

# Securing Defi: a comprehensive review of ML approaches for detecting smart contract vulnerabilities and threats

Dhivyalakshmi Venkatraman, Manikandan Kuppusamy

School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology (VIT University), Vellore, Tamil Nadu, India

## Article Info

### Article history:

Received Jun 23, 2024

Revised Oct 19, 2025

Accepted Nov 5, 2025

### Keywords:

Explainable AI  
Machine learning  
Security attacks  
Smart contracts

## ABSTRACT

The rapid evolution of decentralized finance (DeFi) has brought revolutionary innovations to global financial systems; however, it has also revealed some major security vulnerabilities, especially of smart contracts. Traditional auditing methods and static analysis tools are prone to fail in identifying sophisticated threats, including reentrancy attacks, front-running, oracle manipulation, and honeypots. This review discusses the growing role of machine learning (ML) in enhancing the security of DeFi systems. It provides a comprehensive overview of modern ML-based methods related to the detection of smart contract vulnerabilities, transaction-level fraud detection, and oracle trust assessment. The paper also provides publicly available datasets, necessary toolkits, and architectural designs used for developing and testing these models. Additionally, it provides future directions like federated learning, explainable AI, real-time mempool inspection, and cross-chain intelligence sharing. While it is full of promise, the application of ML in DeFi security is plagued by issues like data scarcity, interoperability, and explainability. This paper concludes by highlighting the need for standardised benchmarks, shared data initiatives, and the integration of ML into development pipelines to deliver secure, scalable, and reliable DeFi ecosystems.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



## Corresponding Author:

Manikandan Kuppusamy

School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology (VIT University)

Vellore, Tamil Nadu, India

kmanikandan@vit.ac.in

## 1. INTRODUCTION

Decentralized finance (DeFi) disrupts financial services by facilitating peer-to-peer transactions using blockchain and smart contracts without legacy intermediaries. Although DeFi is transparent, accessible, and programmable, it is also increasingly targeted by security risks because it is open [1]. Smart contracts, although capable, are susceptible to coding errors and exploitation, evidenced by high-profile hacks such as the DAO hack and oracle attacks. Legacy security solutions, although helpful, are not scalable and do not keep pace with emerging threats [1]-[6]. Machine learning (ML), on the other hand, offers a promising solution by allowing adaptive, data-driven detection of anomalies and vulnerabilities and providing real-time DeFi system protection [7]-[11]. This study provides an in-depth examination of how ML methods are presently being applied and can be further optimised to improve DeFi security, namely in the areas of smart contract vulnerability scanning, oracle trustworthiness assessment, and transactional threat mitigation.

The scope of the endeavour is defined by three underlying goals:

1. To improve front-running detection algorithms by combining behavioural, transaction-level, and token-specific features with ML models. Front-running, where attackers reorder transactions in the mempool to earn arbitrage gains, is one of the most exploitable weaknesses in DEXs.

2. To create an ML-based system to estimate data trust value received through oracles, third-party services employed by smart contracts to retrieve real-world information (e.g., asset prices). Compromised oracles can result in improper contract execution, liquidation operations, and market manipulation.
3. Harness and experiment with sophisticated ML techniques, such as long-short-term memory (LSTM) networks, graph neural networks (GNNs), and transformer-based architectures, to identify structural and semantic vulnerabilities in smart contracts.

DeFi is a new financial system based on public blockchains such as Ethereum, in which smart contracts execute transactions in an intermediary-free environment [12]-[15]. Composed in programming languages such as Solidity and run on the Ethereum Virtual Machine, these contracts enable applications such as Uniswap, Aave, and DAI. DeFi provides transparency and openness, but composability and immutability create sophisticated risks. Interdependent contracts and immutable code make systems vulnerable to irremediable weakness [16]-[19]. Smart contracts, wonderful as they are, are not immune to coding bugs and design flaws. Certain vulnerabilities have led to the loss of vast amounts of money in DeFi [20]-[22]. Reentrancy is a serious smart contract vulnerability because a contract calls external code before changing its state, leaving the door open for an attacker's external contract to repeatedly call the original function and drain funds (as in the 2016 DAO hack that resulted in theft of more than \$60 million in Ether value) [23]-[26]. Integer underflow and overflow is another primary vulnerability because early Solidity versions did not include arithmetic checks, which allowed the flaw to be used by attackers to wrap values around (decrementing 0, e.g., results in  $2^{256} - 1$ ) to circumvent balance or limit checks, although since then it has been mostly eradicated by the introduction of SafeMath libraries and a better language [1]. Access control weaknesses are another risk, where buggy or exposed access modifiers (such as the ones that secure administrative actions, such as pausing a contract or minting coins) allow an attacker to take privileged actions and thus gain control of protocol behavior, mint unknown tokens, or drain reserve holdings [1], [9]. Oracle manipulation involves manipulating the external data sources (or oracles) on which smart contracts depend for their off-chain information (e.g., the price of assets), which the contracts then act upon based on false or stale information; as an example, in 2020 the bZx protocol was attacked through the manipulation of the price feeds, resulting in some \$1 million in losses by artificially mispricing assets [10].

