identify and categorize smart contracts based on user-provided labels.
2. Etherscan Label Cloud is a functionality within Etherscan specifically designed for finding other addresses sharing the same labels.

- Web Forums[15]: Forums such as Chainabuse and Web3guard compile individual reports of malicious smart contracts and serve as a good source of community-driven insights.
- Twitter: Tweets documenting historical incidents involving malicious smart contracts

The rationale for malicious contract data collection strategy is to incorporate diverse sources, combining public databases, social media, and historical insights from blogs and forums to comprehensively identify historical threats in the Ethereum blockchain.

The combination of sources ensures a diverse dataset, capturing various perspectives and sources of information. Using Twitter as a data source allows for the incorporation of real-time updates, crucial in an ever-evolving cybersecurity landscape. Blog posts and forums provide historical context, aiding in the identification of trends and recurrent issues. Lastly, crowdsourcing labels from public databases and community-driven forums served as a fast way to collect labels at scale given the time constraint of the research.

### *3.1.2 Safe Smart Contracts*

Aside from malicious labels, a negative sample of "safe" contracts also needs to be gathered so the model can differentiate from bad actors. The methodology for finding safe contracts was done through ranking contracts by user activity using the public database provided by Dune analytics[16] to select a balanced number of well-known contracts for comparison against the malicious dataset. To compile a total of 455 safe smart contracts, the following method is employed:

- Ranking by User Activity: Smart contracts are rank ordered based on the number of users they have had in the past 30 days, using Dune analytics. This ranking ensures a focus on contracts with significant and recent user engagement, reflecting their popularity and presumably indicating a higher level of trust.

- Limiting Top N Contracts: To create a population of safe smart contracts comparable in size to the malicious dataset, a specific number of top-ranked contracts is selected. This approach assumes that the most frequently utilized smart contracts are safe from scams and vulnerabilities.

For the identification of safe smart contracts, widely adopted contracts in the Web3 ecosystem with the highest volumes of organic activity are presumed to be labeled as "safe" for the purpose of this experiment.

This methodology aims to identify and gather safe smart contracts from the Web3 ecosystem, considering their widespread adoption and user activity as indicators of safety. The use of publicly available datasets, such as Dune analytics, provides transparency and replicability in the selection process, contributing to the reliability of the collected data for subsequent analysis.

### *3.2 Data Preprocessing*

The goal of data preprocessing to construct a feature matrix in which a vector for a smart contract is an exact encapsulation of the bytecode's architecture, almost like the DNA of a smart contract. To do this, two steps need to be taken:

1. Opcode extraction
2. Feature matrix creation

#### *3.2.1 Opcode Extraction*

In Solidity, opcodes[17] are low-level machine instructions executed by the EVM when smart contracts are deployed and interacted with on the Ethereum blockchain. They represent fundamental operations like arithmetic, storage manipulation, and conditional branching. These opcodes are crucial for understanding how Solidity code translates into bytecode, optimizing gas usage, ensuring security, and debugging smart contracts. Common opcodes include PUSH, POP, SSTORE, SLOAD, JUMP, JUMPI, CALL, and RETURN. Understanding opcodes is essential for developers to write efficient, secure, and reliable smart contracts on Ethereum.

The first step of data preprocessing is to transform collected smart contract bytecode by disassembling the hexadecimal representation into a comprehensive opcode list. This process was aligned with the foundational principles outlined in Ethereum's Yellow Paper, which provides the exact mappings of hexadecimal to opcode values.

*3.2.2 Feature Matrix Creation*

Once opcodes have been extracted in a list format, the next step of data preprocessing is to construct a feature matrix with 118 columns to be passed to the machine learning analysis. The feature matrix was crafted by quantifying the occurrences of each opcode identified during the disassembly process. This vectorized representation translated the Ethereum Yellow Paper's opcodes into numerical values, establishing a structured foundation for training machine learning models.

| | opcode | STOP | ADD | MUL | SUB | DIV | SDIV | MOD | SMOD | ADDMOD | ... | DELEGATECALL | CREATE2 | STATICCALL | TXEXECGAS | REVERT | INVALID | SELFDESTRUCT | SHA3 | SELFBALANCE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | [(PUSH1, 0x80), (PUSH1, 0x40), (MSTORE,), (PUS... | 2 | 78 | 4 | 11 | 4 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 22 | 7 | 0 | 12 | 2 |
| 1 | [(PUSH1, 0x80), (PUSH1, 0x40), (MSTORE,), (PUS... | 2 | 476 | 81 | 218 | 8 | 0 | 2 | 0 | 0 | ... | 0 | 0 | 2 | 0 | 143 | 31 | 0 | 17 | 0 |

Fig. III: Example of transformed feature matrix from preprocessing data before modelling.

