mathematics

MDPI

Article

Deep Learning-Based Malicious Smart Contract and Intrusion Detection System for IoT Environment

Harshit Shah [1], Dhruvil Shah [1], Nilesh Kumar Jadav [1], Rajesh Gupta [1,*], Sudeep Tanwar [1,*], Osama Alfarraj [2], Amr Tolba [2], Maria Simona Raboaca [3,4,*] and Verdes Marina [5]

[1] Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad 382481, India
[2] Computer Science Department, Community College, King Saud University, Riyadh 11437, Saudi Arabia
[3] Doctoral School, University Politehnica of Bucharest, Splaiul Independentei Street No. 313, 060042 Bucharest, Romania
[4] National Research and Development Institute for Cryogenic and Isotopic Technologies—ICSI Rm. Vâlcea, Uzinei Street, No. 4, P.O. Box 7 Râureni, 240050 Râmnicu Vâlcea, Romania
[5] Faculty of Civil Engineering and Building Services, Department of Building Services, Technical University of Gheorghe Asachi, 700050 Iasi, Romania
* Correspondence: rajesh.gupta@nirmauni.ac.in (R.G.); sudeep.tanwar@nirmauni.ac.in (S.T.); simona.raboaca@icsi.ro (M.S.R.)

Abstract: The Internet of Things (IoT) is a key enabler technology that recently received significant attention from the scientific community across the globe. It helps transform everyone's life by connecting physical and virtual devices with each other to offer staggering benefits, such as automation and control, higher productivity, real-time information access, and improved efficiency. However, IoT devices and their accumulated data are susceptible to various security threats and vulnerabilities, such as data integrity, denial-of-service, interception, and information disclosure attacks. In recent years, the IoT with blockchain technology has seen rapid growth, where smart contracts play an essential role in validating IoT data. However, these smart contracts can be vulnerable and degrade the performance of IoT applications. Hence, besides offering indispensable features to ease human lives, there is also a need to confront IoT environment security attacks, especially data integrity attacks. Toward this aim, this paper proposed an artificial intelligence-based system model with a dual objective. It first detects the malicious user trying to compromise the IoT environment using a binary classification problem. Further, blockchain technology is utilized to offer tamper-proof storage to store non-malicious IoT data. However, a malicious user can exploit the blockchain-based smart contract to deteriorate the performance IoT environment. For that, this paper utilizes deep learning algorithms to classify malicious and non-malicious smart contracts. The proposed system model offers an end-to-end security pipeline through which the IoT data are disseminated to the recipient. Lastly, the proposed system model is evaluated by considering different assessment measures that comprise the training accuracy, training loss, classification measures (precision, recall, and F1 score), and receiver operating characteristic (ROC) curve.

Keywords: blockchain; artificial intelligence; intrusion detection system; internet of things; malicious smart contract

MSC: 68M25

Citation: Shah, H.; Shah, D.; Jadav, N.K.; Gupta, R.; Tanwar, S.; Alfarraj, O.; Tolba, A.; Raboaca, M.S.; Marina, V. Deep Learning-Based Malicious Smart Contract and Intrusion Detection System for IoT Environment. *Mathematics* **2023**, *11*, 418. https://doi.org/10.3390/math11020418

Academic Editors: Daniel Ramotsoela, Adnan M. Abu-Mahfouz, Bruno Silva, Umair Mujtaba Qureshi and Zuneera Umair

Received: 18 December 2022
Revised: 9 January 2023
Accepted: 10 January 2023
Published: 12 January 2023

1. Introduction

The rise of Internet of Things (IoT) devices has decisively stood out in the industrial market to orchestrate the massive roll-out of applications, such as smart cities, smart healthcare, smart grids, and smart homes. The IoT devices are equipped with smart sensors that obtain data from the surroundings and process it to obtain intuitive information,

such as managing energy efficiency in smart grids, temperature and pressure control in smart industries, and optimizing route patterns in vehicular communication. Moreover, it paved the way for individuals to remotely connect and control their IoT devices using smartphones, tablets, and computers via the Internet. These devices are synergistically interconnected over the Internet to accomplish the shared objective of the application (e.g., smart home systems). As a result, IoT applications are progressively becoming a part of people's daily lives that enhance their quality of life. Despite its remarkable advantages, the IoT environment is still in its infancy from the perspective of security and privacy. The IoT ecosystem comprises heterogeneous devices, where each device has a colossal amount of data relaying via the insecure Internet. Moreover, the communication link between IoT devices is vulnerable to various security threats, such as volumetric attacks (distributed denial of service (DDoS)) [1], data integrity attacks, session hijacking, and man-in-the-middle attacks. Furthermore, the current IoT systems are centralized in nature; thereby, it suffers from a bottleneck of low scalability. This is because as the number of IoT devices increases, the authentication, authorization, and control and management become unmanageable by the current centralized IoT systems. Therefore, the IoT administrator has to employ large data servers for the information exchange, resulting in a cost-expensive IoT environment. Moreover, the scalability deteriorates if the central server is compromised or goes down by the well-known DDoS attack [2,3]. Therefore, there is an imperative requirement for organizations to invest in modern technological support to confront the aforementioned security issues.

Recently, blockchain has attracted a lot of attention and emerged as a breakthrough technology to resolve security and scalability issues in the IoT environment [4,5]. It has unique capabilities, such as immutability, decentralization, transparency, and high reliability, that are tempted by the research community across the globe. For instance, Rodrigues et al. in [6] proposed a queue-theoretical analytical model to improve the processing efficiency of data transactions and the integrity of stored data in the blockchain-enabled IoT network. Further, the authors of [7] explored the optimal block assignment of the blockchain technology in the wireless IoT system. Their proposed solution ensures that each blockchain node efficiently verifies every transaction between the IoT nodes. Further, genetic algorithms are adopted to optimize the block assignment problem in the blockchain network. Their results show that the proposed solution improves the convergence time and occupancy storage rate. However, they have not considered the security parameters that restrict the data integrity and volumetric attacks in the IoT ecosystems.

Ren et al. in [8] presented a joint solution for the low-latency, task offloading, and security provision problems in wireless body area network-based IoT systems. Toward this aim, the authors first proposed a decision-making algorithm to manage resource-constrained IoT devices. Then, they employed blockchain technology to offer a robust security solution for data tampering, authentication, and cyber-physical attacks. Nevertheless, there is no security provision for the smart contract used to authenticate the incoming data. Then, Ref. [9] presented a resource constraint problem of IoT devices in the smart healthcare system. They employed an Ethereum-based blockchain technology to address the load distribution between the resources in the IoT environment. Here, they consider rich and thin clients based on their resource capacity, i.e., high resources (rich clients) and low resources (thin clients). Only rich clients can participate in the blockchain mining process, whereas thin clients can only access the blockchain and participate in the transaction process. Their proposed solution outperforms in terms of efficiency compared to other baseline IoT-enabled blockchain architectures. However, the security prospect is not considered, for instance, the integrity of a smart contract is not verified, i.e., whether it is manipulated or not.

From the literature, we can observe that most of the blockchain-based security solutions are centered on secure data storage and task offloading for IoT applications [10,11]. However, none of them verify the authenticity of the smart contract, which is an integral part of the blockchain network. Smart contracts verify and authenticate the data that are

required to be stored in the blockchain immutable ledger. Earlier, this was accomplished using various third-party intermediaries (middlemen), which manually check and authenticate the data. However, these intermediaries are non-trusted and can exploit the operations of the blockchain network. On the contrary, smart contracts are programmable code that automates the agreement workflow, i.e., when certain conditions are met, possible actions get executed. Nevertheless, there is a possibility that malicious users maneuver the smart contracts, where it accepts the non-secured data and discards the secure data from the blockchain network, deteriorating the security performance of the IoT systems. Recently, artificial intelligence (AI) captured the IoT market by offering staggering benefits in the field of healthcare [12], banking, agriculture, and robotics. If offers predictive services, intelligence, automation, and security for various real-world applications. To respond to the aforementioned challenges, the authors in [13,14] utilized AI-based algorithms and an attention mechanism to detect reentrancy programs (a single program executes multiple outputs) and vulnerabilities in the smart contracts. However, the above-mentioned approaches do not consider all aspects of the security threats, such as the access control, time manipulation, low-level calls, denial of service, and bad randomness vulnerabilities in smart contracts.

As mentioned above, a lot of work has been performed in aligning the blockchain with IoT systems. Nevertheless, the smart contracts used in the blockchain network are not properly assessed for their malicious behavior. They merely checked the possible security threats while deploying smart contracts and how to eliminate them from the IoT applications. In order to create a system that considers all aspects of security, there is a need for an architecture that has selective permeability for users to deploy the smart contract. By analyzing the network data of a user, an AI model can detect multiple attacks, such as the DDoS, access control, and many more. Once it ensures that the user does not possess malicious intent, the second level of security, analyzing smart contracts for vulnerabilities, is executed. This shall lead to enhanced security because the main two concerns of security concerning blockchain are addressed. To tackle the above-mentioned security issues, this paper presents an AI-based malicious smart contract (using deep learning (DL)) and intrusion detection system for blockchain-enabled IoT systems. First, the proposed system examines the user's network attributes to predict if the user may possess malicious intent or not. Different AI-based algorithms are incorporated, such as decision trees, logistic regression, random forest, multi-layer perceptron, and Naive Bayes, that efficiently predict the user's behavior, i.e., malicious or non-malicious, by utilizing the features of the standard dataset. In the next stage, the proposed system model collects smart contracts from non-malicious users that want to store their sensitive IoT-based data in the blockchain; for that, this paper used DL-based algorithms that classify the smart contracts, i.e., a malicious and non-malicious smart contract. If the smart contract is safe, it can deploy (validate the IoT data) on the blockchain. Finally, this paper evaluates the performance of the proposed system model using various performance parameters, such as the training accuracy, classification measures, and blockchain scalability.

