Università degli Studi di Camerino

School of Advanced Studies

Doctoral course in

Blockchain and Distributed Ledger Technology

Curriculum

Industry 4.0

Cycle XXXVIII

Bytecode-based Image Analysis for Security Vulnerability Detection in Ethereum Smart Contracts

PhD Student

Muhammad Usman Tahir

Supervisor

Prof. Antonella Guzzo

Coordinator of the PhD Programme

Prof. Flavio Corradini

Summary

This paper underscores the vital role of blockchain technology in Industry 4.0, aiming to inspire researchers and industry professionals to recognize its transformative potential in creating decentralized, automated, and data-driven industrial settings. It explains core concepts, components, and varieties of blockchain systems, and assesses their uses across various sectors. The research delves into security and privacy issues, particularly relating to Ethereum platforms and smart contracts. It meticulously details common vulnerabilities of smart contracts and their implications for industrial systems. A critical comparison of existing vulnerability-detection techniques reveals current limitations. The paper also investigates the potential of artificial intelligence to enhance security analysis in blockchain contexts, systematically reviewing machine learning and deep learning strategies for identifying smart contract issues. A novel detection framework is introduced and tested against real-world datasets, showing improved accuracy and robustness compared to traditional methods. Ultimately, the paper aims to foster the development of secure and trustworthy blockchain infrastructure for applications in Industry 4.0.

Keywords (EN)

Blockchain Security; Ethereum; Smart Contract; Vulnerability detection

Scientific field of the dissertation (SSD):

Blockchain and Distributed Ledger Technology

School the Ph.D. Student belongs to at UNICAM:

School of Advanced Studies

University where the Ph.D. students carried out his/her program:

University of Calabria

Table of Contents

1 Introduction

In the context of Industry 4.0, understanding blockchain technology and its implications is crucial for effective implementation where blockchain technology plays a critical role in enabling decentralized, automated, and data-driven industrial ecosystems. As blockchain offers significant advantages, particularly in the realm of financial transactions and becomes increasingly integrated into smart manufacturing, IoT networks, and digital supply chains, where it can enhance trust and facilitate controlled supply transactions thereby ensuring its security and reliability is of paramount importance.Additionally, blockchain's capabilities extend to identifying products and their assembly processes, linking it to multiple areas within Industry 4.0.

For successful integration of blockchain, organizations must undergo a structural transformation to maintain user support. As one of the most influential technological advancements, blockchain has seen remarkable development in recent years, especially within the manufacturing sector. It is often associated with intelligent industries [83] and is characterized as a decentralized, encrypted, distributed ledger system that enables the creation of tamper-proof, real-time logs across a network of computers. Overall, blockchain stands out as a transformative force within various industries, facilitating a range of applications essential for the evolution of modern manufacturing.

In this study, we focus on securing blockchain-based systems through the use of smart contracts, which automate transactions and enforce predefined rules without human intervention. By addressing security challenges such as vulnerabilities, malicious behavior, and unauthorized access, this work aims to strengthen the trustworthiness and resilience of blockchain infrastructure in modern industrial environments by highlighting the potential benefits and limitations of this innovative combination of technology and information management.

Ethereum smart contracts are increasingly integrated into modern supply chain systems,

shifting from centralized to decentralized, automated processes. However, vulnerabilities such as logical flaws and improper input handling pose significant risks to supply chain security and often elude traditional testing methods.

Reentrancy is a type of attack in smart contracts that exploits vulnerabilities through repeated function calls, potentially resulting in funds being stolen. To address this, various analysis methods have emerged, particularly involving AI and deep learning. However, they often suffer from high false-positive and false-negative rates, complex code dependencies, and the need for source code access. [198] introduces a novel lightweight method that converts smart contracts into RGB images based on opcodes and trains a VGG16 CNN model to classify them as "Vulnerable" or "Not Vulnerable." The application of image augmentation techniques to address class imbalance ensures the robustness of the model, achieving an accuracy of 99.07% on a publicly available dataset, underscoring the method's reliability. Further [65] enhances the previous approach with a deep learning-based methodology that converts smart contract bytecode into image representations for effective classification of vulnerability patterns, rather than reentrancy only. Unlike prior methods, this framework operates without source code and overcomes limitations of dynamic analysis, proving versatile across systems. Experimental results from a curated dataset indicate a robust performance with 92.24% accuracy and an 89.06% F1-score. The framework also emphasizes reproducibility and adaptability, thereby enhancing the resilience of blockchain smart contracts.

