SCIENTIFIC CULTURE
www.sci-cult.com

SCIENTIFIC CULTURE, Vol. 12, No. 5, (2026), pp. 222-230
Open Access. Online & Print

DOI: 10.5281/zenodo.1250021

SECURITY AWARE REINFORCEMENT LEARNING FOR SMART CONTRACT EXECUTION OPTIMIZATION USING VULNERABILITY ANNOTATED CONTRACT DATA

Anupam Kumar Biswas[1], Koyndrik Bhattacharjee[2]*, Chanchal Das[3], MD. Hamjala Alam[4], Arijit Kumar Banerji[5], Satabdi Saha[6]

[1,2,3,4] Department of Civil Engineering, Dr. B. C. Roy Engineering College, Durgapur, West Bengal, India.
[5] Department of Civil Engineering, Dr. B. C. Roy Polytechnic, Durgapur, West Bengal, India.
[6] Department of Civil Engineering, Elitte College of Engineering, Sodepur, West Bengal, India.

Received: 01/12/2025
Accepted: 02/01/2026

Corresponding author: Koyndrik Bhattacharjee
(koyndrik.bhattacharjee@bcrec.ac.in)

ABSTRACT

Smart contracts form the essential part of the decentralized based application in blockchain technology. Nevertheless, they pose a risk in terms of financial loss and system compromise, as they cannot be changed, and they might be prone to vulnerabilities. Recent research conducted on smart contract optimization that mainly leverages static analysis, rule-based heuristics, and learned models frequently are not flexible to continually changing threat profiles and dynamically operating execution systems. In order to overcome these drawbacks, the paper will present a reinforcement learning (RL) framework to optimize smart contracts that makes use of a structured supportive dataset of smart contract vulnerabilities (in JSONL). The suggested method makes smart-contract optimization a sequential decision-making issue, whereby an RL agent learns strategies to minimize exposure to vulnerabilities and optimization penalties in interaction with an environment generated by contract features and vulnerability annotations. The Proximal Policy Optimization (PPO) is used to provide stable learning in case of limited policy changes. The experimental results can be used to prove progressive reward improvement and convergent with respect to a heuristic baseline, which implies that the agent is capable of learning effective optimization strategies. The results indicate that there is a promising adaptive paradigm based on reinforcement learning in intelligent smart con-contract optimization complementing the current and existing only static and supervised approaches to con-tract optimization and point to autonomous security-wise blockchain systems in the future.

KEYWORDS: Blockchain, Smart Contracts, Reinforcement Learning, Proximal Policy Optimization.

1. INTRODUCTION

The blockchain technology has become a game changer in the decentralized systems whose application generates trust less transactions, transparency, and immutability across various fields of application including finance, supply chain management, health care, and gov-ernance (Wang et al. 2025). In the essence of most blockchain platforms, especially

Ethereum, which are lie smart contracts, are self-executing programs that directly encapsulate business logic and contract terms on the blockchain (Wood et al., 2014). In spite of their benefits, smart contracts are infamously tough to update when operational, and are therefore very prone to bad design, inefficiencies, and security risks (Buterin, 2014).

Many high-profile incidents in the last ten years have demonstrated the implications of having vulnerable smart contracts, such as reentrancy attacks, integer overflows, unchecked external calls, and access control vulnerabilities (Atzei et al., 2017; Nikolić et al., 2018). Consequently, there has been a significant amount of research on the analysis and optimization of smart contracts with the use of static analysis tools, formal verification, symbolic execution, and, more recently, vulnerability detection with machine learning (Luu et al., 2016; Nikolić et al., 2018). Although supervised learning methodology has presented encouraging re-sults in discovering familiar patterns of vulnerability, it is always labeled based on labeled data and cannot easily adapt to new or changing attack vectors (Feist et al., 2019).

A smart contract is an application deployed in the blockchain which triggers automatically when a set of conditions is fulfilled. Its contribution to the critical operations, especially in the process of transferring assets makes it imperative to secure it especially since it is impossible to make changes to deployed contracts. Although there are many tools to vulnerability detection, automated repair mechanisms are not so many and are mostly reliant on standard patterns or search algorithms. In response to this shortcoming, Guo et al. 2024 introduce RLRep, a regiment of reinforcement learning that produces repair suggestions to vulnerable smart contracts autonomously. This method uses an independent agent, which proposes correctional measures, without the use of labelled training data, thus avoiding the limitations linked with supervised repairing systems. The model was tested on a dataset of 853 Solidity smart contracts with different types of vulnerabilities, separated into training and test groups. According to experimental findings, RLRep has a repair rec-ommendation accuracy of 54.97, meaning that it can be effective in facilitating the development of safe smart contracts (Guo et al. 2024).