## 2. FRONT-RUNNING DETECTION USING BEHAVIOURAL ML FEATURES

Front-running is one of the most persistent and damaging threats in DeFi, enabled by the transparent nature of blockchain systems [1], [12]. Unlike traditional financial markets, where insider trading is often hidden and illegal, blockchain technology allows all pending transactions to be visible in the mempool, a public queue of unconfirmed transactions [5], [13]. This gives malicious actors, often using bots, an opportunity to monitor, analyse, and manipulate these transactions for financial gain. ML, with its ability to detect behavioural patterns and sequence anomalies, has emerged as a powerful tool in detecting and potentially preventing front-running attacks in real-time [10], [14]. It is a blockchain exploit method in which profitable transactions are seen in the mempool, and competitive transactions with higher gas fees are added afterwards to secure them to execute first [9]. This exploit is especially effective in automated market makers (AMMs) such as Uniswap, where asset prices are dynamically determined based on executed trades, enabling attackers to benefit by executing in front of large transactions [15]. Displacement, where the first transaction is pushed out, resulting in it failing, insertion, where the attacker executes their transaction immediately in front of the target transaction to take advantage of expected price movement, and sandwiching, where attackers execute transactions in front of and behind the victim's transaction to take advantage of price movement from both sides, are typical front-running examples. These methods together are well-known examples of maximal extractable value (MEV) exploitation in blockchain networks [9], [15].

### 2.1. Traditional detection methods

Historically, front-running detection or prevention has consisted of looking for the same or similar transactions submitted in quick succession, the detection of inordinately high gas fees or precipitous gas price bid spikes, and the examination of swap size and slippage parameters on trades on DEXS [16]. These rule-based and reactive techniques have limitations because they are not successful against clever bots that are able to dynamically modify their actions. In addition, these techniques are likely to have high false positive rates and are not suited to the detection of the fine-grained temporal and behavioural characteristics typically displayed by MEV bot activity [11], [17].

### 2.2. Machine learning-based detection

ML models are highly apt to observe complicated sequences of transactions, movements of gas prices, and user behavioural patterns and, therefore, are particularly suited for detecting front-running attacks in blockchain systems. Key features leveraged in the models are gas price delta, or the differential of the

transaction gas fee compared with recent network averages, intervals in transaction times, swap rates, slippage settings created to identify sandwich attacks, user address behaviors showing typical patterns of historic transactions, and interaction frequency with specific DeFi contracts [1], [14], [18]. Typical ML approaches to detecting front-running are LSTM networks, demonstrating capability in sequential transaction pattern and temporal relationship learning, Isolation Forest algorithms, an unsupervised approach, that detects outliers based on the utilization of multidimensional feature space, supervised learning like random forests (RF) and support vector machines (SVMs) that rely on labelled data sets with recognized instances of attack; and DBSCAN clustering to find clusters reflecting bot-driven action or spikes in transactions [19].

**2.3. Challenges in traditional methods**

Over the last few years, numerous tools and frameworks have come into being to enhance the security analysis of Ethereum smart contracts. The tools utilize a variety of techniques such as symbolic execution, static analysis, fuzzing, formal verification, and ML to identify potential vulnerabilities. Every method has some advantages and disadvantages in detection potential, accuracy, and scalability. Symbolic execution-based tools like Oyente, Mythril, and Maian analyze Ethereum smart contract bytecode to detect logical errors and common patterns of weaknesses. Oyente was among the earliest tools that analyzed Ethereum bytecode, focusing on frequent problems including timestamp dependency and reentrancy. Mythril augments symbolic execution with better detection of arithmetic and reentrancy weaknesses and Maian is focused on trace weaknesses like suicidal contracts and infinite Ether leaks. Static code analysis tools like Slither, Securify, and SmartCheck analyse Solidity code without the requirement of execution. Slither is unique in that it is fast to process and provides full code metrics, thereby serving as a developer resource in the development phase. Securify employs pattern-matching techniques to adhere to security protocols for contract property checks, while SmartCheck employs XML translation to scan for vulnerabilities through known patterns. Fuzzing-based techniques such as ContractFuzzer and Harvey offer dynamic analysis through the creation of many inputs to reveal runtime vulnerabilities. Table 1 shows the key features of different frameworks. The several types of Ethereum smart contract analysis tools that are frequently used for security audits and vulnerability discovery are depicted in Figure 1.