*3.3 Model Selection*

In the exploration of machine learning techniques for predicting malicious smart contracts based on opcode patterns, various technologies were considered along with their strengths and weaknesses. The research yielded that the most feasible approach for the prediction use case with a labeled dataset would be to use supervised models.

Supervised learning techniques, such as Logistic Regression, Support Vector Machines, Decision Trees, Random Forest, Naive Bayes, and K-Nearest Neighbor, have been extensively evaluated using a labeled dataset. The labeled dataset contains instances of both safe and malicious contracts, allowing the models to learn and generalize patterns. Among these, Random Forest emerged as the most promising, showcasing high accuracy, precision, and balanced recall. Supervised learning techniques leverage known outcomes during training, making them well-suited for classification tasks where the goal is to predict predefined labels based on input features.

Supervised regression techniques were deemed as being the most reasonable way to detect malicious smart contracts based on a labeled dataset. The availability of labeled data allowed for the training of models to recognize patterns of bytecode that were linked to smart contracts with known malicious behavior. Using supervised learning, it

is important to ensure continuous refinement and adaptation and continuous training on new data may be necessary.

## 4. Results and Discussion

After narrowing down the scope of optimal machine learning models for classifying smart contracts as safe or malicious based on opcode patterns, a comprehensive evaluation of multiple supervised algorithms was conducted. The performance metrics, including accuracy, precision, and recall, were utilized to test the effectiveness of each model. Table I encapsulates the key findings:

| | **Accuracy** | Precision | Recall | **F1** |
|---|---|---|---|---|
| Random Forest | 85.17% | 86.36% | 80.00% | 83.06% |
| Logistic Regression | 81.82% | 75.22% | 89.47% | 81.73% |
| K-Nearest Neighbor | 77.99% | 72.07% | 84.21% | 77.67% |
| Decision Tree | 77.99% | 73.33% | 81.05% | 77.00% |
| Support Vector Machine | 75.60% | 69.64% | 82.11% | 75.36% |
| Naive Bayes | 71.29% | 64.46% | 82.11% | 72.22% |

Table I: Performance metrics for the 6 different algorithms under examination.

The evaluation of various machine learning models for the detection of malicious smart contracts provides valuable insights into the tradeoffs between using Random Forest and Logistic Regression. Both models exhibit distinct strengths and weaknesses, influencing considerations in selecting the most suitable approach for the task.

### *4.1 Random Forest*

Random Forest, with an accuracy of 85.17%, precision of 86.36%, and recall of 80.00%, emerged as the highest overall performance model. Random Forest is an ensemble learning method that combines multiple decision trees, making it good at capturing non-linear patterns and interactions within the data. However, the tradeoff includes increased complexity, and is significantly harder to interpret the inner workings of the Random Forest model compared to a simple Logistic Regression.

### *4.2 Tradeoff Analysis*

In the tradeoff analysis, Random Forest's higher accuracy and precision should be weighed against its increased complexity, potentially making it less straightforward to interpret and implement. Logistic Regression, while slightly worse in accuracy and precision, offers simplicity and interpretability. These are important factors in a productionized model where a clear understanding of the model's decision-making process may be more important than overall performance. Therefore, the choice between these two models ultimately depends on the specific requirements of the application.