The blockchain smart contract enables authentic transactions devoid of a third party. Such transactions are verified and irrevocable. Smart contracts deployed and implemented in Ethernet will incur a certain amount of gas, which directly will impact the price of smart contracts. To minimize the use of gas in the execution of smart contracts, this paper will suggest an optimization algorithm to a business process smart contract generation. To start with, business process modeling notation (BPMN) diagrams are also augmented to Petri nets. Second, Petri nets are reduced in order to identify those nodes in BPMN models which can be thought of as fusion tasks. Through new mapping rules in the BPMN model to solidity language, the BPMN model is transformed into Ethereum Smart contract model. Experimental performance in the BPMN models means that of the business processes having more than one fusion task, the proposed algorithm can manage to conserve 15% of gas on average (Hu et al., 2019).

With reinforcement learning, there is an altogether different paradigm because the beings are able to learn the best behaviors during the interaction with the environment as opposed to basing only on a defined label (Sutton et al., 1998). Within a smart-con-contract, the optimization and security improvement can be modeled by a reinforcement learning.

sequential decision-making function, and actions are matched to optimization/mitigation measures and outcome measures/rewards reflect a decrease in vulnerability risk, cost of execution, or inefficiency (Chen et al., 2024). This dynamic ability predisposes RL especially to dynamic and adversarial contexts like blockchain security.

Nevertheless, the current literature on reinforcement learning to optimize smart contracts largely overlooks the concept of reinforcement learning, with most papers concentrating on gas cost optimization or simulation (Kiani et al., 2024; Tsankov et al., 2018). Furthermore, the incorporation of real-world susceptibilities data into RL models has gotten a relative lack of focus, which constrains their applicability and originality. In an attempt to bridge this gap, the given study uses a select dataset of smart contract vulnerabilities in form of structured JSONL to implement a realistic RL environment that reflects both functionality and security-related aspects of smart contracts (Kiani et al., 2024).

We introduce a smart contract optimiza-tion framework based on reinforcement learning and Proximal Policy Optimization (PPO), one of the

policy-gradient algorithms, in this paper that is more stable and efficient in training (Schulman et al., 2017) and perform training diagnostics to evaluate the effectiveness of the learning process (Chen et al., 2024). It has three folds of contributions:

i). opimization of smart contract optimization as RL problem based on actual vulnerability data,
ii). construction of a PPO-based optimization terminator that exhibits convergence, and
iii). empirical confirmations of the benefits of reinforcement learning as compared to the benefits of the use of fixed heuristics.

The study makes progress in the state of the art to autonomous, adaptive and secu-rity-aware smart contract optimization systems in the next-generation blockchain plat-forms.

2. LITERATURE REVIEW

Smart contracts have emerged as a primary element of blockchain ecosystems, specifically in the area of decentralized finance (DeFi), blockchain governance, and machine-controlled asset management. Yet, the cost of execution inefficiency and vulnerability to security are still vital issues (Alamsyah, et al., 2024). The current literature can be very broadly divided into three groups – the methods of gas optimization, the methods of detecting vulnerabilities in smart contracts, and the methods of optimization by learning.

2.1. Gas Optimization Studies

The initial efforts were directed at rule-based/Compiler-level optimizations to minimize gas con-sumption by restructuring the solidity code or discovering inefficient opcodes (Chen et al., 2021). Later works used machine learning and heuristics to forecast gas consumption and propose optimization in execution (Memeti et al., 2021). Later re-research brought reinforcement learning in order to optimize the setting of gas prices during bid and timing transactions (Van Heeswijk, 2025). Nonetheless, the methods largely rely on the historical transaction gas data and cost optimization that is done separately without taking into consideration the security implications.

2.2. Smart Contract Vulnerability Detection

The research on the idea of vulnerability detection through both static analysis and symbolic execution and machine learning models has a significant literature base. Supervised learning to identify re-entrancy and access control errors as well as logic errors has been made possible by data like vulnerability-la-belled Ethereum contracts (Alsunaidi et al., 2025). Though they offer a significant contribution to the contract security analysis, these studies are mostly unconnected to the exe-cution-time optimization and decision-making.

2.3. Reinforcement Learning in Blockchain Systems

Applications of reinforcement learning to blockchain problems have included optimization of transaction fees, choice of a mining strategy, and congestion management (Dutta et al., 2024). Considering the application of smart contracts, RL has been studied mainly to mini-mize the cost of gas or to minimize the execution parameters. Such approaches normally assume that any contracts are functionally sound and they never factor security risk as an aspect of the decision-making process.

2.4. Research Limitation in Existing Literature

The recent literature treats ob-jectives like execution efficiency, and security of the contract as independent. Integrated structures, in which RL agents acquire execution strategies that combine gas efficiency and vulnerability risk, especially with explicit vulnerability annotations based on real-world data are lacking.