Table 1. Key framework and features

S.No	Tool/Framework	Methodology	Key features
1	Oyente	Symbolic execution	Analyses Ethereum bytecode to detect standard vulnerabilities.
2	Mythril	Symbolic execution	Popular open-source tool for detecting reentrancy and arithmetic bugs.
3	Maian	Symbolic execution	Focuses on identifying trace vulnerabilities (e.g., suicidal contracts)
5	Securify	Static analysis + Patterns	Verifies compliance with predefined safety and liveness properties.
6	Slither	Static analysis	Efficient solidity code analyzer; provides security insights and metrics.
4	Smart check	Static analysis	Translates solidity to XML for pattern-based vulnerability detection.
7	Contract fuzzer	Fuzzing	Generates random inputs to detect runtime vulnerabilities.
8	Harvey	Greybox fuzzing	Uses genetic algorithms for smarter test case generation.
9	Contract ward	Machine learning	Uses feature learning to classify contract vulnerabilities.
10	ESCORT	Machine learning	Ensemble learning for multiclass vulnerability classification.
11	ZEUS	Formal verification	Combines abstract interpretation and model checking.

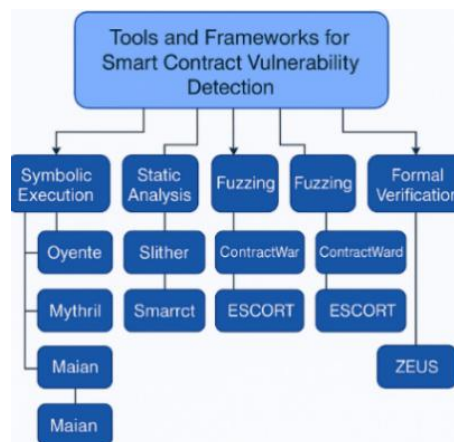


Figure 1. Category-wise Ethereum smart contract analysis tools

Though promising, ML-based front-running detection models are hampered by significant challenges, mainly due to the limited availability of accurately labelled data owing to the insufficient documentation of identified attacks, which hinders supervised training. The models also suffer from false positives since legitimate users can unknowingly display bot-like behaviour, e.g., charging excessive gas fees during network congestion. The requirement for real-time inference also makes things more difficult since blockchain transactions must be processed within milliseconds to avoid potential exploitation effectively. Additionally, high computational costs involved in running ML models on-chain make a solution involving off-chain inference combined with on-chain triggers necessary. Future work may involve the application of ML combined with streaming data pipelines, full address-level profiling, and oracle-based price monitoring to enhance detection. Integration of RL techniques may also enable adaptive defence mechanisms, allowing protocols to dynamically reorder, delay, or reject transactions based on learned behavioural knowledge.

### 3. EVALUATING ORACLE TRUSTWORTHINESS WITH MACHINE LEARNING

Oracles are key elements of DeFi systems, serving as middlemen that provide off-chain data like asset prices, exchange rates, and interest rates to on-chain smart contracts. [4.1] while they are valuable, oracle dependency introduces vulnerabilities, creating centralised points of failure in decentralised systems. Manipulated, delayed, or erroneous data from oracles can initiate incorrect executions of smart contracts, with severe financial consequences. To prevent these risks and enhance oracle reliability, researchers have suggested using ML methods to evaluate and authenticate the credibility of oracle-provided data in real-time [4.2].

#### 3.1. The problem with oracles

Smart contracts work deterministically based entirely on external data supplied by third-party sources called oracles, as they cannot access off-chain data directly. However, oracles can add vulnerabilities to smart contracts if poorly developed or compromised, or as a result of any contract being actively tampered with outside of the smart contract. Some examples of these flaws can include, but should not be limited to, price feed manipulation through low-liquidity pools, delay via data updates feeding outdated data to contracts, oracle centralised ability eroding access to external data resulting in a single point of failure, including allocated collusion or Sybil attacks by multiple oracle nodes. A poignant example of these risks is the bZx protocol hack of 2020, where attackers initially manipulated the price oracle, and took advantage of that price oracle based on that manipulation to extract funds from a contract by way of an undercollateralized loan-showcasing the wider observation of how the entire DeFi ecosystem suffers from the same vulnerabilities in the vulnerabilities of lending platforms, prediction markets, and synthetic asset services.

#### 3.2. Traditional mitigation techniques

To defend against oracle risks, several DeFi protocols proactively implement various defensive measures. These risk-minimizing defense strategies can employ methods such as data aggregation, which collects multiple data sources together to mitigate the effect of a single compromised input; time-weighted average prices (TWAPs), which average incoming prices over a certain period to smooth out any short-term volatility; and delay protocols, which can delay executing an order until a preset time has passed, allowing actors on the protocol to manually interact when the reported prices have dramatically changed. Even if these measures contribute to increasing protocol resilience, they are passive measures that do not creatively seek out new, or less obvious, neutral threats to the protocol. These measures also cannot adjust dynamically to try to evaluate the trustworthiness of multiple oracles in the marketplace, and they probably cannot respond to more subtle deviations in how the data flow has pattern shifted over time.