1.1. Motivation

A smart contract helps business owners conduct their business and trade securely and privately without third-party services. This reduces the operating cost for maintaining the authenticity and legitimacy of the IoT applications. The present research has primarily focused on static smart contract analysis, code issues or bug identification, and intrusion detection, with little attention to the AI-based classification of smart contracts (malicious or safe). However, a combined work of intrusion detection for malicious users and DL-based smart contract classification has yet to be discussed. Thus, there is a need to design a system which first detects malicious users through intrusion detection, followed by the smart contract classification i.e., malicious and non-malicious.

1.2. Novelty

In recent years, there has been a major rise in the use of blockchain technology across various domains, such as the IoT, to offer significant advantages, such as transparency, decentralization, immutability, and security. Moreover, DL is a subfield of AI that uses neural networks to learn patterns and make decisions based on that learning. Using blockchain, we can tackle data injection attacks in IoT systems, where the attackers cannot tamper with the data. The scientific community has given many indispensable solutions to tackle security threats, where first smart contracts validate the roles and permissions and then the data are forwarded to the blockchain immutable ledger. However, it is observed from the literature that many researchers have not explored the malicious behavior of users and their deployed smart contracts. For example, a malicious user (intruder) can deploy their malicious smart contract in the blockchain to deteriorate the performance of IoT systems. This research gap is not yet fully explored by the global research community. Therefore, there is a stringent requirement for a secure architecture that first classifies malicious users and their deployed smart contracts. Toward this aim, this paper proposed an AI and blockchain-based architecture where the first ML classifier classifies the malicious users using an intrusion detection system dataset. Then, a malicious smart contract dataset is used, where DL algorithms are trained to bifurcate malicious and non-malicious smart contracts. So, the proposed architecture allows only non-malicious smart contracts to store their data in the public blockchain.

1.3. Contributions

The following are the major contributions of this paper.

- This paper proposes an AI-based system model that first classifies the IoT users as malicious and non-malicious from their respective IoT environments. Only the data of the non-malicious user are dispatched to the IoT application, which is associated with blockchain technology to offer tamper-proof storage.
- This paper utilizes DL models to efficiently predict a malicious and non-malicious smart contract before it validates or authenticates the data of non-malicious users. The dual objective of the proposed system model improves the security and privacy of the IoT systems.
- The proposed system model is evaluated using different performance assessment measures, such as the training accuracy, classification measures, and blockchain scalability, to substantially improve the simulation results for the proposed system model.

1.4. Organization

The organization of this paper is as follows. Section 2 discusses the related work comprised of a recent literature review. Section 3 shows a brief introduction to the system model and the objective function formulated for the proposed work. Section 4 shows a detailed description of our multi-layered system model. Section 5 presents the performance analysis of our proposed system model. Lastly, Section 6 concludes the proposed work by highlighting the research findings.

2. Related Works

The research community makes several solutions with their insightful viewpoints to mitigate the security issues of the IoT ecosystem. For example, Xiao et al. in [15] proposed an AI and blockchain-enabled secure and intelligent model for smart city applications, i.e., a natural gas IoT system. The authors used the long short-term prediction (LSTM) model to predict the deliverability of natural gas. Further, blockchain allows buyers and sellers to maximize the chances of obtaining transaction contracts. Their results outperformed the prediction and transaction matching; however, the security aspect on the natural gas IoT system was not explored. Next, Das et al. in [16] presented an AI and blockchain-based key management protocol for cyber-physical systems. Here, they acquired the data using fog servers from the smart device, then verified it with the blockchain miners to

add it to the blockchain's decentralized ledger. They employed the benefits of AI in the private blockchain to secure IoT devices from potential cyber-physical system attacks. Their results showed that the proposed scheme has less computational complexity than other state-of-the-art works. Then, the authors in [17] studied issues such as trustworthiness and authentication in wireless sensor network-based IoT systems. Toward this challenge, they employed public and private blockchain that maintains the authentication between the IoT-IoT node and the IoT node and cluster head. Moreover, AI algorithms were utilized to confront denial-of-service (DoS) attacks that target the cluster head to exploit the authentication and registration of the IoT nodes. Nevertheless, both [16,17] did not consider security threats, such as man-in-the-middle, data integrity, and session hijacking attacks. Moreover, there was no security provision for smart contracts. Although, a few researchers have given solutions to deal with smart contract shortcomings. For instance, Li et al. in [18] reviewed a few smart contract deployment issues, such as a low-speed updating mechanism, recompilation, and redeployment. For that, they proposed a new smart contract deployment architecture and optimization technique using the application-oriented instruction for blockchain-based IoT systems. The results showed that their proposed model has an efficient update latency and reduces the ledger size and gas consumption compared to the existing baseline works.

Then, ref. [19] presented a smart contract static analysis tool called SmartCheck to comprehensively classify solidity code issues using XML-based representation that analyzes XPath patterns in the solidity code. However, their solution is not adaptable to the new vulnerabilities introduced in recent years. The authors in [20] dealt with reentrancy bugs in Ethereum-based smart contracts using a fuzzy-based analyzer. It focused on one of the most common bugs encountered in Ethereum smart contracts; however, they have not incorporated AI models, due to which there is no bifurcation between malicious and non-malicious data. So, a smart contract has to process both malicious and non-malicious data, resulting in high computational overhead to the system. The authors of [21] classified smart contracts into four classes, such as greedy, suicidal, prodigal, and safe. Then, they implemented a symbolic analysis-based tool, i.e., "MAIAN", that validates and exhibits real-time exploits in the smart contract. The result showed that their proposed solution efficiently finds the real exploit at a true positive rate of 89%. However, the solution was limited to source code vulnerabilities. It does not discuss the passive attacks that jeopardize the performance of the entire system. In [22], the authors presented a novel definition of lightweight smart contracts that provides a high level of confidence, even in the absence of blockchain. By separating requirement specifications from smart contract implementations using special objects, referred to as historical objects, the approach provides transparency, immutability, and protection against corrupted or erroneous smart contracts. The authors proposed a framework that enables the creation of smart contracts using an imperative, executable language. Adding a new object, named the history object, it stores the history of transactions pertaining to the contract, i.e., the messages for method calls and method returns, which offers a runtime verification of a contract's defined behavioral features, as well as control over the safety, security, and privacy, as well as a trusted asset transfer. However, their work entirely depends on blockchain technology, where they have not utilized the essential properties of AI algorithms, which can significantly improve the performance of IoT systems. For instance, if the blockchain-based smart contract has malicious code, then there is no checkpoint that classifies the smart contract as "malicious" or "non-malicious". This also implies that the blockchain has to store malicious data in its immutable ledger, degrading the applications' performance. Liao et al. [23] demonstrated a vulnerability assessment approach, termed SoliAudit. SoliAudit uses machine learning and fuzz testing to check for vulnerable smart contracts. The results of their experimentation are up to 90% in terms of vulnerability identification. They have not considered intrusion detection attacks for their system. Further, Alkadi et al. [24] presented a deep blockchain framework (DBF) built to provide a privacy-based blockchain with smart contracts and security-based distributed intrusion detection in IoT networks. They utilized a DL technique, namely

BiLSTM (Bidirectional long short-term memory) to train a model for the UNSW-NB15 and BoT-IoT datasets. Their analysis provides a thorough architecture and highly accurate results on the respective datasets. However, they do not consider the vulnerability of smart contracts themselves. Similarly, Gupta et al. [3] presented a malicious smart contract detection system to analyze the vulnerability of smart contracts using DL. They used the LSTM, GRU, and ANN for experimentation purposes which result in a accuracy close to 99%. However, they did not explore the possible intrusion attacks on the IoT network. Table 1 shows the comparative analysis of the proposed system model with the state-of-the-art work approaches.

Table 1. Comparing the existing works with the proposed work by considering different parameters.