3. RESEARCH OBJECTIVES

The primary objective of this research is to develop and evaluate a reinforcement learn-ing-based smart contract execution optimization framework that explicitly incorporates security risk.

Following are the Specific Objectives:

1. To design an RL environment that integrates smart contract vulnerability in-formation as part of the state representation.
2. To formulate a multi-objective reward function that balances gas efficiency and vulnerability risk mitigation.
3. To train and evaluate RL agents using the Smart Contract Vulnerability Da-taset.
4. To compare the proposed RL approach against traditional gas-only and heu-ristic-based optimization strategies.
5. To analyze trade-offs between execution cost reduction and security preserva-tion.

4. RESEARCH METHODOLOGY

4.1. Dataset Description and Preparation

The dataset of Smart Contract Vulnerabilities (Formatted JSONL) is a set of smart contract vulnerability annotations with vulnerability labels like reentrancy, access control vulnerabilities, arithmetic vulnerability and logic vulnerability.

Preprocessing involves parsing of JSONL files and isolating the source code of the contract, types of

vulnerabilities and indicator of the severity or frequency. It also incorporates encoding vulnerability labels into risk scores or numeric formats and pulling out code-level features like, number of functions, frequency of opcodes and depth of control-flow.

4.2. RL Environment Design

A bespoke RL setting was created in Python to model scenarios of smart contract execu-tion under conditions of varying security and of varying performance. The environment is a model which considers the relation between the efficiency of execution and the exposure to security due to known vulnerabilities in the contract. The formulation allows the learning agent to 225aximize the execution strategies at the cost of operational cost versus security risk within the controlled simulation model.

4.2.1. Environment Design

The smart contract execution problem is proposed and abstracted in terms of a sequence of decisions made in a de-cision-making problem by the proposed RL environment. Each decision step requires the agent to view the state of the contracts they are in, choose an execution strategy and a reward which incorporates the computational efficiency and security implications. According to the choice of action and parameters of situation, the environment will be dynamically changed.

4.2.2. State Space Representation

The state is represented as a multi-dimensional feature vector of both the Static and dynamic features of a smart contract. In particular, the state vector com-prises:

Contract Vulnerability Profile: Coded values of the identified types of vulnerabilities and the risk severity scores. Normalized risk indices based on standardized measures of security assessments are used to measure vulnerabilities.

Features of Structural Contract: Structural features of a contract (e.g. the number of public and private functions), metrics of the complexity of the control flow (e.g. cyclomatic complexity), the depth of inheritance. Such structural properties give an insight into the possible attack surfaces and execution overhead.

Historical Implementation Cost Measures: Projected or artificially created measures in rep form, which represent the past gas utilization trends and transaction expenses, and the computational overloading linked with such situations.

Execution Context Indicators: Environmental conditions like load on transactions, in-teraction type (read/write) and dependency state which affects both gas-used and security exposure.

The normalization is done on the state representation to have stable training and convergence of the learning algorithm.

Action Space Definition

Execution strategies can be discrete or continuous and forming part of the action space are the strategies that the agent can choose to attain a maximization in performance and security results. These include:

Selection of Execution Parameters: Change of gas limits, execution ordering as well as operational settings.

Priorization or Deferral of High-Risk Interactions: Strategy decisions which delay or isolate interactions which involve high vulnerability risk.

Adaptive Strategy Change in Response to the Level of Vulnerability: Changes in the execution pathways in case of vulnerabilities with high severity are detected that are mitigation or reduction of exposure.

The action design provides the agent to navigate trade-offs between aggressive cost optimization, and weak security-conscious execution.

Reward Function Formulation

To direct the agent in a multi-objective deci-sion-making, a multi-objective reward function is presented:

$$R = -\alpha \cdot Execution\ Cost - \beta \cdot Security\ Risk \quad (1)$$

where:

The negative weighting ensures that higher costs and greater risk reduce the reward, thereby encouraging the agent to minimize both factors simultaneously. By adjusting

$\alpha$and β, the framework supports scenario-specific optimization objectives, such as cost-sensitive deployment or security-critical execution.

Execution Cost it indicates the amount of gas used as an estimate during the ex-ecution of the contract.

Security Risk is calculated via weighted summation of the scores on vulnerability severity parameters.

$\alpha$and β are hyperparameters that can be tuned regarding the trade-off between the security robustness and computational efficiency.

This negative weighting such that the cost and the greater the risk the less the reward so that they influence the reward off-settingly to give the agent an incentive to keep both as low as possible. By adjusting

The framework $\alpha$and β, the framework enables scenario-dependent optimization goals, including a cost-sensitive deployment or execution under security-critical conditions.