#### 3.3. ML-based trust evaluation models

ML is particularly advantageous for improving oracle security, by having real-time monitoring of the oracle data feed and identifying a pattern that is not typical. If we think about oracle outputs as data streams, we can use ML models to identify anomalies, assess reliability, estimate likely future behaviour, and compare it all across many oracles. Some of the relevant features these models will typically use are: deviations from expected prices, how often and how quickly oracles refresh their data; measures of volatility; any correlation structures across the oracles; and some historical accuracy metrics. Some of the ML techniques we would consider include: LSTM networks to identify anomalous conditions and future price estimates from time-series data; autoencoders as unsupervised models which could identify potential manipulations, through reconstruction error of unusually high values; anomaly detection models, such as isolation forest and DBSCAN, to identify unusual distributions of data and erratic behaviour patterns; and reinforcement learning (RL) which reflects a trust score based reward, deducting or adding trust scores dynamically to oracles with each transaction, letting performance give reward or penalty feedback signals for future learning.

#### 4. INTEGRATED ML-BASED DEFI SECURITY FRAMEWORK

The ever-mutating and complex risks that are to be mitigated in DeFi demand a cohesive and intelligent security environment that observes not only smart contract behaviour but also transaction dynamics, with off-chain data inputs being evaluated as well. This section describes the lining of a novel ML-driven framework to weave together three core layers of DeFi security: front-running attack detection, oracle-market assessment, and smart contract vulnerability evaluations. Each layer, using ML models trained for its own set of inputs like user transaction patterns, oracle data streams, or smart contract opcodes and logic structures, aims at another type of attack that is crucial.

Being modular and scalable, the framework proposed here ensures that each security layer can operate independently while feeding into a consolidated framework for decision-making. Such a consolidated architecture enables proactive threat detection, real-time risk alerts, and intelligent risk scoring across the entire pipeline of DeFi transactions. The system consolidates the correlation of insights from many sources, which enhances each detection mechanism while reducing false alarms and providing a more complete picture of potential threats. Furthermore, it offers great integration possibilities with any DeFi platform due to the common architectural base, supporting adaptive responses such as transaction delays, access restrictions, or automated audits from the security rating in real-time. This complete, ML-aided approach will not only fill the gaps posed by way of isolated detection mechanisms but would also evolve into an intelligent, context-aware defence mechanism that adapts promptly to the dynamic nature of the DeFi world. Through these interwoven, multi-layered ML techniques, the framework would secure decentralised financial systems better, giving them a durable, trustworthy foundation.

##### 4.1. Architecture overview

The framework proposed is capable of three separate and yet complementary ML-based security layers, with each serving to address a core aspect of DeFi protocol protection. The front-running detection layer exists to monitor mempool activity and transaction flow behaviours in real time. Using ML techniques, this layer identifies patterns where front-running attacks may occur. By monitoring gas price anomalies, transaction sequencing, and behaviour, this layer intends to flag suspect transactions even before they make their way onto the chain. The second layer is called the oracle trust evaluation layer. It assesses the trustworthiness of off-chain data sources that supply vital information such as token prices and exchange rates into smart contracts. On the other hand, this layer evaluates and scores oracle trustworthiness dynamically with anomaly detection algorithms, time-series analysis, RL, and more in order to discredit or prevent exploits caused by late, manipulated, or incorrect data. Thirdly, the smart contract vulnerability detection layer uses graph-based models such as GNNs and semantic analysis models such as transformers for the verification of smart contract codes against logic errors, unsafe function interactions, and other well-known vulnerability patterns such as reentrancy or access control flaws. ML driven security framework is shown in Figure 2.

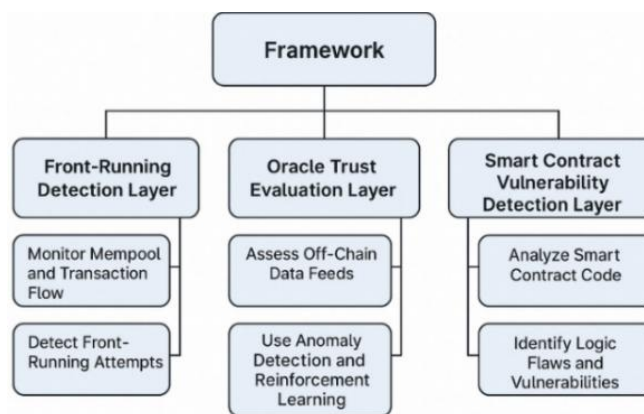


Figure 2. ML-driven security framework for DeFi

One good property of a security framework is that it allows all layers to function semi-independently so that it supports modular training, testing, and deployment. The modularity aspect of the framework gives it flexibility, whereby developers can take one or more layers as the case demands for a particular security breach without necessitating an overhaul of existing infrastructure. But together, these layers forge a

seamless and intelligent security ecosystem delivering a comprehensible-level protection-all the way through the DeFi stack. Meanwhile, this shared architecture enables communication and decision-making across layers, thus better correlating threats and reacting adaptively. As it integrates behavioral analysis, data integrity assessment, and code analysis, this framework will provide a robust and scalable solution to some of the various security problems affecting DeFi. Thus, the framework equips DeFi protocols with the ability to proactively defend against sophisticated attacks while ensuring the transparency, composability, and decentralization that characterize the entire ecosystem. Table 2 shows the integrated framework.