### *4.3 Limitations*

Given the resource and timing constraints of this research, there are two major limitations when evaluating the provided methodology and model:

- Sample size: the research sample size is relatively small, and there is a potential concern of cherry-picking incidents for model training. A larger and more diverse sample could offer a more comprehensive representation of the diverse smart contracts present on the Ethereum blockchain. A broader dataset may lead to more reliable performance metrics and a better understanding of the generalization capabilities of the model. It is essential to recognize that the current findings might be influenced by the specific incidents chosen for analysis, and broader inclusivity in the dataset would enhance the robustness of the model.
- Emerging threats: the machine learning model developed in this research is trained on historical data, and its effectiveness is contingent on the patterns observed in past incidents. However, the ever-evolving landscape of Ethereum and, more broadly, of Web3.0 introduces new and unforeseen threats that might not be captured by the model. The inherent "lag" in the model's responsiveness to emerging threats could pose a limitation in providing proactive defense. Regular updates and adaptations to the model are necessary to address the dynamic nature of blockchain security threats.

## 5. Future Work

The research conducted lays the foundation for future endeavors aimed at refining and expanding the proposed model for identifying malicious smart contracts on the Ethereum blockchain.

### *5.1 Performance Testing*

While the research has provided valuable insights into the efficacy of the selected machine learning models, further performance testing is essential for the practical deployment of the model at scale. The computational speed of the chosen model, particularly when productionized, needs to be rigorously evaluated. Comparative analyses of computation times for different models, especially Random Forest and Logistic Regression, should be conducted to determine the most efficient and scalable solution. This aspect is pivotal for real-world applications where timely identification of malicious smart contracts is crucial.

### *5.2 Vector Database Solutions*

The landscape of machine learning and vector databases is dynamic, presenting opportunities for advancements that may enhance the accuracy and efficiency of predictive models. Exploring new methods of computation, such as vector database solutions, could offer improved predictive capabilities.

Vector databases store and query data represented as vectors, optimized for similarity searches and machine learning tasks.[18] They feature specialized indexing for high-dimensional spaces, supporting fast lookup operations. These databases scale horizontally for large datasets, employing distributed architectures. Integration with machine learning workflows is common, enabling real-time inference. Applications span image search, recommendation systems, genomics, and more. This type of database is particularly useful in applications where similarity search, recommendation systems, and machine learning tasks are common.

Investigating the integration of vector databases for the storage and retrieval of opcode vectors may contribute to faster model execution and scalability. Continuous monitoring of emerging technologies in this domain and their integration into the existing model architecture can be instrumental in refining the predictive power of the system.

### *5.3 Continuous Data Collection*

The dynamic nature of threats in the Ethereum blockchain highlights the importance of continuous data collection. Ongoing efforts to gather labels for malicious smart contracts are essential to keep the model updated and adaptive to evolving threat landscapes. Regular updates to the dataset will ensure that the model remains effective in identifying new patterns of malicious behavior. Collaborations with cybersecurity communities, monitoring forums, and leveraging real-time data sources, including social media, can contribute to a more comprehensive and up-to-date dataset.

### *5.4 Smarter Features*

Feature engineering[19] plays a pivotal role in the performance of machine learning models. Exploring different approaches to feature engineering can unlock potential improvements in model accuracy and precision. Investigating the inclusion of additional features, such as contract activity, historical transaction patterns, or developer reputation, could provide richer information for the model to discern between safe and malicious smart contracts. Systematic exploration and experimentation with smarter features will contribute to a better understanding of the underlying characteristics that separate malicious smart contracts from good.

### *5.5 Implementation and Integration*

The implementation and integration of the selected machine learning model into practical, user-friendly tools represent a critical step toward enhancing Ethereum security. The envisioned goal is to empower users with accessible and proactive defenses against malicious smart contracts. Two primary avenues for implementation and integration are explored: the development of a browser extension for MetaMask, a widely used Ethereum wallet, and the integration of an API for seamless interaction with specific wallet providers.

One potential implementation involves the creation of a browser extension, akin to existing antivirus tools, integrated with the popular MetaMask Ethereum wallet. The extension would serve as an additional layer of security, providing real-time analysis and warnings when users interact with smart contracts. This integration ensures a seamless user experience, as MetaMask is widely adopted within the Ethereum community.

The extension could leverage the machine learning model's predictions to notify users of potential threats, offering clear and actionable information. For instance, when a user attempts to interact with a smart contract, the extension could display a visual indicator or prompt, indicating the model's assessment of the contract's safety. This immediate feedback empowers users to make informed decisions and adds a valuable layer of protection against potential risks.