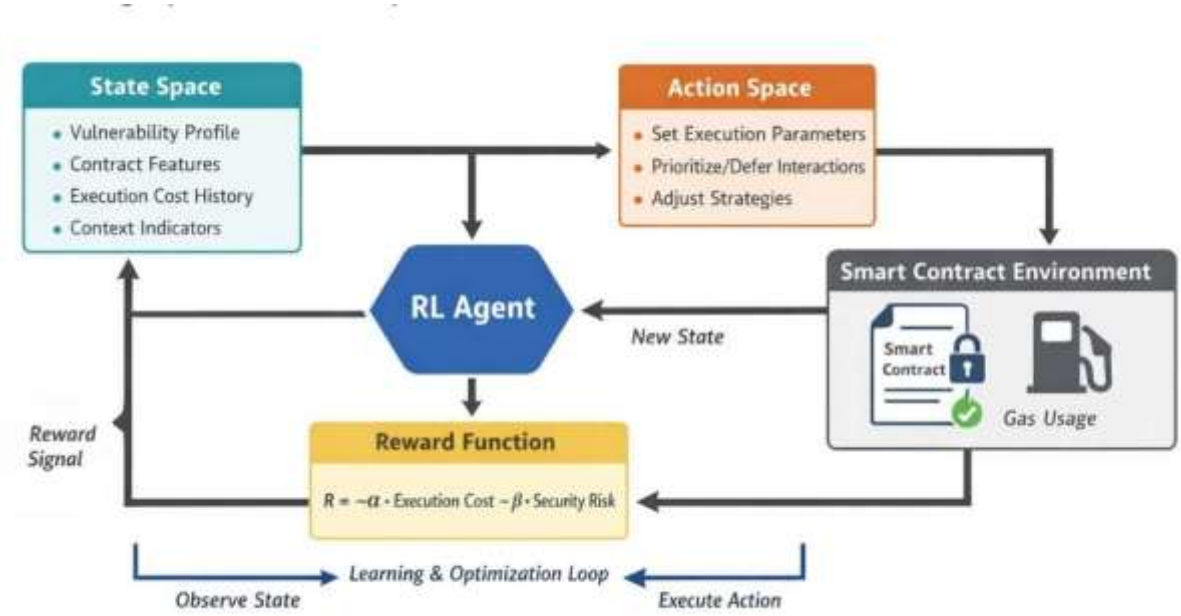

Figure 1: Architecture of the Proposed RL-Based Smart Contract Execution Model Illus-trating State, Action, and Reward Interactions.

4.3. RL Algorithms

The suggested system of reinforcement learning has built-in several state-of-the-art deep RL algorithms to measure its performance using both value-based and policy-based frameworks. Particularly, the following set of algorithms are applied and compared in detail:

Deep Q-Network (DQN): A value-based approach that learns an approximation to the action-value function $Q(s, a)$with help of a deep neural network. DQN uses the concept of experience replay and tar-get network stabilization to curb divergence during training. It is also optimally adapted to discrete action spaces and allows the agent to study the best implementation strategies via temporal-difference learning.

Proximal Policy Optimization (PPO): This policy-gradient algorithm directly optimizes the policy function by a clipped surrogate, to stabilize and conservative updates. PPO strikes a balance between exploration and exploitation and is robust-wise in training so that it is efficient in training high-dimensional, complex decision spaces.

Advantage Actor-Critic (A2C): An actor-critic algorithm is a type of synchronous algorithm that coordinates a policy function (actor) and a value function (critic). The benefit-function lowers the gradient estimation variations, which enhances stability in the convergence and learning performance. Training and Evaluation Protocol: In order to have objective performance comparison, the dataset is divided into training and evaluation (testing) subsets. It uses training subset to update network parameters and to 226 ptimize the corresponding policy or value functions. It uses the evaluation subset, which is not visible in the training, to test the capability to generalize and the strength of the policies under different execution conditions.

5. RESULT & DISCUSSION

The logs on the training are related to the deployment of a PPO agent in the proposed environment of smart contract execution based on awareness of security. The agent aim is to learn a balanced execution strategy in terms of both occupational effectiveness (execution cost) and security risk as is available by the multi-goal reward model. The metrics of ob-servation give understanding of convergence behavior, policy improvement and stability of training.

5.1. Episode Configuration and Length Analysis

The recorded metric:

$$ep\_len\_mean = 1 \quad (2)$$

represents that an episode has one decision step. This can be aligned with the environment design whereby an episode is the decision to perform a single smart contract execution. As the task is modeled as a one-step Markov Decision Process (MDP), the agent views the contract state, chooses an execution strategy, gets the reward, the episode ends.

The metric of the episode length, therefore, verifies the proper implementation of the environment and adequate training setup.