Table 2. Integrated framework

Layer	Primary function	ML models used	Input data sources	Expected output
Front-running detection layer	Detect front-running by analysing transaction behaviors	LSTM, isolation forest	Mempool transaction data, gas price fluctuations, slippage rates	Real-time alerts on suspicious transaction patterns
Oracle trust evaluation layer	Evaluate trustworthiness and integrity of oracle data	LSTM, RL	Oracle price feeds, update frequency, volatility metrics	Dynamic oracle trust scores and anomaly detection
Smart contract vulnerability detection layer	Analyze smart contract code for logic flaws and known vulnerabilities	GNN, CodeBERT (transformer)	Smart contract bytecode, Abstract Syntax Trees (AST), Slither static analysis outputs	Classification of vulnerabilities and logic flaw insights

#### 4.1.1. Layer 1: front-running detection

The front-running detection layer aims at the discerning errant activities in the Ethereum mempool with attention to front-running heuristic-based attacks. Features like gas price increase, slippage, transaction timings, and others are analyzed using LSTM networks and Isolation Forests to flag suspicious behavior. It studies flow of transactions over time to identify certain patterns, allowing for detection of MEV exploits. It is aimed at initiating mitigation strategies which will minimize damage before a transaction is committed to the blockchain.

#### 4.1.2. Layer 2: oracle trust evaluation

The oracle trust evaluation layer scans off-chain data sources that are pertinent for the operation of DeFi protocols. As DeFi protocols integrate oracles to fetch asset pricing data, any obstruction or altercation could lead to the improper execution of smart contracts. This layer implements time-series anomaly detection using LSTM models and scores oracles based on RL policies on their historical performance. This layer guarantees that DeFi systems function on reliable data by scrutinizing data feeds for contradictions.

#### 4.1.3. Layer 3: smart contract vulnerability detection

This layer analyses smart contract code to identify vulnerabilities and logic bugs using the smart contract vulnerability detection layer. It applies GNNs to analyse control flow and semantic structures, while transformer models like “Codebert” interpret “the intent and behaviour of the code.” This layer also incorporates bytecode and ASTs alongside outputs from static analysis tools like Slither, enabling the detection of security vulnerabilities like reentrancy, access control vulnerabilities, and logical flaws. This automated assessment increases the reliability of contracts before and after deployment.

## 5. DATASETS AND TOOLS FOR MACHINE LEARNING IN DEFI SECURITY

The most widely used is SmartBugs, a benchmarking dataset of hundreds of hand-labeled contracts for some classes of vulnerabilities, best utilized for comparative assessment of ML models and static analysis tools. Ethernaut, developed by OpenZeppelin, provides a gamified framework with vulnerable contracts, hence very well-suited for behavioral modeling and simulation of exploitative attacks. The HoneyBadger dataset is intended for honeypot contracts—suspicious smart contracts intended to catch attackers—so models learn about adversarial behavior and catch scams. REKT.News also provides detailed postmortems of actual DeFi attacks; while not an official dataset label, these can be converted to labeled samples for training. Altogether, these resources aid the construction of ML models for improving smart contract security. Table 3-5 shows the various bench mark datasets. There aren't many public data sets that merely record how well oracles do. There are some platforms, though, that reveal past feed data that can be used to train models to recognize problems or anomalies.

Table 3. Smart contract security datasets and resources

Dataset/Tool	Description	Access link(s)
SmartBugs dataset	Benchmark suite of Solidity contracts with labeled vulnerabilities	<a href="https://github.com/smartbugs/smartbugs">https://github.com/smartbugs/smartbugs</a>
Ethernaut	Game-based platform by OpenZeppelin with intentionally vulnerable contracts	<a href="https://ethernaut.openzeppelin.com/">https://ethernaut.openzeppelin.com/</a> <a href="https://github.com/OpenZeppelin/ethernaut">https://github.com/OpenZeppelin/ethernaut</a>
HoneyBadger dataset	Collection of honeypot smart contracts used to detect and study scam behavior	<a href="https://github.com/christofortorres/HoneyBadger">https://github.com/christofortorres/HoneyBadger</a>
REKT.News	Repository of real-world DeFi exploit postmortems with analytical insights	<a href="https://rekt.news/">https://rekt.news/</a>