Author	Year	Objective	Pros	Cons
Xiao et al. [15]	2021	Blockchain and AI-powered secure natural gas IoT system	Results that outperform prediction and transaction matching	Security of this system is unexplored
Das et al. [16]	2022	AI and blockchain-enabled key management for industrial cyber physical systems	Less complexity than other works	Security aspect from various attacks is unexplored
Ismail et al. [17]	2022	Secure authentication for wireless sensor networks in IoT using blockchain	Security and authentication is explored	Security from various attacks is unexplored
Li et al. [18]	2022	Proposed a new smart contract deployment architecture to overcome certain smart contract shortcomings	Better results for update latency, reduced ledger size and gas consumption	Deals only with shortcomings of smart contracts
Tikhomirov et al. [19]	2018	Presented a smart contract static analysis tool called SmartCheck to provide a comprehensive classification of code issues in solidity	SmartCheck helps improve security by detecting code issues in solidity	Unable to predict unknown vulnerabilities
Chao Liu et al. [20]	2018	Presented ReGuard to detect reentrancy bugs in Ethereum smart contracts	Focuses on most common security bug in smart contract	Only single attack vector is explored; cannot predict unknown vulnerabilities
Nikolic et al. [21]	2018	Proposed a system classifying smart contracts as greedy, prodigal, suicidal, and safe	A tool called MAIAN was developed to classify the smart contracts into four categories	Not detected IDS attacks
Owe et al. [22]	2022	Proposed a new type of smart contract known as lightweight smart contract	Preserves trust in blockchain system	Computationally expensive (AI is not incorporated)
Liao et al. [23]	2019	SoliAudit: Smart Contract Vulnerability Assessment	Utilizes machine learning and fuzz testing to analyze smart contract	Security from intrusion attacks is not explored
Alkadi et al. [24]	2020	A deep blockchain framework with secure intrusion detection in IoT networks	Uses DL to secure the IoT network from intrusion attacks	Vulnerability of smart contracts is not explored
Gupta et al. [3]	2022	DL-based malicious smart contract detection in IoT network	DL-based vulnerability assessment of smart contracts	Intrusion detection in IoT network is not explored
Proposed System	2022	To secure blockchain-enabled IoT systems using AI	Explores security aspects of blockchain-enabled IoT systems against various attacks and vulnerable smart contracts	-

3. System Model and Problem Formulation

In this section, we formulate our objective function that has two objectives, i.e., to secure the IoT ecosystem from intrusion attacks and to enhance the security and privacy of blockchain-based smart contracts. Then, based on the formulated problem, a systematic system model is designed, which briefs the proposed solution to tackle the security threats of the IoT environment. In the IoT environment, there are numerous devices, which are

remotely managed by the users to complete the shared task. Let us consider a user entity set $\{u_1, u_2, u_3, \ldots, u_n\} \in U$ that controls IoT devices, such as $\{d_1, d_2, \ldots, d_m\} \in D$, where a user u_1 has one or many d_l.

$$\{d_1, d_2, \ldots, d_m\} \in D \text{ where,} \tag{1}$$

$$u_i \in d_i \text{ or,} \tag{2}$$

$$u_i \in \{d_1, \ldots, d_i\} \tag{3}$$

Here, an intruder (u_o) frames a malicious packet to exploit an authenticated IoT device (d_i); further, u_o can also place an already compromised IoT device (d_o) to deteriorate the performance of IoT environment.

$$u_o \xrightarrow{\text{malicious packet}} d_i \tag{4}$$

$$s.t.\ u_o \notin \{u_1, u_2, u_3, \ldots, u_n\} \tag{5}$$

$$u_o \xrightarrow{\text{place}} d_o \xrightarrow{\text{in}} \{d_1, d_2, \ldots, d_m\} \tag{6}$$

$$s.t.\ d_o \notin \{d_1, d_2, \ldots, d_m\} \tag{7}$$

To confront the intrusion attacks, researchers have adopted AI and blockchain-based security solutions where AI uses classification algorithms to bifurcate malicious and non-malicious data packets. On the other hand, blockchain securely stores the non-malicious data packets in the decentralized immutable ledger. For that, the non-malicious data packet has to first pass the smart contract security (authenticate the data using various authentication conditions) where, based on certain predetermined conditions, the data get stored in the blockchain ledger. In the proposed system, we assume the user entity set $\{u_1, u_2, u_3, \ldots, u_n\} \in U$ that develops and deploys the different smart contracts (S) for their respective IoT applications. Each $u_i \in U$ are easily tracked by their unique $ID_i \in \{ID_1, ID_2, ID_3, \ldots, ID_n\}$. Every user is passed through the malicious user detection layer, where each user is classified n times, i.e., malicious and non-malicious. A counter $\{x_1, x_2, x_3, \ldots, x_n\} \in X$ is associated with ID of each user, i.e., $ID_i : X_i$, which signifies number of times a user u_i is detected malicious.

$$\exists X \in \{u_1, u_2, u_3, \ldots, u_n\} \in U \tag{8}$$

Assume $x = 0$, the user is classified p times. If the count of user detected "malicious" is $x_i > p/2$, then that user is suspended and would not able to deploy smart contract.

$$ID_i : x_i \begin{cases} x_i > p/2 & \text{user is suspended} \\ x_i \leq p/2 & \text{user can deploy smart contract} \end{cases} \tag{9}$$

The safe users $\{su_1, su_2, su_3, \ldots, su_r\} \in \text{SU}$ after passing through the user detection layer can safely deploy smart contracts $\{s_1^j, s_2^j, s_3^j, \ldots, s_t^j\} \in S$. Now, safe users su_i create the contract s_i and forward it to the data preparation layer to prepare it for training purposes. After that, the smart contract is compiled into two entities, i.e., bytecode (B) and opcode (O), which are encoded by utilizing one-hot encoding which on compilation formed feature vector (FV_i) for given contract s_i. The opcodes are part of a machine-language instruction that specifies the action to be taken. There are 256 types of opcodes, such as addition, multiplication, division, store, etc. As opcodes contain categorical data, we cannot directly give inputs to the AI models, so the one-hot encoding technique is used. It converts categorical data into binary vectors where 1 represents the feature's existence. We prepared our dataset by encoding these opcodes using a one-hot encoding technique. Categorical variables are transformed through this procedure into numeric variables that AI algorithms can use to make predictions more accurately. The maximum number of possible different

opcodes generated from smart contracts can be 256. The one-hot encode vector (V_{hot}^{k}) of size 256×1 is prepared for each opcode, where 1 represents the existence of that instruction and the rest of all the values are 0.

$$V_{hot}^{k} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \cdots \begin{bmatrix} 0 \\ 0 \\ 1 \\ . \\ . \\ . \\ 0 \end{bmatrix} \cdots \begin{bmatrix} 0 \\ 0 \\ 0 \\ . \\ . \\ . \\ 1 \end{bmatrix} \tag{10}$$

After that, the one-hot encoding vector (V_{hot}) is created by adding all the instruction vectors V_{hot}^{k}

$$\underset{256 \times 1}{V_{hot}} = \sum_{k=1}^{t} V_{hot}^{k} \tag{11}$$

$$V_{hot} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ . \\ . \\ . \\ 0 \end{bmatrix} + \cdots + \begin{bmatrix} 0 \\ 0 \\ 1 \\ . \\ . \\ . \\ 0 \end{bmatrix} = \underset{256 \times 1}{\begin{bmatrix} 8 \\ 5 \\ 3 \\ . \\ . \\ . \\ 7 \end{bmatrix}} \tag{12}$$

where t depends on length of opcode (O). The term V_{hot} is represented as the input feature vector. The feature vector contains the frequency of each opcode. Training dataset is prepared by obtaining V_{hot} vector of each smart contract. The prediction probability (P) provides the probability of the smart contract being malicious or not.

$$P = \sigma(WF + b) \tag{13}$$

where W is weight, b is bias, and σ is a sigmoid function. A threshold value, i.e., 0.5, is set, where $P < 0.5$ is considered as safe contract or else a malicious contract.

$$S : \begin{cases} safe & P < 0.5 \\ vulnerable & P \geq 0.5 \end{cases} \tag{14}$$

Based on the P, i.e., $P < 0.5$, the smart contract is not malicious; this allows the smart contract to deploy and securely store the IoT data in the immutable blockchain ledger. The two-way security of intrusion detection and smart contract classification using feature vectors with the help of DL-based models helps to improve the security of the IoT ecosystem. Based on the aforementioned security requirements, we formulated our objective functions, i.e., to maximize the security of user and smart contracts in the IoT systems.

$$\mathbb{O} = \max \sum_{i=1}^{n} \text{Secure}(U + S) \tag{15}$$

where $U + S$ is the user and its associated smart contract.

4. The Proposed System Model

In this section, we present the proposed system model consisting of four layers, i.e., deployment, malicious user detection, blockchain layer, and application layer, that

confronts different security threats to improve the security and privacy of IoT applications. Figure 1 shows an exemplary illustration of the proposed system model. A detailed description of each layer of the proposed system is as follows.