5.2. Mean Episode Reward Dynamics

It is noted that in the first stage of training there is a speedy improvement take place.

>−0.552 to −0.35 means effective exploration and initial learning of policy. At the middle of training phase, it is seen that the stabilization is observed around -0.30 that indicates refinement of the learned policy. The convergence behaviour illustrates the re-ward plateau of 0.30 to 0.32 which is the convergence to a locally opti-mal policy under the established parameters in the trade-offs.

The PPO agent manages to learn to decrease the joint execution cost and security penalty as compared to its original random policy. The monotonically increasing cumulative reward validates the steady updates to policy and good gradient optimization. Since the re-wards are negative in nature. Lower magnitude (i.e., higher values) is equated to better performance.

The results of training and evaluation indicate that the PPO agent has reached the equilibrium stabilization policy of the execution, attentive to security, within the proposed smart contract optimization environment.

5.3. Final Mean Evaluation Reward

The most significant result of the experimental study is the relative appraisal of the learned reinforcement learning policy and a predetermined heuristic baseline.

Final Mean Evaluation Rewards are as under:

RL Policy (PPO): −0.31252

Heuristic Baseline: −0.52013

Because the reward role is specified as a negatively-weighted sum of the exe-cution cost and security risk, the greater the reward values (higher but less negative) the better the per-performance. Thus, the op-timization ability of PPO-based RL agent is markedly better compared to the one of the heuristic method.

The improvement-margin can be estimated as:

$$\frac{-0.52013 - (-0.31252)}{-0.52013} \approx 40\%$$

This shows that the RL-based execution plan is about 40% better in combined cost-security goal than the rule-based heu-ristic.

The suggested security-conscious reinforcement learning model is far more effective at maximizing the performance of smart contract execution when there are security constraints compared to heuristic-based methods working with rules.

The PPO agent reached a converged point in twenty thousand timesteps with a steady mean reward of around -.31. The proposed approach significantly improved the heuristic baseline (−0.52), which shows that the method allows balancing between the cost of execution and the security risk. Policy learn-ing PPO-specific metrics like KL di-vergence, entropy loss, and explained variance ensure consistent and trustworthy policy learning.