Table 4. Datasets for front-running and transactional threat detection

Source/Tool	Description	Access link(s)
Etherscan	Provides historical transaction data, contract calls, and gas usage.	<a href="https://etherscan.io/">https://etherscan.io/</a> <a href="https://docs.etherscan.io">https://docs.etherscan.io</a>
Flashbots/MEV-explore	Offers insights into real-world MEV behaviour like sandwich attacks and arbitrage.	<a href="https://docs.flashbots.net/">https://docs.flashbots.net/</a> <a href="https://explore.flashbots.net/">https://explore.flashbots.net/</a> <a href="https://github.com/flashbots/mev-inspect-py">https://github.com/flashbots/mev-inspect-py</a>
Simulated mempool datasets	Used for generating synthetic transaction data in controlled environments for front-running analysis.	Ethereum Testnets (Sepolia, Goerli) <a href="https://trufflesuite.com/ganache/">https://trufflesuite.com/ganache/</a> <a href="https://tenderly.co/">https://tenderly.co/</a>

Table 5. Oracle data sources for ML-based trust evaluation

Source/Platform	Description	Access link(s)
Chainlink data feeds	Offers transparent, on-chain price feeds across multiple blockchains. Historical data helps assess reliability, latency, and price deviation.	<a href="https://data.chain.link/">https://data.chain.link/</a>
UMA oracle dashboard	Provides an optimistic oracle with dispute windows. Time-series data can be used for anomaly detection and forecasting models.	<a href="https://uma.xyz/">https://uma.xyz/</a>

## 6. CHALLENGES AND FUTURE POTENTIAL

ML models for smart contract security have come a long way but still suffer several drawbacks that today impede their practical deployment and real-time reliability in a DeFi environment. The principal difficulty they suffer is one of generalization wherein models trained on clean, labeled test contracts find it difficult to detect vulnerabilities in contract codes that are obfuscated, minified, or real-world contract-free codes that deviate from the usual patterns. There is also a concern about interpretability, especially in finance where transparency and auditability are crucial. Deep learning models are powerful, but the mechanism and process of how a certain vulnerability prediction was made by a specific tooling remains unclear to the developer, and auditor. Real-time training and inference on-chain are also prevented by computational requirements and Ethereum gas limitations. Going forth, the refinement of several critical areas will put an end to the above restrictions. In hybrid systems combining the deterministic reliability of static analysis tools with the adaptive learning capabilities of ML models, one achieves better accuracies coupled with an understanding of context. XAI methods are being studied to allow for transparency of the model so that users can follow the decision pathway and understand why a contract is marked as risky. Semi-supervised and self-supervised learning methods that can take advantage of huge repositories of unlabeled smart contracts are under development, in an attempt to minimize dependence on expert-labeled data. Also, embedding real-time ML-based scanning tools within IDEs or on deployment platforms would provide developers with security warnings before a deployment is even made on-chain. All these efforts intend to close the gap between the theoretical power of ML and the real-world needs of smart contract security in live DeFi ecosystems.

## 7. CONCLUSION

DeFi is still revolutionising the financial world, yet its fast growth has also brought with it tremendous security issues, especially regarding threats to smart contracts. This paper has given a comprehensive overview of emerging trends in ML methods for identifying and mitigating such risks. By comparing various ML frameworks ranging from supervised and unsupervised learning to deep learning and hybrid methods, we emphasise their ability to identify complex patterns of attacks, including reentrancy, front-running, and phishing. While in improving the security of DeFi platforms, significant gaps still exist that need to be filled. Availability of data, interpretability of models, real-time detection, and adversarial robustness are still open areas of concern. In addition, combining ML-based methods with current formal verification and audit tools may introduce even more robust and dependable security paradigms. Upcoming

research should emphasise the development of standardised datasets, explainable AI models for analysing smart contracts, and the development of robust, real-time defence systems. Closing the gap between theoretical ML research and real-world DeFi deployment will be key to the development of a secure and reliable decentralised financial future.

**ACKNOWLEDGMENTS**

We would like to thank our institution, Vellore Institute of Technology, Vellore (VIT University), for their invaluable support during the writing of this article.

**FUNDING INFORMATION**

The authors declare that no funding was received for this review work.

**AUTHOR CONTRIBUTIONS STATEMENT**

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Dhivyalakshmi Venkatraman	✓	✓				✓		✓	✓		✓			
Manikandan Kuppusamy				✓	✓					✓		✓		

- C : **C**onceptualization
- M : **M**ethodology
- So : **S**oftware
- Va : **V**alidation
- Fo : **F**ormal analysis
- I : **I**nvestigation
- R : **R**esources
- D : **D**ata Curation
- O : **O**riting - **O**riginal Draft
- E : **E**riting - **R**eview & **E**ditng
- Vi : **V**isualization
- Su : **S**upervision
- P : **P**roject administration
- Fu : **F**unding acquisition

**CONFLICT OF INTEREST STATEMENT**

The authors declare no conflicts of interest.