## 6. Conclusion

In response to the heightened risks within the Ethereum blockchain, this research presents a basic security model which leverages traditional machine learning techniques. The evaluation of supervised learning models yields promising results, with Random Forest and Logistic Regression emerging as the top two contenders.

Despite limitations in sample size and the model's tradeoff between complexity and interpretability, the paper outlines future work to address these issues. Key areas for improvement include performance testing, exploration of vector database solutions, continuous data collection, and different feature engineering approaches before being able to productionize the model in user-friendly tools for widespread adoption.

As the Ethereum blockchain evolves, the deployment of proactive security measures becomes crucial. This research contributes a model that has the potential to fortify the Ethereum ecosystem against the risks posed by malicious smart contracts. By combining machine learning strengths with user-friendly tools, the aim is to empower Ethereum users with accessible defenses, enhancing overall security and trust in Web 3.0.

## 7. Acknowledgement

The authors would like to acknowledge the research assistance from the Data Science program (2023 Fall) at University of California, BERKELEY, namely: Mahir Shah, Shannon Xiang, Tanay Appannagari.

## References

[1] EtherVM. (2021). Retrieved from https://www.ethervm.io

[2] Anchain.ai *"AnChain.AI 2023 Year-End Web3 Risk Report."* Retrieved from

https://www.anchain.ai/web3risk2023.
San Jose: AnChain.ai, 2024.

[3] Coinsbench. (2022). How Scammers Manipulate Smart Contracts to Steal and How to Avoid It. Retrieved from https://coinsbench.com/how-scammers-manipulate-smart-contracts-to-steal-and-how-to-avoid-it-8b4e4a052985

[4] Hu, Huiwen, Qianlan Bai, and Yuedong Xu. "Scsguard: Deep scam detection for ethereum smart contracts." *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2022.

[5] Huang, Jianjun, et al. "Hunting vulnerable smart contracts via graph embedding based bytecode matching." *IEEE Transactions on Information Forensics and Security* 16 (2021): 2144-2156.

[6] Feist, Josselin, Gustavo Grieco, and Alex Groce. "Slither: a static analysis framework for smart contracts." *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*. IEEE, 2019.

[7] Brent, Lexi, et al. "Vandal: A scalable security analysis framework for smart contracts." *arXiv preprint arXiv:1809.03981* (2018).

[8] Hu, Tianyuan, et al. "Detect defects of solidity smart contract based on the knowledge graph." *IEEE Transactions on Reliability* (2023).

[9] Chen, Jiachi, et al. "Defectchecker: Automated smart contract defect detection by analyzing evm bytecode." *IEEE Transactions on Software Engineering* 48.7 (2021): 2189-2207.

[10] Kushwaha, Satpal Singh, et al. "Systematic review of security vulnerabilities in ethereum blockchain smart contract." *IEEE Access* 10 (2022): 6605-6621.

[11] Rameder, Heidelinde, Monika Di Angelo, and Gernot Salzer. "Review of automated vulnerability analysis of smart contracts on Ethereum." *Frontiers in Blockchain* 5 (2022): 814977.

[12] Breidenbach, Lorenz. *An In-Depth Look at the Parity Multisig Bug* . (2017)

[13] Etherscan. (2024). Retrieved from https://etherscan.io/

[14] Etherscan Label Word Cloud. (2024). Retrieved from https://etherscan.io/labelcloud

[15] Web3Guard. (2023). Ethereum Ransomware Reports. Retrieved from https://web3guard.io/reports/ethereum?reporttype=ransomware

[16] Dune Analytics. (2024). Retrieved from https://dune.com/browse/dashboards

[17] Wood, G. “Ethereum: A Secure Decentralised Generalised Transaction Ledger. Yellow

Paper." Retrieved from https://ethereum.github.io/yellowpaper/paper.pdf (2018)

[18] Kemka, Martin, and Simon Kornblith. "An Introduction to Vector Databases." arXiv:2010.09886 [cs.DB].

[19] Brownlee, Jason. "Feature Engineering for Machine Learning." Machine Learning Mastery, 2020.