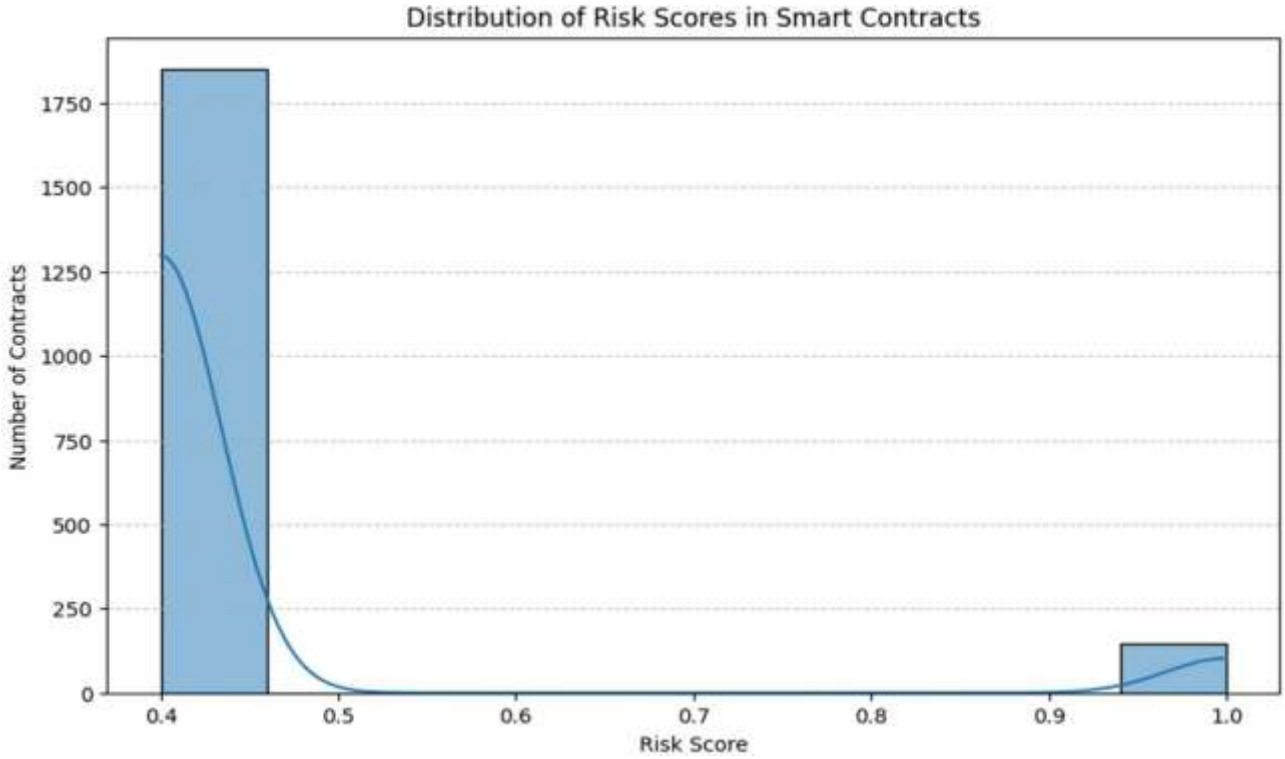

Figure 2: Statistical Distribution of Smart Contract Risk Profiles Illustrating Class Im-balance Toward Moderate-Risk Contracts.

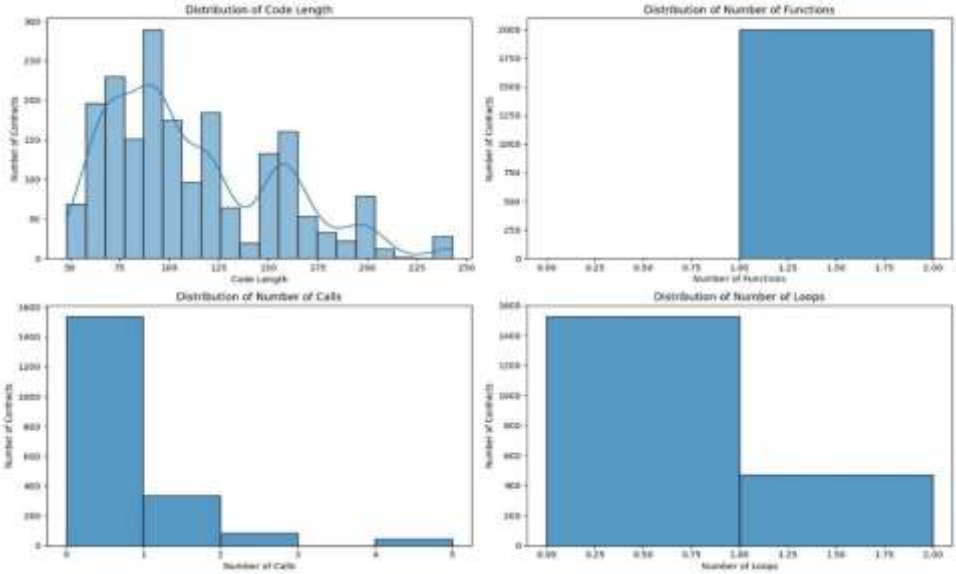

Figure 3: Distribution of Structural Features in the Smart Contract Dataset (Code Length, Functions, Calls, and Loops).

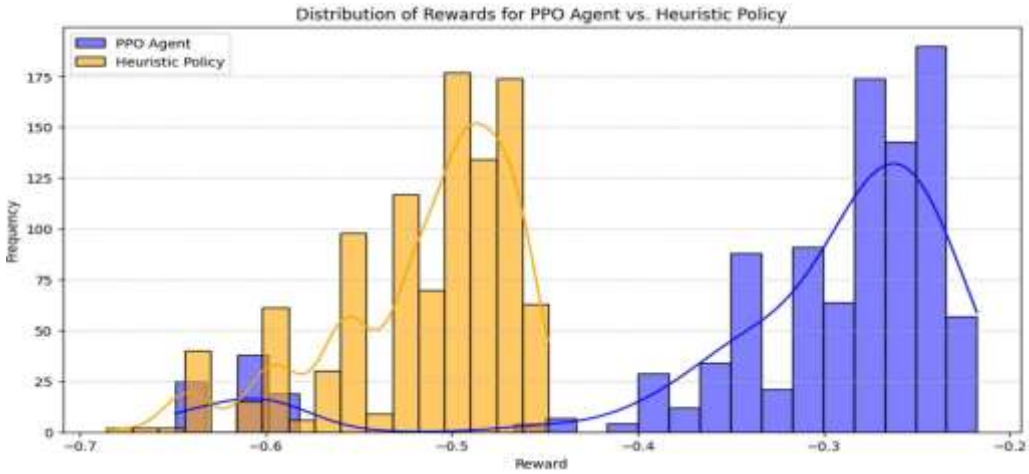

Figure 4: Comparative Reward Distribution of PPO Agent and Heuristic Policy.