**DATA AVAILABILITY**

The data used in this study were obtained from publicly available sources, including Ethereum on-chain data, Uniswap transaction logs, and open MEV-related datasets. Any additional datasets supporting this study are available upon reasonable request to the corresponding author.

**REFERENCES**




- [1] J. J. Lohith, K. A. Manoj, P. G. Nanma, and P. Srinivasan, "TP-Detect: trigram-pixel based vulnerability detection for Ethereum smart contracts," *Multimedia Tools and Applications*, vol. 82, no. 23, pp. 36379–36393, 2023, doi: 10.1007/s11042-023-15042-4.
- [2] O. Zaaza and H. El Bakkali, "Unveiling the landscape of smart contract vulnerabilities: a detailed examination and codification of vulnerabilities in prominent blockchains," *International Journal of Computer Networks and Communications*, vol. 15, no. 6, pp. 55–75, 2023, doi: 10.5121/ijcnc.2023.15603.
- [3] Z. Li *et al.*, "VulHunter: hunting vulnerable smart contracts at EVM bytecode-level via multiple instance learning," *IEEE Transactions on Software Engineering*, vol. 49, no. 11, pp. 4886–4916, Nov. 2023, doi: 10.1109/TSE.2023.3317209.
- [4] S. S. Kushwaha, S. Josh, D. Singh, M. Kaur, and H.-N. Lee, "Ethereum smart contract analysis tools: a systematic review," *IEEE Access*, vol. 10, pp. 57037–57062, 2022.
- [5] Z. Wang and S. Guan, "A blockchain-based traceable and secure data-sharing scheme," *PeerJ Computer Science*, vol. 9, p. e1337, Apr. 2023, doi: 10.7717/peerj-cs.1337.
- [6] V. Merlo, G. Pio, F. Giusto, and M. Bilancia, "On the exploitation of the blockchain technology in the healthcare sector: a systematic review," *Expert Systems with Applications*, vol. 213, p. 118897, Mar. 2023, doi: 10.1016/j.eswa.2022.118897.
- [7] S. K. Lo *et al.*, "Analysis of blockchain solutions for IoT: a systematic literature review," *IEEE Access*, vol. 7, pp. 58822–58835, 2019, doi: 10.1109/ACCESS.2019.2914675.
- [8] Y. Xu, G. Hu, L. You, and C. Cao, "A novel machine learning-based analysis model for smart contract vulnerability," *Security and Communication Networks*, vol. 2021, pp. 1–12, Aug. 2021, doi: 10.1155/2021/5798033.
- [9] S. Aggarwal and N. Kumar, "Cryptographic consensus mechanisms," in *Advances in Computers*, vol. 121, 2021, pp. 211–226.
- [10] W. Wang, H. Huang, Z. Yin, T. R. Gadekallu, M. Alazab, and C. Su, "Smart contract token-based privacy-preserving access control system for industrial Internet of Things," *Digital Communications and Networks*, vol. 9, no. 2, pp. 337–346, 2023, doi: 10.1016/j.dcan.2022.10.005.