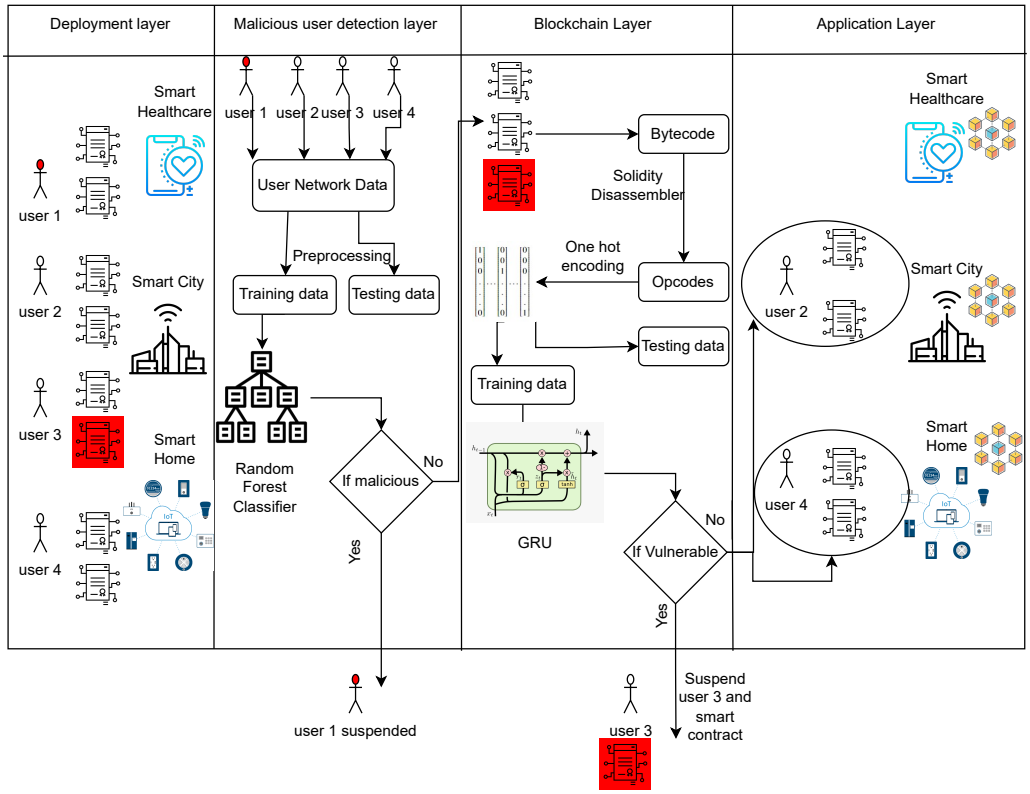


Figure 1. Proposed system model.

4.1. Deployment Layer

The deployment layer is the first layer presented in our architecture, which is the initial system of the Internet of Things (IoT). Here, we considered any IoT-based application, such as smart healthcare, smart homes, and smart cities, where our security system may be applied. This layer facilitates the data exchange between different components of the IoT ecosystem. In the IoT environment, there are numerous devices which are remotely managed by the users to complete the shared task. Let us consider a user entity set $\{u_1, u_2, u_3, \ldots, u_n\} \in U$ that controls IoT devices, such as $\{d_1, d_2, \ldots, d_m\} \in D$, where a user u_1 has one or many d_l.

$$\{d_1, d_2, \ldots, d_m\} \in D \text{ where,} \tag{16}$$

$$u_i \in d_i \text{ or,} \tag{17}$$

$$u_i \in \{d_1, \ldots, d_i\} \tag{18}$$

The layer also comprises malicious users trying to jeopardize the operation of the IoT environment. Using various IoT-based attacks, they can exploit the IoT systems for personal benefit and cause major disruptions. Attacks, such as denial-of-service (DOS), privilege escalation, probing, and session hijacking attacks, are used to disrupt the overall performance of IoT systems. In order to avoid such nefarious attacks, there is a need for an automated and intelligent system that bifurcates the malicious behavior, i.e., malicious and non-malicious user from the IoT systems. To accomplish this task, this layer takes

the network data for malicious user detection and passes them on to the malicious user detection layer.

$$\tau, S \longrightarrow \textit{malicious user detection layer} \tag{19}$$

The deployment layer is the one that has first contact with the malicious user. There may be three cases, the first being that a user is safe and is also deploying a safe smart contract. This is the ideal case where a user should be safely allowed to deploy a smart contract on the blockchain. Secondly, it may be that the user is safe but deploys a malicious smart contract that should not find a way into the blockchain. Lastly, it may be possible that the user has malicious intent, so they should not be allowed to deploy a smart contract. The case where a user has malicious intent and deploys a vulnerable smart contract would not occur under ideal situations because the layer for malicious user detection would detect it and suspend the user. This system proposes to secure the blockchain from the last two cases. First, there is a check for the malicious user, and if the user is predicted to be safe by the system, it allows them to store their data in the blockchain's immutable ledger. By the final layer, the user and the smart contract would have been checked for unwanted activities.

4.2. Malicious User Detection Layer

This layer is the first security layer that activates as soon as a user requests. It checks the credibility of the user to ensure if the user is malicious or not. This is achieved by passing the user's network information to a machine learning model. This machine learning model is trained for the purpose of predicting whether the user is malicious or not.

4.2.1. Dataset Description

The dataset used for the simulation of malicious user detection is X-IIoTID [25]. X-IIoTID is a dataset specifically for IIoT (Industrial Internet of Things) systems. The heterogeneous nature and interoperability requirements of IIoT systems are well suited by its connectivity- and device-agnostic capabilities. Deep packet inspection tools were used to extract the essential characteristics from the network and system logs. With its innovative and extensive capabilities, X-IIoTID provides a unique perspective on the dangers and assaults that IIoT networks and systems are subject to. It has 820,834 instances and 59 features (without labels). The labels are of three sorts, one being a binary classification of "Attack" and "Normal", second being a multi-class classification of 9 attacks, and third being another multi-class classification of 18 attacks.

4.2.2. Data Preprocessing

This dataset is used to distinguish the connections based on network parameters (τ), and to predict whether the system is being attacked or not, and if the system is being attacked. The dataset was distributed and had the feature and class variables put separately. Considering the application, we use the binary classification labels that distinguish between a "Normal" or "Attack" request. Hence, we get rid of the other two class columns of multi-class classification. We also get rid of set of unnecessary features, such as IP addresses and ports. The dataset had multiple values missing or null, and hence needed to be replaced. We replaced each missing value in a column by the median. Once the presence of null values and missing values was dealt with, we needed to convert the categorical features to numeric values. For example, the values of column "Protocol" were either "tcp", "udp", or "icmp". They were mapped to 0, 1, and 2, respectively. Next, we proceeded to scale the dataset in 0 to 1 so as to avoid any feature having bias. Finally, we split the data into training and testing set for experimentation. The training set to testing set ratio is 70:30 (574,583 training instances and 246,251 testing instances).

4.2.3. Prediction

We trained our data with multiple AI models to make prediction between safe or bad connection. On basis of the features present, the model is able to identify whether the user has a malicious intent. Because the features contain fields such as "duration", "protocol", and different rates between source and target, it is possible that a single prediction may not be completely accurate. Hence, this prediction should be made for p counts, depending upon the capacity of the system being used where p is any odd number.

$$p = 2k + 1 \text{ where k} \geq 0 \tag{20}$$

The single prediction on data may not be accurate, so prediction should be made more than one time. To avoid situations when there are an equal number of safe and vulnerable predictions, the value of prediction p is taken an odd number of times, depending upon the capacity of the system. This odd number is given by k whose value is greater than or equal to 0. The value of k would be less than half of the system's capacity. So, by predicting p times if the count of user predicted malicious will be greater than count of user predicted safe, then it will be suspended, and they will not be able to deploy smart contract. Now, in order for safe users to deploy smart contracts, they are passed through malicious smart contract detection layer to detect for any vulnerable contract.

4.3. Blockchain Layer

The blockchain layer is one where the smart contracts play a major role. For any task, there will be smart contracts issued by the user to be validated. This layer receives the list of safe users $\{su_1, su_2, su_3, \ldots, su_r\} \in$ SU and their respective smart contracts $\{s_1^j, s_2^j, s_3^j, \ldots, s_t^j\} \in S$ from deployment layer and malicious user classification layer. These contracts at their base are programs written by developers. Solidity is one of the most popular languages used to deploy contracts in Ethereum blockchain [26].

4.3.1. Dataset Collection

To obtain sample data for simulation in this problem, multiple smart contracts were taken from repository by smartbugs-wild [27]. There were 47,398 contracts on given repository. Before passing it to preprocessing, it was still necessary to classify them as vulnerable or non-vulnerable. The results of the ICSE 2020 paper [28] were used to obtain classified contracts. This paper compared working of different tools used to analyze smart contracts such as slither, MAIAN, and mythril. Another set of smart contracts were gathered from Google's bigQuery database [29]. These contracts were not classified into vulnerable and non-vulnerable. Hence, slither tool was used to classify some of the contracts and use for simulation. Finally, we were able to acquire 650 non-vulnerable smart contracts and 131 vulnerable smart contracts.

4.3.2. Data Preprocessing

The contracts written in solidity undergo certain steps before being viable for use as data for training and testing. The solidity code is first converted to "bytecode" by a solidity disassembler. This bytecode is now converted to opcodes that were predefined and can be found in the Ethereum yellow paper [30]. The original solidity codes were converted into opcodes, and they can now be used for prediction using DL algorithms. These opcodes are categorical in nature and hence cannot be directly in an algorithm. To use them efficiently for training and prediction, one-hot encoding process is used. This process creates specific frequencies for each opcode possible. On basis of these frequencies, now DL algorithms can be used. These frequencies act as vectors and are passed on to the layer, that is, the Malicious Smart Contract Prediction layer.