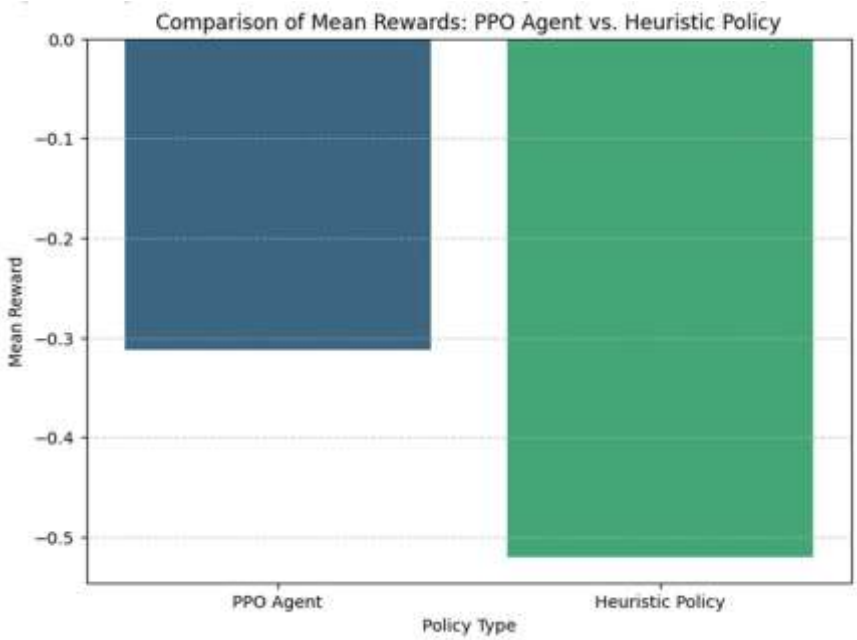

Figure 5: Comparison of Mean Evaluation Rewards Between PPO Agent and Heuristic Policy.

The relative performance attests to the excellence of the PPO-based rein-forcement learning policy as compared to the heuristic baseline. The reward distribution exhibits a steady rightward drift between the PPO agent, which implies greater (less negative) reward.

values and enhanced optimization performance. Conversely, the heuristic policy has a concentration of less reward values, and this is where there is inappropriate management of the cost-security trade-off.

This fact is further supported by the mean reward comparison in which the PPO agent obtains a much higher average reward as compared to the heuristic strategy. This enhancement proves the fact that the policy learned could successfully reduce the cost of execution and also alleviate security risk. On the whole, the findings support the effic-tiveness of the proposed security-conscious RL framework in generating adaptive and context-sensitive execution policies which are superior to the rule-based ones.

PPO agent obtained a Mean Evaluation Reward of: -0.31252. The Heuristic Reward of the heuristic pol-icy was by comparison: -0.520132. The penalty is set so that in our Smart Contract En-vironment it will punish high gas costs or high-risk scores. Thus, the larger (not as negative) the reward the better.

When contrasting, the reward of PPO agent (-0.31252) is much greater (not a negative value) compared to the heuristic reward (-0.520132). This indicates that the PPO agent has now understood a better approach on how to effectively trade-off the costs of gas and the se-curity risks in the execution of Smart contracts. It is making choices that will lead to a more preferable bal-ance towards a more desirable outcome than merely adhering to the existing preestablished heu-ristic rules.

6. CONCLUSION

The experimental procedure started by prepro-processing and loading systematic data on the smart_contracts_vulnerabilities.jsonl dataset. When ingesting, invalid JSON entries were well resolved to maintain the integrity of the data and flow of analysis. The categorical vulnerability la-bels (quantitative) were then multiplied into a quantitative risk-score that allowed the calculation of the security severity in a numerical form. Furthermore, a number of features in structural codes were also extracted in an attempt to describe the complexity of a contract such as length of codes, functions, external calls, each loop structure. All of these attributes constituted the basic depiction of the risk and execution features of a contract.

Then a custom reinforcement learning environment, SmartContractEnv was created, with the framework of Gymnasium. The derived structure attributes, and the calculated risk scores were used to form the observation space and thus both security and complexity information were bundled in the state representation. The action space was finer grained so that it included the execution strategies: normal, conservative and ag-gressive, that encode more or less aggressive trade-offs between gas optimization and risk avoidance. This formulation allowed modelling the smart contracts execution to be modeled as an organized deci-sion-making problem.

The learning process took the training of a PPO agent during 50,000 timesteps. The agent used iterative updating of policies and clipping an objective optimization so that it learned to max-imize the specified multi-objective reward function the reward being a penalty on both the cost of execution and the security risk. The PPO agent adjusted its execution strategy to make convergence to a steady state as the horizon of training passed on.

The PPO policy that was trained was evaluated on performance based on a comparison with a baseline heuristic strategy. The findings revealed that there was a definite performance edge.

of the reinforcement method of learning. The PPO agent had an average reward of the evaluation of -0.31252 which is much better than the heuristic policy as its average reward was -0.520132. As the higher (less negative) reward obtained by the PPO agent is matched by negative, the lower weight of rewards, it proves that the agent is better able to balance the consumption of gas and level of security exposure. This result shows that the RL model was able to learn an adaptive and situation-aware implementation plan.