- [11] B. Jiang, Y. Liu, and W. K. Chan, "ContractFuzzer: fuzzing smart contracts for vulnerability detection," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, Sep. 2018, pp. 259–269, doi: 10.1145/3238147.3238177.
- [12] L. Zhang *et al.*, "CBGRU: a detection method of smart contract vulnerability based on a hybrid model," *Sensors*, vol. 22, no. 9, p. 3577, May 2022, doi: 10.3390/s22093577.
- [13] L. Zhang *et al.*, "A novel smart contract vulnerability detection method based on information graph and ensemble learning," *Sensors*, vol. 22, no. 9, p. 3581, May 2022, doi: 10.3390/s22093581.
- [14] L. Zhang *et al.*, "SPCBIG-EC: a robust serial hybrid model for smart contract vulnerability detection," *Sensors*, vol. 22, no. 12, p. 4621, Jun. 2022, doi: 10.3390/s22124621.
- [15] T. H.-D. Huang, "Hunting the Ethereum smart contract: color-inspired inspection of potential attacks," *arXiv preprint, arXiv:1807.01868*, Jul. 2018, [Online]. Available: <http://arxiv.org/abs/1807.01868>.
- [16] J.-W. Liao, T.-T. Tsai, C.-K. He, and C.-W. Tien, "SoliAudit: smart contract vulnerability assessment based on machine learning and fuzz testing," in *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, Oct. 2019, pp. 458–465, doi: 10.1109/IOTSMS48152.2019.8939256.
- [17] X. Yu, H. Zhao, B. Hou, Z. Ying, and B. Wu, "DeeSCVHunter: a deep learning-based framework for smart contract vulnerability detection," in *2021 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2021, vol. 2021-July, pp. 1–8, doi: 10.1109/IJCNN52387.2021.9534324.
- [18] Z. Feng *et al.*, "CodeBERT: a pre-trained model for programming and natural languages," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 1536–1547, doi: 10.18653/v1/2020.findings-emnlp.139.
- [19] S. A. Salloom, R. Khan, and K. Shaalan, "A survey of semantic analysis approaches," in *Advances in Intelligent Systems and Computing*, vol. 1153 AISC, 2020, pp. 61–70.
- [20] A. Al-Boghdady, M. El-Ramly, and K. Wassif, "iDetect for vulnerability detection in internet of things operating systems using machine learning," *Scientific Reports*, vol. 12, no. 1, p. 17086, Oct. 2022, doi: 10.1038/s41598-022-21325-x.
- [21] A. Haddad, M. H. Habaebi, M. R. Islam, N. F. Hasbullah, and S. A. Zabidi, "Systematic review on AI-blockchain based E-healthcare records management systems," *IEEE Access*, 2022.
- [22] M. Alaeddini, M. Hajizadeh, and P. Reaidy, "A bibliometric analysis of research on the convergence of artificial intelligence and blockchain in smart cities," *Smart Cities*, vol. 6, no. 2, pp. 764–795, Mar. 2023, doi: 10.3390/smartcities6020037.
- [23] R. Kumar, Arjunaditya, D. Singh, K. Srinivasan, and Y.-C. Hu, "AI-powered blockchain technology for public health: a contemporary review, open challenges, and future research directions," *Healthcare*, vol. 11, no. 1, p. 81, Dec. 2022, doi: 10.3390/healthcare11010081.
- [24] M. S. B. Kasyapa and C. Vanmathi, "Blockchain integration in healthcare: a comprehensive investigation of use cases, performance issues, and mitigation strategies," *Frontiers in Digital Health*, vol. 6, Apr. 2024, doi: 10.3389/fgth.2024.1359858.
- [25] M. Gu *et al.*, "Software security vulnerability mining based on deep learning," *Jisuanji Yanjiu yu Fazhan/Computer Research and Development*, vol. 58, no. 10, pp. 2140–2162, 2021, doi: 10.7544/issn1000-1239.2021.20210620.
- [26] R. Kiani and V. S. Sheng, "Ethereum smart contract vulnerability detection and machine learning-driven solutions: a systematic literature review," *Electronics (Switzerland)*, vol. 13, no. 12, 2024, doi: 10.3390/electronics13122295.

## BIOGRAPHIES OF AUTHORS



**Dhivyalakshmi Venkatraman**    was born in Arni, Tiruvannamalai District, Tamil Nadu, India. She obtained her master's degree in computer applications from Sri Balaji Chockalingam Engineering College, affiliated with Anna University, Chennai, Tamil Nadu, in 2013. Currently, she is pursuing her Ph.D. in Computer Science at Vellore Institute of Technology (VIT), Vellore, where she is actively engaged in cutting-edge research at the intersection of blockchain technology and machine learning. After completing her postgraduate studies, she began her academic career as an assistant professor in the Department of MCA at Sri Balaji Chockalingam Engineering College. Her research focuses on developing intelligent systems to detect and prevent smart contract attacks in DeFi, with particular emphasis on front-running and sandwich attacks on the Ethereum blockchain. Her broader interests include blockchain security, trustworthy AI, and applying machine learning in high-risk domains such as finance and cryptocurrency. She is currently exploring advanced detection methodologies using transaction-level and token-level features, as well as data integrity frameworks for DeFi oracles. She can be contacted at email: [dhivyalakshmi.v2023@vitstudent.ac.in](mailto:dhivyalakshmi.v2023@vitstudent.ac.in).



**Dr. Manikandan Kuppasamy**    is working as professor in the School of Computer Science and Engineering at Vellore Institute of Technology (VIT), Vellore. He received the best techno faculty award from ICTACT, Govt of Tamilnadu in 2014. He graduated in B.E. (CSE) with first class with distinction at Vellore Engineering College (Presently VIT) under the affiliation of University of Madras in 2004. He secured a Master of Engineering (M.E.) in CSE at Sona College of Technology, Salem under the affiliation of Anna University in 2006 with a University GOLD Medal. He Completed an MBA (Systems) at Alagappa University in 2008. He Completed a Ph.D. in the field of wireless networks at VIT in 2015. He has 18+ years of teaching and research experience. He has presented a number of papers at the National and International Conferences and published 50+ papers in reputed journals. His area of interest includes wireless networks, cloud computing, IoT, AI, and data science. He is a life member of CSI and IAEng. He can be contacted at email: [kmanikandan@vit.ac.in](mailto:kmanikandan@vit.ac.in).