4.3.3. Smart Contract Prediction

The Malicious Smart Contract Prediction layer acts as the second layer of security that checks the smart contract for vulnerability. The trained model is able to make a binary classification, which is if the smart contract is vulnerable or safe. In this layer, the smart contracts are classified as vulnerable or non-vulnerable using DL models (e.g., long short-term memory (LSTM), artificial neural network (ANN), and gated recurrent units (GRUs)). The internal working of LSTM is shown as

$$\begin{gathered} fg_t = \sigma\Big(Wh_f \cdot [h^o_{t-1}, z_t] + bs_f\Big) \\ in_t = \sigma(Wh_i \cdot [h^o_{t-1}, z_t] + bs_i) \\ \tilde{C}_t = \tanh(Wh_C \cdot [h^o_{t-1}, z_t] + bs_C) \\ C^l_t = f_t * C_{t-1} + i_t * \tilde{C}_t \\ op_t = \sigma(Wh_o \cdot [h^o_{t-1}, z_t] + bs_o) \\ h^o_t = o_t * \tanh(C^l_t) \end{gathered}$$

The sigmoid function is represented by *sigma*, while the hyperbolic tangent function is represented by tanh. The symbol *in* is input gate, *op* is output gate, and fg is forget gate of the LSTM model. *Wh* stands for weight matrices, while *bs* stands for biases. The candidate state and the new state are denoted by the letters $\tilde{C}$, C^l. h^o is the output, z is the input, and t is the input time.

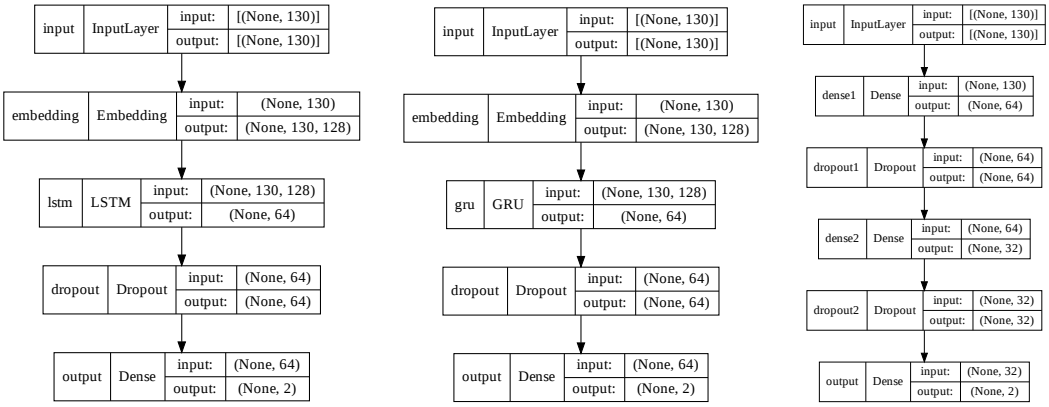
GRU model is an updated form of LSTM having 2 gates rather than 3, namely reset and update gate. The internal calculation formula of the GRU neural network is given below:

$$\begin{aligned} up_t &= \sigma\big(Wh_z \cdot \big[h^o_{t-1}, z_t\big] + bs_u\big) \\ rs_t &= \sigma\big(Wh_r \cdot \big[h^o_{t-1}, z_t\big] + bs_r\big) \\ \tilde{h}_t &= \tanh\big(Wh \cdot \big[rs_t * h^o_{t-1}, z_t\big] + bs_c\big) \\ h^o_t &= (1 - up_t) * h^o_{t-1} + up_t * \tilde{h}_t \end{aligned}$$

Here, up_t denotes the update gate, which chooses which information to delete and which to add fresh. The reset gate, abbreviated as rs_t, controls the degree to which prior data are deleted. σ is the sigmoid activation function as seen above; *Wh* is the weight matrix and h^o_t is the output.

There will be N number of safe users, $\{su_1, su_2, su_3, \ldots, su_N\} \in SU$, having M numbers of smart contacts,$\{s_1, s_2, s_3, \ldots, s_M\} \in U$, to deploy on the blockchain. The generated feature vector after one-hot encode of opcodes obtained from the smart contracts are passed through DL models. The data were randomly split into 70, 15, 15 ratio as train size, test size, and validate data, respectively. The optimal number of epochs for training were found using Keras Earlystopping callback function. The early stopping function is defined as stopping the training when the monitored metric has stopped improving. Then, the training dataset transits through LSTM, GRU, and ANN models as shown in Figure 2. As shown in these figures, we have used embedding layer with input length equal to shape of training dataset followed by spatial dropout layer, which randomly sets input to 0, which overcomes the problem of overfitting. As it is a binary classification of vulnerable or non-vulnerable, we used sigmoid activation function for transformation of output signal. As described in Algorithm 1, it returns set of classified smart contracts as vulnerable or non-vulnerable. Algorithm 2 depicts the whole procedure for the classification of malicious smart contract.

If the model predicts a certain smart contract is vulnerable, it shall be discarded and would not be allowed to proceed further to IPFS hash and into the blockchain. This way, no vulnerable contract shall find its way into the blockchain. An example of prediction is displayed in Figure 3 that shows the probability of a smart contract being malicious using DL algorithms.



(**a**) LSTM. (**b**) GRU. (**c**) ANN.

Figure 2. Sequence model for LSTM, GRU, and ANN.

Algorithm 1 Algorithm for detecting malicious users

Input : U, M
Initialization : **U** $= \{u_1, u_2, u_3, \ldots, u_n\}$ (Set of safe users)
M = Model for classification
Output : SU (Set of safe Users)

procedure MALICIOUSUSERS(U)
  for $u_i \in U$ **do** ▷ users present in U set
    $count = n$ ▷ classifying n times
    $arr =$ empty array
    while $count \neq 0$ **do**
      $Label \leftarrow M.classify(F)$ ▷ classification of smart contract
      $arr.add(Label)$ ▷ appending result into array
      $count \leftarrow count - 1$
    end while
    $CountSafe =$ arr.count(0)
    $CountVulnerable =$ arr.count(1)
    if $CountSafe > CountVulnerable$ **then**
      $SU.add(u_i)$
    end if
  end for
  return SU ▷ data of safe users
end procedure

```
Bytecode :  6080604052348015610010576000080fd5b506040516020806100fa83398101604052516000556c78061003360003960000f300

Opcodes :  00 02 04 05 06 07 08 0B 0C 0E 0F 10 11 12 14 15 17 18 1B 1C 1D 1E 1F 21 22 23 25 26 28 29 2C 2E 2F 31 32

Probability of safe smart contract :
+---------+---------------+
| Model   |   Probability |
+=========+===============+
| LSTM    |   0.00489696  |
+---------+---------------+
| GRU     |   0.000134742 |
+---------+---------------+
| ANN     |   0.400781    |
+---------+---------------+
```

Figure 3. Probability of a smart contract being vulnerable by ANN, GRU, and LSTM.

Algorithm 2 Algorithm for malicious smart contract detection.

Input : U, S, M
Initialization :
U : $\{u_1, u_2, u_3 \ldots u_n\}$ (Set of safe users)
S : $\{s_1, s_2, s_3 \ldots s_n\}$ (Set of smart contract)
M : Model for classification
Output : SC
M : Map containing smart contract and its label

procedure CLASSIFY(U, S) :
  for $u_i \in U$ **do** ▷ users present in U set
    if $\neg MaliciousUsers$ **then** ▷ checks user behavior from MUD Layer
      for $s_j^i \in S^i$ **do**
        $O \leftarrow opcode(s_j^i)$ ▷ generates opcode from its function
        $F \leftarrow encode(O)$ ▷ through one-hot encoding
        $Label \leftarrow M.classify(F)$ ▷ classification of smart contract
        $SC.put(s_j^i, Label)$
      end for
    end if
  end for
  return SC ▷ data of classified smart contract
end procedure

4.4. Application Layer

The final layer is the blockchain layer which obtains a safe smart contract after it has passed through the previous layers and is ready to be hashed by IPFS. Once the hashing process is carried out, it is then deployed on the blockchain.

5. Results and Discussion

In this section, we discuss the evaluation of the proposed system architecture by considering different assessment metrics, such as the classification measures (e.g., training accuracy, training loss, recall score, precision score, and F1 score), receiver operating characteristic (ROC), and blockchain scalability. Further, we also elaborate on the experimentation setup and different tools utilized in implementing the proposed system model.

5.1. Experimental Setup and Tools

The proposed system model is implemented on a system with specifications such as 12 GB RAM, 1 TB HDD, 512 GB SSD, and Intel (R) Core(TM) i7-9750H CPU @ 2.60 GHz processor. The system has a 64-bit Windows 11 operating system. In addition, we also enabled an NVIDIA-based graphics processing unit (GPU), i.e., GeForce GTX 1650 with 4 GB memory, for the AI processing and classification. Moreover, the proposed system model is implemented in an open source software utility, i.e., Jupyter Notebook from Anaconda distribution. In addition, different libraries are used to implement the proposed system model, for example, Pandas is used for the data analysis and manipulation of the data columns in the dataset. Specific Pandas functions, such as $pd.read_csv()$, $df.isna.sum()$, and $df.column.value_counts()$, are used to analyze the columns of the dataset. Further, the Numpy library is used to convert the Pandas dataframe into multidimensional array objects that are helpful in performing faster calculations. Matplotlib and SciPy library are used for visualization, where we illustrate a few graphs on the accuracy, training loss, confusion matrix, and ROC curve for the proposed system model. In addition, this paper utilized TensorFlow and the Keras library to import DL algorithms, such as the LSTM, GRU, and ANN, where we not only used built-in functions but also framed out a few user-defined functions that enhance the performance of the above-mentioned algorithms. Table 2 shows the simulation parameters used while developing the proposed system model.

Table 2. Simulation Parameters.