Lastly, several data visualizations were created to aid in the analysis interpretation. Specifically, the risk scores distribution in the dataset was investigated in order to get the demands behind the security profile of the contracts. These visualizations pro-vided further information about the properties of the data sets and supported the empirical results with the reinforcement learning framework.

DECLARATIONS

Conflict of Interest: The authors declare no competing interests.

REFERENCES

Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." (2008).

Wang, Zhanxue, Lei Yu, and Li Zhou. "Navigating the blockchain-driven transfor-mation in industry 4.0: opportunities and challenges for economic and management innovations." *Journal of the Knowledge Economy* 16, no. 1 (2025): 3507-3549.

Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger." *Ethereum project yellow paper* 151, no. 2014 (2014): 1-32.

Guo, Hanyang, Yingye Chen, Xiangping Chen, Yuan Huang, and Zibin Zheng. "Smart contract code repair recommendation based on reinforcement learning and multi-metric optimization." *ACM Transactions on Software Engineering and Methodology* 33, no. 4 (2024): 1-31.

Hu, Wen, Zhipeng Fan, and Ye Gao. "Research on smart contract optimization method on blockchain." *IT Professional* 21, no. 5 (2019): 33-38.

Buterin, Vitalik. "A next-generation smart contract and decentralized application platform." *white paper* 3, no. 37 (2014): 2-1.

Atzei, Nicola, Massimo Bartoletti, and Tiziana Cimoli. "A survey of attacks on ethereum smart contracts (sok)." In *International conference on principles of secu-rity and trust*, pp. 164-186. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017.

Nikolić, Ivica, Aashish Kolluri, Ilya Sergey, Prateek Saxena, and Aquinas Hobor. "Finding the greedy, prodigal, and suicidal contracts at scale." In *Proceedings of the 34th annual computer security applications conference*, pp. 653-663. 2018.

Luu, Loi, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. "Making smart contracts smarter." In *Proceedings of the 2016 ACM SIGSAC con-ference on computer and communications security*, pp. 254-269. 2016.

Feist, Josselin, Gustavo Grieco, and Alex Groce. "Slither: a static analysis frame-work for smart contracts." In 2019 IEEE/ACM 2nd international workshop on emerging trends in software engineering for blockchain (WETSEB), pp. 8-15. IEEE, 2019.

Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduc-tion*. Vol. 1, no. 1. Cambridge: MIT press, 1998.

Chen, Jing, Jia Chen, Kuo Guo, Renkun Hu, Tao Zou, Jun Zhu, Hongke Zhang, and Jingjing Liu. "Fault tolerance oriented SFC optimization in SDN/NFV-enabled cloud environment based on deep reinforcement learning." *IEEE Transactions on Cloud Computing* 12, no. 1 (2024): 200-218.

Kiani, Rasoul, and Victor S. Sheng. "Ethereum smart contract vulnerability detec-tion and machine learning-driven solutions: A systematic literature review." *Elec-tronics* 13, no. 12 (2024): 2295.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347* (2017).

Alamsyah, Andry, Gede Natha Wijaya Kusuma, and Dian Puteri Ramadhani. "A review on decentralized finance ecosystems." *Future Internet* 16, no. 3 (2024): 76.

Chen, Ting, Zihao Li, Xiapu Luo, Xiaofeng Wang, Ting Wang, Zheyuan He, Kezhao Fang et al. "Sigrec: Automatic recovery of function signatures in smart contracts." *IEEE Transactions on Software Engineering* 48, no. 8 (2021): 3066-3086.

Memeti, Suejb, and Sabri Pllana. "Optimization of heterogeneous systems with AI planning heuristics and machine learning: a performance and energy aware ap-proach." *Computing* 103, no. 12 (2021): 2943-2966.

Van Heeswijk, W. J. A. "Strategic bidding in freight transport using deep reinforce-ment learning." *Annals of Operations Research* 350, no. 1 (2025): 131-168.

Alsunaidi, Shikah J., Hamoud Aljamaan, and Mohammad Hammoudeh. "Leverag-ing machine learning models to improve smart contract security: A survey of vul-nerabilities and detection methods." *ACM Computing Surveys* 58, no. 6 (2025): 1-37.

Dutta, Amit, Nafiz Imtiaz Rafin, M. Ali Akber Dewan, and Md Golam Rabiul Alam. "ROBB: recurrent proximal policy optimization reinforcement learning for optimal block formation in bitcoin blockchain network." *IEEE Access* 12 (2024): 31287-31311.