Deep Learning (DL) and Machine Learning (ML) techniques have improved detection, but they suffer from high false-positive rates and limited interpretability. A proposed interpretable One-dimensional Convolutional Neural Network (1D CNN) utilizes integrated Gradients within an Explainable AI (XAI) framework to mitigate these issues by interpreting smart contract opcodes as RGB-encoded sequences. This approach improves transparency in smart contract analysis, achieving 97% accuracy for reentrancy alone, making it more comprehensible for researchers and AI developers.

Additionally, the framework's development intends to facilitate multi-vulnerability detection with XAI, which is crucial for blockchain technology seeking robust smart contract security.

This paper is organized to proide a comprehensive and technical exploration of blockchain technology, its security aspects, and the role of techniques in smart contract vulnerability detection. Section 2 introduces the fundamental concept of blockchain, detailing its core components and architectural framwork. Section 3 categorizes different types of blockchains and examines their applications across various industries. Section 4 focuses on blockchain security and privacy mechanisms, highlighting key challenges and protection strategies. In section 5, Ethereum is discussed in detail, including its architecture and operational principles, which serve as a foundation for the discussion of smart contract in Section 6. Section 7 investigates common vulnerabilities and their impacts on smart contracts, while section 8 presents and compares existing vulnerability detection techniques. Section 9 explores the integration of Artificial Intelligence in smart contract security analysis. Section 10 and 11 further extend this discussion by reviewing Machine Learning and Deep Learning approaches, respectively, along with a comparative analysis of existing studies. Section 12 presents our proposed frameworks for smart contract vulnerability detection, including experimental setup, performance evaluation, and results. Finally, section 13 concludes the paper by summarizing key findings and outlining future research directions.

2 Understanding what is Blockchain?

A blockchain is defined as "a distributed database that maintains an ever-growing number of ordered records known as blocks." These blocks are connected via encryption. Each block includes a cryptographic hash of the preceding block, a timestamp, and transaction data [155, 57, 100, 129]. A blockchain is a decentralized, distributed, and public digital ledger used to record transactions across multiple computers, such that

the record cannot be changed retroactively without affecting all subsequent blocks and requiring network consensus.

History

Blockchain, established by Satoshi Nakamoto in 2008, is primarily used to record and store cryptocurrency transactions, such as those involving Bitcoin. However, proponents are creating and testing new applications for blockchain technology, including payment processing, supply chain monitoring, digital IDs, data sharing, copyright and royalty protection, Internet of Things network management, and healthcare [178].

What is decentralization?

Decentralization in blockchain refers to the transfer of power and decision-making from a centralized body to a distributed network, thereby reducing trust and discouraging authority-seeking. Decentralization [230, 39, 74] is not a novel concept but rather a sliding scale that should be extended to all areas of blockchain applications. Decentralizing management and resource access can result in better and more equitable service. While there may be certain sacrifices, such as decreased transaction speed, the better stability and service levels brought about by decentralization make the compromises worthwhile. Decentralized networks are often utilized in blockchain technology, but their use should be expanded to include all areas in order to provide better and more equitable service.

Data Storage Management in Blockchain Technology

Blockchain technology presents a novel approach to data storage and management. Unlike standard databases, it employs cryptographic procedures to uniquely and securely link blocks. Although blockchain may hold a wide range of data, its primary application has been as a record of transactions. It was initially designed to support Bitcoin, but it now has numerous applications beyond cryptocurrency [158].

Blockchain data storage and security offer a dramatic paradigm shift in how information is stored, accessed, and safeguarded. Blockchain technology is fundamentally a decentralized and tamper-resistant ledger that gives a transparent and secure means

of recording data. Unlike centralized databases, which are prone to severe security issues due to a single point of failure, blockchain distributes data throughout a network of nodes, making it resistant to unauthorized alterations or hacking attempts. This decentralized method improves data integrity while simultaneously opening up new opportunities for privacy, transparency, and user control. In this dynamic context, the combination of blockchain and data storage solutions is reinventing how organizations manage sensitive information, opening the door to new applications and reshaping the cybersecurity landscape [158].

When it comes to digital currencies like Bitcoin, distributed ledger technology, specifically blockchain, eliminates the need for a central authority or control. Once data is placed on the ledger, it becomes irreversible, a crucial property of decentralized blockchains that demonstrates immutability.

2.1 Blockchain components

Blocks

Blocks are data units that contain transaction information on a blockchain. Every block in the blockchain contains a timestamp, a transaction history, and a link to the previous block.

Chains

The chain, also known as the distributed ledger, is the backbone of the blockchain. In blockchain technology, a chain is a linked sequence of blocks that store data. Each block is linked to the one before it, and this connection is what lends blockchain its "chain" character.

Blockchain may be visualized as a massive book, with each page representing a block. Each block contains data, which can range from financial transactions (such as Bitcoin) to medical information or even votes in elections. The blocks are then connected in chronological sequence, resulting in a tamper-proof record [223].