Hyperparameters	**Values**
Epochs	20
Batch size	256
Dropout	0.6
Activation	sigmoid

5.2. Evaluation Metrics

The AI models can be evaluated using metrics such as the training loss and accuracy and classification measures (i.e., the precision, F1 score, and recall score). In this section, we discuss the performance metrics that are used to evaluate the proposed system model.

5.2.1. Confusion Matrix

A confusion matrix is a table used widely to determine the binary prediction ability of an AI model. It consists of predicted and actual values segmented by positive and negative samples, i.e., true and false positives and true and false negatives. In addition, various other evaluation metrics can be formed using these values, such as the accuracy, precision, recall, and F1 score. In the context of smart contract security, true positive means safe smart contracts according to the ground truth (s_i^j); false positive is where malicious contracts are misclassified as safe (p_i^j); true negative is where an AI model correctly predicts the malicious smart contracts (v_i^j); and false negative is where smart contracts that are safe according to the ground truth are classified as malicious (m_i^j). According to the aforementioned values, the evaluation metrics help in evaluating the performance of the proposed system model.

5.2.2. Accuracy

The accuracy of a prediction model is the fraction of total samples correctly identified by the model. It is the ratio between positive samples ($s_i^j + v_i^j$) and all the samples of the prediction ($s_i^j + v_i^j + p_i^j + m_i^j$).

$$Accuracy = \frac{s_i^j + v_i^j}{s_i^j + v_i^j + p_i^j + m_i^j} \tag{21}$$

5.2.3. Precision

Precision is defined as the fraction of positive samples that are actually positive. It is the ratio between the positive samples correctly predicted by the model (s_i^j) to all the positive samples ($s_i^j + p_i^j$).

$$Precision = \frac{s_i^j}{s_i^j + p_i^j} \tag{22}$$

5.2.4. Recall

Recall is the fraction of all positive samples correctly predicted as positive by the AI model (s_i^j) to both positive and negative samples ($s_i^j + m_i^j$).

$$Recall = \frac{s_i^j}{s_i^j + m_i^j} \tag{23}$$

5.2.5. F1 Score

The F1 score comes into the picture when both the precision and recall values are similar. It depends on the β value; the higher the value of β, the more significant the importance of precision. The lower value of β signifies a higher recall value. It is the combination of precision and recall, which takes the harmonic mean of both the precision and recall values.

$$F1score = \frac{2 * s_i^j}{2 * s_i^j + p_i^j + m_i^j} \tag{24}$$

5.3. Analysis of User Detection Layer

In this subsection, this paper utilized different AI models to classify malicious and non-malicious users from the IoT environment. AI models, such as decision trees, random forest, logistic regression, and Naive Bayes, were used and compared with each other using the evaluation metrics. Table 3 shows the evaluation metrics of all the AI models in classifying the user, i.e., malicious and non-malicious. Figure 4 shows the testing accuracy of each AI model used for the simulation. It is evident from Table 3 and Figure 4 that the random forest and decision tree show the highest accuracy compared to the other models. Among these, the random forest shows the highest accuracy, i.e., 99.72%, because it narrows down the dataset samples into individual decision trees. Then, it applies the decision-making approaches to each tree. Further, based on the voting criteria, the best decision tree is selected iteratively; this improves the overall accuracy of the random forest model. We use $n = 100$ estimators to fit the dataset samples on various decision trees, yielding a higher accuracy. The random forest classifier not only shows the highest accuracy but also a high value of 99.72% as the precision and recall and F1 score.

Table 3. Evaluation metrics of malicious user detection models.

Model	Accuracy	Precision	Recall	F1 Score
Gaussian NB	56.75	70.06	70.06	49.49
Logistic Regression	94.99	95.11	95.11	94.97
Random Forest Classifier	99.72	99.72	99.72	99.72
Decision Tree	99.57	99.58	99.57	99.57

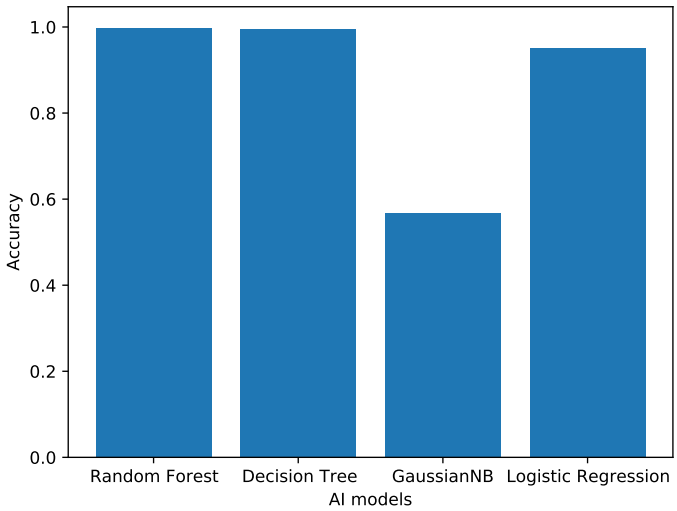


Figure 4. Accuracy graph for AI models.

The experimentation in this paper for the intrusion detection in the IoT network can be compared to the experiments on the same dataset (X-IIoTID) by the authors of [25]. Their

accuracy, precision, and recall for the decision tree classifier are 99.54%. For the results obtained by using a decision tree on the same dataset, we obtain the accuracy, recall, and F1 score as 99.57% and the precision as 99.58%. The data preprocessing performed for our experimentation consisted of important steps, such as removing unnecessary features, filling in missing values, and converting the data types of the features. To fill the missing values, the median of all the values is considered. According to our experimentation, the best classifier is a random forest classifier. A random forest classifier is a culmination of multiple decision trees. By experimenting with different hyperparameters, we observed the highest evaluation results for the random forest with a 99.72% accuracy, recall, precision, and F1 score. On the other hand, Gaussian Naive Bayes that follows the Gaussian distribution has the least accuracy of 56.75% in classifying the malicious and non-malicious IoT user. This happens because it could not find the intuitive patterns inside the feature space of the dataset; it merely relies on the conditional probabilities of each feature. It also gives the least precision and recall values of 70.06% and 49.49% as the F1 score. The major reason for the low performance of the Naive Bayes classifier is that it considers every feature as independent. The logistic regression is a statistical AI model that offers a solution to the binary classification problem using the activation function (e.g., the sigmoid function). It works efficiently when the independent variable (target) is categorical in nature. However, it assumes a "linear" relationship between the dependent and independent variables of the dataset. With the default number of iterations for the logistic regression, which is 100, the line search algorithm does not reach convergence. That is why we set the max iterations as 500 to tackle this, which increased the accuracy significantly to 94.99%, as shown in Table 3.

In addition, decision trees are used to build a small subset from the dataset. Then, each subset is trained using effective decision rules emanating from the features. It is similar to the random forest; however, the random forest uses numerous decision trees to improve its accuracy. Here, the decision tree uses splitting criteria to split the entire tree from root to leaf, where the testing attribute is evaluated at each split node.

It is clear from Table 3 that the random forest has the highest accuracy. However, accuracy cannot be considered the only evaluation metric to compare; it is better to validate the accuracy result with another evaluation metric. Hence, to depict the model's overall performance, we consider the ROC curve, which is a performance statistic for classification at various threshold levels. It works based on the area under the curve (AUC) that indicates the degree or measure of separability. It indicates how well the model can discriminate between classes. The better the model predicts 0 courses as 0 and 1 class as 1, the higher the AUC [31]. The ROC curve is shown in Figure 5, which depicts the AUC for all four models.

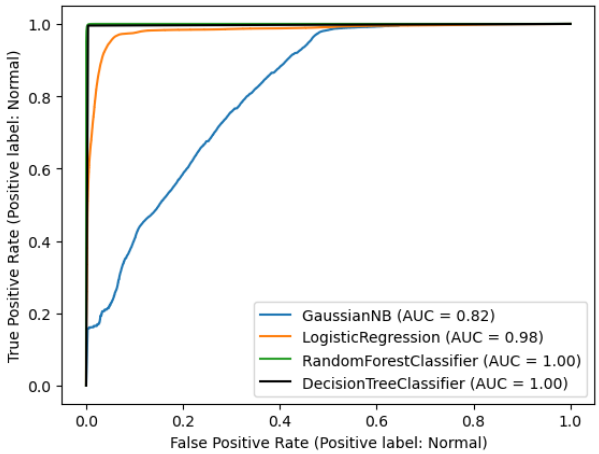


Figure 5. ROC curve for AI models.

5.4. Analysis of Malicious Smart Contract Detection

In this subsection, we evaluate the performance of the AI-based smart contract detection layer using the training and loss curves. First, to predict whether a smart contract is malicious or not, we trained AI-based DL models, such as the ANN, GRU, and LSTM. Then, we compared these models based on various performance evaluation metrics, such as the training accuracy, loss, and ROC curve. Furthermore, the confusion matrix is generated by applying the above-mentioned AI models to the standard curated dataset, i.e., malicious and non-malicious smart contracts. Figure 6 shows the confusion matrix of all the AI models.

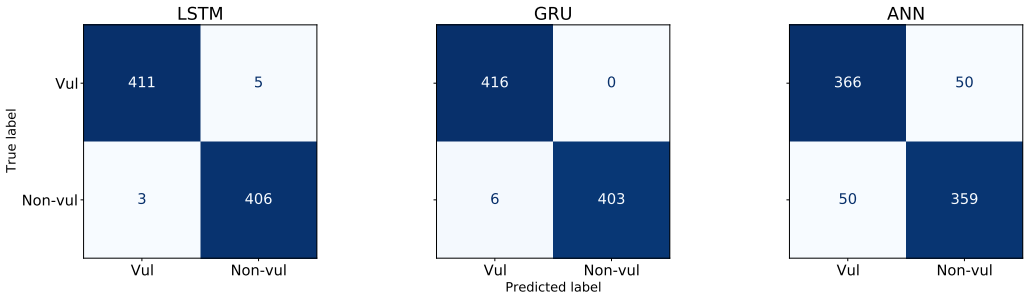
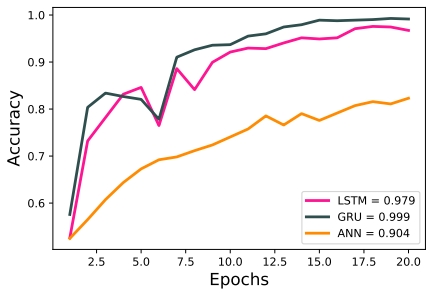


Figure 6. Confusion Matrix.

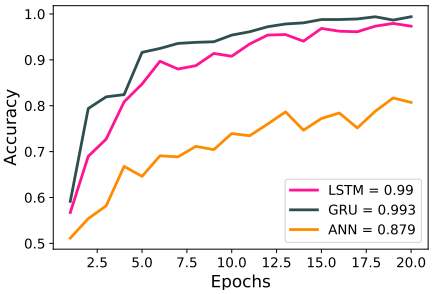
Figure 7 shows the training curve of all the AI models (LSTM, GRU, and ANN), compared at different epochs (training iterations) with the different optimizers, such as Adam, Nadam, and RMSProp. These optimizers help in optimizing the weights and reducing the training loss. From the graph (Figure 7), it is clear that the GRU has a better training accuracy, i.e., 0.99 at the 20th epoch, compared to the LSTM and ANN, which have accuracies of 0.979 and 0.904, respectively. This is because the GRU has better competency than the LSTM model as it resolves the vanishing gradient problem and is a faster, less resource-constrained algorithm than the LSTM. A vanishing gradient problem is where a change in the weights of the neurons in a network is orders of magnitudes lower so that it effectively shows no change. The GRU consists of two gates, i.e., the update and reset gates, that help the algorithm retain information from further back. Contrarily, the LSTM requires more resources to train and overfit to random weights initialization.

Figure 8 shows the training–loss curve between different AI models and optimizers at different epochs. The loss curve depicts the change in the deviation of the predicted values from the model and the ground truth presented. A model should have a decreasing loss curve which eventually tends to zero at each successive iteration. The same is corroborated by the graphs presented in Figure 8. The GRU model has a small loss with all the optimizers because it has a better training accuracy and is more effective than the other AI models. To validate the result of the smart contract detection layer, we also utilized the ROC curve that measures the classification at different thresholds. From Figure 9, it is clear that the GRU and LSTM have better results than the ANN because of their ability to retain old information.

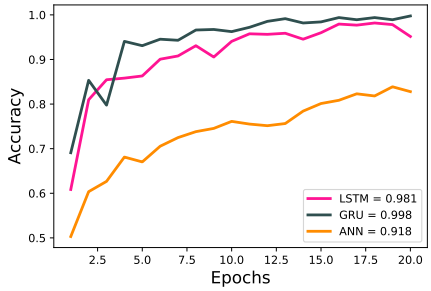
Gupta et al. [3] have similarly performed experiments for a malicious smart contract analysis. The results presented in this paper supersede the results presented in [3]. This can be confirmed by comparing the ROC curve in this paper, Figure 9, and the ROC curve in their paper, Figure 10. From Figure 10, the AUC obtained for the LSTM, GRU, and ANN is 0.93662, 0.93551, and 0.93440, respectively. Similarly, from Figure 9, the AUC obtained for the LSTM, GRU, and ANN is 0.99, 0.99 and 0.97, respectively, which are significantly greater than the existing works.



(**a**) Adam Optimizer

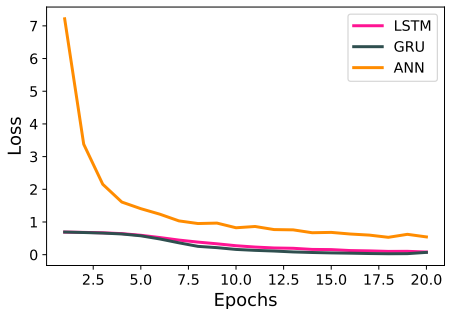


(**b**) Nadam Optimizer

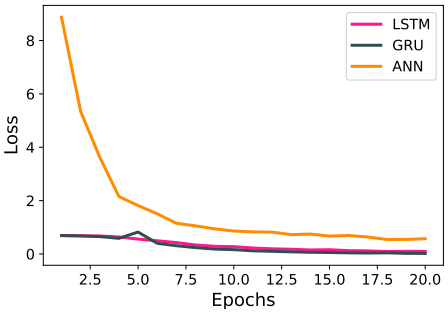


(**c**) RMS-prop Optimizer

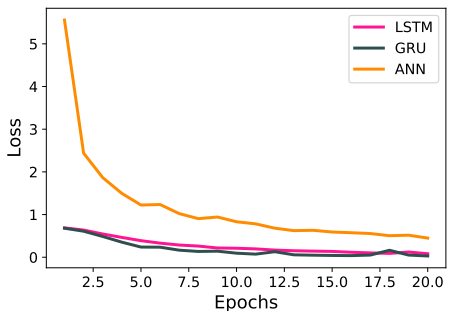
Figure 7. Comparison of LSTM, GRU, ANN training accuracy with different optimizers.



(**a**) Adam Optimizer



(**b**) Nadam Optimizer



(**c**) RMS-prop Optimizer

Figure 8. Comparison of LSTM, GRU, ANN training loss with different optimizers.

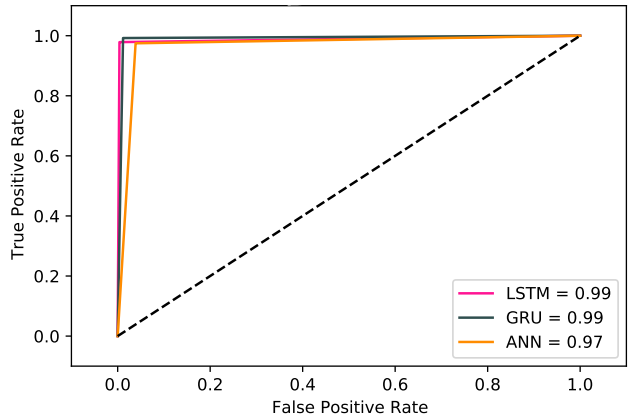


Figure 9. ROC curve for the proposed work.

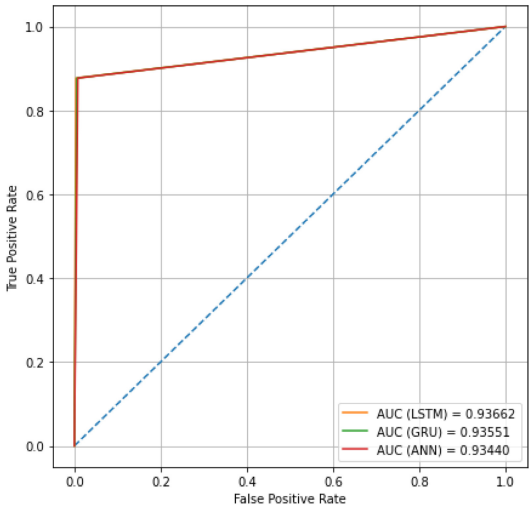


Figure 10. ROC curve for [3].

6. Conclusions

This paper proposed an AI-based intrusion and malicious smart contract detection system model for the IoT ecosystem. First, AI-based algorithms, such as the random forest, decision tree, logistic regression, MLP, and Bayesian model, were considered to efficiently detect malicious users (intruders) from the IoT ecosystem. Here, malicious users are discarded from the IoT environment; only non-malicious users are allowed to forward their data to the intended recipient. To protect the data from security threats, it is stored inside the blockchain immutable ledger where it first authenticates via smart contracts. Further, we strengthened the security of the smart contracts by utilizing DL algorithms, such as the LSTM, GRU, and ANN, that bifurcate malicious and non-malicious smart contracts. Only non-malicious smart contracts are allowed to authenticate the IoT data. The proposed system model efficiently detects the intruders in the IoT system with an accuracy of 99.72% and predicts the smart contracts, i.e., malicious or non-malicious, with an accuracy of 99.27%.

In the future, we will improve the security performance of the AI model by analyzing it with adversarial attacks on the IoT system.