Blockchain is made up of temporally linked interconnected blocks that unite to form an unbreakable, continuous chain. Each block has a unique cryptographic hash based on the previous block's contents and hash. The blockchain approach uses this type of technology to verify its data and transactions. Instead, they are immutable, meaning that even if changes are made to the block, as indicated by changes in its hash, the preceding blockchain cannot be altered.

Blockchain is used for supply chain management, voting, identity verification, smart contracts, and decentralized funding, in addition to cryptocurrencies. It disrupts commercial companies by developing safe, transparent networks that eliminate intermediaries. However, it is critical to consider alternative approaches and carefully plan implementation based on the specific use case.

What are the components featured in Blockchain Technology?

A blockchain is comprised of the three essential components listed below:

Block

A blockchain is a secure, sequential data collection in which each block contains several transactions and a unique hash of the block before it. These blocks cannot be modified or removed since they are immutable [223]. Data tampering is difficult because changes made to one block will not affect the hash of the next. Because blocks are dispersed over a network of nodes, no single entity can control the bulk of the blockchain. As a result, data storage is both transparent and secure.

- The data composition of a block may vary depending on the kind of blockchain and its intended use. Transactions involving the transfer of digital assets between addresses are often recorded in Bitcoin blocks. Other blockchain apps may contain voting records, health histories, identifying details, supply chain information, and executable code for smart contracts.
- A nonce is a number used in the trial-and-error process of adding blocks to the blockchain during Bitcoin mining. It is hashed and part of the block header. The

block gets added to the blockchain if its hexadecimal value is less than or equal to the network's difficulty target. With each attempt, the nonce is incremented by one, requiring a substantial amount of computational power.

- A block hash is a unique identifying number assigned to each block on the blockchain. It is a cryptographic function that generates a unique string of characters, acting as the block's digital fingerprint, and is used on a fixed-size block of data. Block hashes are created using complex mathematical techniques that take into consideration both the block's data and a nonce value, ensuring that the hash is generated only once.

Each block consists of three items that, when added consecutively to the blockchain, form a continuous and unchangeable log of transactions or data [129].

Miners

Miners are responsible for maintaining the security and integrity of blockchain networks such as Bitcoin. Miners complete the proof-of-work (PoW) algorithm, which is a mathematical problem with a unique hash for each block. Once a suitable solution has been identified, it is broadcast to the network, where more nodes confirm its legality. If the majority of the network accepts that the solution is valid, the block is uploaded to the blockchain, and the miner is compensated with newly created bitcoin [223, 74]. Block incentives, which frequently include newly minted cryptocurrency and transaction fees paid by users, entice miners to join the network. This incentive encourages them to invest in processing capacity and discover legitimate blocks. Decentralized mining makes the network more resilient to censorship and attacks.

Not all blockchains use PoW or mining as their consensus method. Some, like Ethereum, use the proof-of-stake (PoS) mechanism, in which users become validators by guaranteeing a certain amount of Bitcoin as collateral. This update aims to address issues related to PoW mining scalability and energy consumption.

Nodes

A blockchain network consists of nodes, which operate as participants by copying the

whole blockchain and certifying transactions and blocks. These computers or devices are responsible for maintaining a backup of the blockchain ledger, which is updated as new blocks and transactions are added. Nodes work together to distribute new transactions and blocks, with each transaction's digital signature, inputs, and outputs validated for validity. Furthermore, when a new block is received, nodes ensure that it satisfies the blockchain protocol's consensus requirements [129].

3 Types of Blockchain

There are numerous ways to categorize blockchain systems into different types. Fundamentally, based on accessibility, governance, and participation rules, blockchain systems [129, 15] can be categorized into three types: public, private, and consortium.

3.1 Public Blockchain

In permissionless or public blockchain technology, participants are afforded the unique opportunity to join the network without prior authorization [29]. This characteristic embodies the essence of decentralization, as it empowers individuals to participate in the consensus process, facilitate transactions, and maintain the integrity of the shared ledger as shown in Fig 1 [218]. The architecture of public blockchains enables the publication, accessibility, and validation of new blocks by all participants, allowing them to maintain a comprehensive copy of the entire blockchain [10].