Author Contributions: Conceptualization: S.T., H.S., R.G., N.K.J. and D.S.; writing—original draft preparation: H.S., D.S., R.G. and N.K.J.; methodology: H.S., D.S., S.T., A.T. and M.S.R.; writing—review and editing: S.T., O.A., A.T., M.S.R. and V.M.; investigation: S.T., O.A., M.S.R. and V.M.; Supervision: S.T., O.A., A.T., V.M., M.S.R. and R.G.; visualization; D.S., O.A., N.K.J. and A.T.; software; R.G., H.S., N.K.J. and V.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Researchers Supporting Project number (RSPD2023R681), King Saud University, Riyadh, Saudi Arabia. Partilally support by the the Ministry of Investments and European Projects through the Human Capital Sectoral Operational Program 2014-2020, Contract no. 62461/03.06.2022, SMIS code 153735. This work is supported by Ministry of Research, Innovation, Digitization from Romania by the National Plan of R & D, Project PN 19 11, Subprogram 1.1. Institutional performance-Projects to finance excellence in RDI, Contract No. 19PFE/30.12.2021 and a grant of the National Center for Hydrogen and Fuel Cells (CNHPC)—Installations and Special Objectives of National Interest (IOSIN). This paper was partially supported by UEFISCDI Romania and MCI through BEIA projects AutoDecS, SOLID-B5G, T4ME2, DISAVIT, PIMEO-AI, AISTOR, MULTI-AI, ADRIATIC, Hydro3D, PREVENTION, DAFCC, EREMI, ADCATER, MUSEION, FinSESCo, iPREMAS, IPSUS, U-GARDEN, CREATE and by European Union's Horizon Europe research and innovation program under grant agreements No. 101073879 (FLEXI-cross).

Data Availability Statement: No data are associated with this research work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kumar, P.; Kumar, R.; Gupta, G.P.; Tripathi, R. A Distributed framework for detecting DDoS attacks in smart contract-based Blockchain-IoT Systems by leveraging Fog computing. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4112. [CrossRef]
2. Ndiaye, M.; Konate, P.K. Cryptocurrency Crime: Behaviors of Malicious Smart Contracts in Blockchain. In Proceedings of the 2021 International Symposium on Networks, Computers and Communications (ISNCC), Dubai, United Arab Emirates, 31 October–2 November 2021; pp. 1–8. [CrossRef]
3. Gupta, R.; Patel, M.M.; Shukla, A.; Tanwar, S. Deep learning-based malicious smart contract detection scheme for internet of things environment. *Comput. Electr. Eng.* **2022**, *97*, 107583. [CrossRef]
4. Gupta, R.; Kumari, A.; Tanwar, S. Fusion of blockchain and artificial intelligence for secure drone networking underlying 5G communications. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4176. [CrossRef]
5. Meng, W.; Wang, J.; Wang, X.; Liu, J.; Yu, Z.; Li, J.; Zhao, Y.; Chow, S.S. Position paper on blockchain technology: Smart contract and applications. In *International Conference on Network and System Security*; Springer: Cham, Switzerland, 2018; pp. 474–483.
6. Da Silva Rodrigues, C.K.; Rocha, V. Towards Blockchain for Suitable Efficiency and Data Integrity of IoT Ecosystem Transactions. *IEEE Lat. Am. Trans.* **2021**, *19*, 1199–1206. [CrossRef]
7. Zhou, J.; Feng, G.; Wang, Y. Optimal Deployment Mechanism of Blockchain in Resource-Constrained IoT Systems. *IEEE Internet Things J.* **2022**, *9*, 8168–8177. [CrossRef]
8. Ren, J.; Li, J.; Liu, H.; Qin, T. Task offloading strategy with emergency handling and blockchain security in SDN-empowered and fog-assisted healthcare IoT. *Tsinghua Sci. Technol.* **2022**, *27*, 760–776. [CrossRef]
9. Bataineh, M.R.; Mardini, W.; Khamayseh, Y.M.; Yassein, M.M.B. Novel and Secure Blockchain Framework for Health Applications in IoT. *IEEE Access* **2022**, *10*, 14914–14926. [CrossRef]
10. Singh, R.; Tanwar, S.; Sharma, T.P. Utilization of blockchain for mitigating the distributed denial of service attacks. *Secur. Priv.* **2020**, *3*, e96, [CrossRef]
11. Banerjee, S.; Odelu, V.; Das, A.K.; Chattopadhyay, S.; Kumar, N.; Park, Y.; Tanwar, S. Design of an Anonymity-Preserving Group Formation Based Authentication Protocol in Global Mobility Networks. *IEEE Access* **2018**, *6*, 20673–20693. [CrossRef]
12. Patel, K.; Mehta, D.; Mistry, C.; Gupta, R.; Tanwar, S.; Kumar, N.; Alazab, M. Facial Sentiment Analysis Using AI Techniques: State-of-the-Art, Taxonomies, and Challenges. *IEEE Access* **2020**, *8*, 90495–90519. [CrossRef]
13. Qian, P.; Liu, Z.; He, Q.; Zimmermann, R.; Wang, X. Towards Automated Reentrancy Detection for Smart Contracts Based on Sequential Models. *IEEE Access* **2020**, *8*, 19685–19695. [CrossRef]
14. Al-Otaibi, Y.D. K-nearest neighbour-based smart contract for internet of medical things security using blockchain. *Comput. Electr. Eng.* **2022**, *101*, 108129. [CrossRef]
15. Xiao, W.; Liu, C.; Wang, H.; Zhou, M.; Hossain, M.S.; Alrashoud, M.; Muhammad, G. Blockchain for Secure-GaS: Blockchain-Powered Secure Natural Gas IoT System with AI-Enabled Gas Prediction and Transaction in Smart City. *IEEE Internet Things J.* **2021**, *8*, 6305–6312. [CrossRef]
16. Das, A.K.; Bera, B.; Saha, S.; Kumar, N.; You, I.; Chao, H.C. AI-Envisioned Blockchain-Enabled Signature-Based Key Management Scheme for Industrial Cyber–Physical Systems. *IEEE Internet Things J.* **2022**, *9*, 6374–6388. [CrossRef]
17. Ismail, S.; Dawoud, D.; Reza, H. Towards A Lightweight Identity Management and Secure Authentication for IoT Using Blockchain. In Proceedings of the 2022 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, 6–9 June 2022; pp. 077–083. [CrossRef]
18. Li, T.; Fang, Y.; Jian, Z.; Xie, X.; Lu, Y.; Wang, G. ATOM: Architectural Support and Optimization Mechanism for Smart Contract Fast Update and Execution in Blockchain-Based IoT. *IEEE Internet Things J.* **2022**, *9*, 7959–7971. [CrossRef]
19. Tikhomirov, S.; Voskresenskaya, E.; Ivanitskiy, I.; Takhaviev, R.; Marchenko, E.; Alexandrov, Y. Smartcheck: Static analysis of ethereum smart contracts. In Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain, Gothenburg, Sweden, 27 May–3 June 2018; pp. 9–16.

20. Liu, C.; Liu, H.; Cao, Z.; Chen, Z.; Chen, B.; Roscoe, B. ReGuard: Finding Reentrancy Bugs in Smart Contracts. In Proceedings of the 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion), Gothenburg, Sweden, 27 May–3 June 2018; pp. 65–68.
21. Nikolić, I.; Kolluri, A.; Sergey, I.; Saxena, P.; Hobor, A. Finding the greedy, prodigal, and suicidal contracts at scale. In Proceedings of the 34th Annual Computer Security Applications Conference, San Juan, PR, USA, 3–7 December 2018; pp. 653–663.
22. A lightweight approach to smart contracts supporting safety, security, and privacy. *J. Log. Algebr. Methods Program.* **2022**, *127*, 100772. [CrossRef]
23. Liao, J.W.; Tsai, T.T.; He, C.K.; Tien, C.W. Soliaudit: Smart contract vulnerability assessment based on machine learning and fuzz testing. In Proceedings of the 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), Granada, Spain, 22–25 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 458–465.
24. Alkadi, O.; Moustafa, N.; Turnbull, B.; Choo, K.K.R. A deep blockchain framework-enabled collaborative intrusion detection for protecting IoT and cloud networks. *IEEE Internet Things J.* **2020**, *8*, 9463–9472. [CrossRef]
25. Al-Hawawreh, M.; Sitnikova, E.; Aboutorab, N. X-IIoTID: A Connectivity- and Device-agnostic Intrusion Dataset for Industrial Internet of Things. *IEEE Internet Things J.* **2021**, *9*, 3962–3977. [CrossRef]
26. Gupta, R.; Tanwar, S.; Kumar, N. B-IoMV: Blockchain-based onion routing protocol for D2D communication in an IoMV environment beyond 5G. *Veh. Commun.* **2022**, *33*, 100401. [CrossRef]
27. Ferreira, J.F. SmartBugs Wild Dataset. Available online: https://github.com/smartbugs/smartbugs-wild (accessed on 10 November 2020).
28. Durieux, T.; Ferreira, J.a.F.; Abreu, R.; Cruz, P. Empirical Review of Automated Analysis Tools on 47,587 Ethereum Smart Contracts. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, Seoul, Republic of Korea, 27 June–19 July 2020; Association for Computing Machinery: New York, NY, USA, 2020; ICSE '20, pp. 530–541. [CrossRef]
29. Allen Day, E.M. Ethereum in BigQuery: A Public Dataset for Smart Contract Analytics. Available online: https://cloud.google.com/blog/products/data-analytics/ethereum-bigquery-public-dataset-smart-contract-analytics (accessed on 10 October 2018).
30. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
31. Narkhede, S. Understanding AUC-ROC Curve. Available online: https://towardsdatascience.com/understandingaucroccurve68b2303cc9c5 (accessed on 14 March 2018).