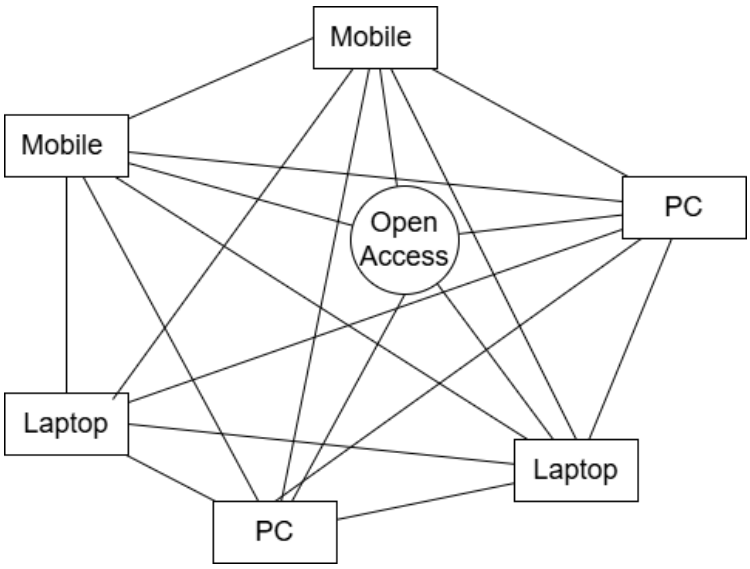

Fig. 1 Public Blockchain

Public blockchains are characterized by their robust security in information and operational processes. While any participant is permitted to join the network and contribute transactions that are subsequently organized into blocks, the integrity of these blocks is ensured through computationally intensive consensus mechanisms. These mechanisms may include complex puzzle-solving tasks or the staking of native cryptocurrencies. The integrity of the block contents is safeguarded through cryptographic hashing and a decentralized consensus framework, which collectively mitigate the risks of tampering. In blockchain technology, a significant advantage is the capacity for numerous nodes to maintain anonymity, thereby safeguarding user privacy [10].

While offering numerous advantages, public blockchains also present a range of unresolved research challenges. One significant issue pertains to the quest for efficiency, which is notably impacted by the extensive participation of users and the computationally intensive consensus mechanisms [29] .

Sustainable Public Procurement (SPP) [157] is based on economic, environmental, and social sustainability, emphasizing the importance of green public procurement (GPP)

for public authorities to mitigate environmental impacts while fostering economic development and social equity. As major consumers, European Union public authorities can utilize their purchasing power to promote sustainability; however, Sustainable Public Procurement (SPP) is not yet widely implemented [11]. The beneficial effects of SPP on national economies include resource efficiency, less CO2 emissions, job development, and innovation. Nonetheless, obstacles such as legislative impediments and inter-departmental collaboration persist.

Conversely, Smart Public Procurement leverages sophisticated technologies, including artificial intelligence, robotic process automation, the Internet of Things (IoT), and blockchain, to optimize procurement procedures. These technologies enable data analysis, automate repetitive operations, and ensure transparency and security in public transactions. The Blockchain Internet of Things (BIoT) concept is evolving as a framework to mitigate inefficiencies and corruption in public procurement by providing a secure and transparent environment for transactions and facilitating smart contracts. Although BIoT can potentially improve procurement efficiency and integrity, its full deployment is still limited, requiring further investigation into its conceptual dimensions and practical applications in the public procurement sector [150].

Recent Advancement. [150] discusses the global challenges encountered in public procurement and highlights the necessity for sustainable and intelligent procurement practices. This study examines the potential of emerging technologies, specifically Blockchain and the Internet of Things (IoT), to drive advancements in this domain. A model for assessing the study is presented, based on six hypotheses, and validated through a survey conducted in Romania that involved key stakeholders. The research employs structural equation modeling (SEM) to demonstrate the beneficial effects of BIoT adoption on achieving sustainable, transparent, and intelligent public procurement while concurrently addressing challenges such as corruption and the need for organizational reform. The objective of the findings is to offer theoretical insights

alongside practical solutions that enhance public procurement processes while promoting innovation and sustainable development.

[200] discusses the global challenges encountered in public procurement and highlights the necessity for sustainable and intelligent procurement practices. This study examines the potential of emerging technologies, specifically Blockchain and the Internet of Things (IoT), to drive advancements in this domain. A model for assessing the study is presented, based on six hypotheses, and validated through a survey conducted in Romania that involved key stakeholders. The research employs structural equation modeling (SEM) to demonstrate the beneficial effects of BIoT adoption on achieving sustainable, transparent, and intelligent public procurement while concurrently addressing challenges such as corruption and the need for organizational reform. The objective of the findings is to offer theoretical insights alongside practical solutions that enhance public procurement processes while promoting innovation and sustainable development.

[163] Cosmos blockchain operates as a public, decentralized network designed to enhance scalability and promote interoperability among independent blockchains. The Tendermint consensus mechanism, along with the Inter-Blockchain Communication (IBC) protocol, enables secure and seamless transactions across multiple platforms. The Atom token is fundamental to the operation, serving a vital function in ensuring network security and governance, which in turn improves cross-chain connections. An in-depth examination of Cosmos architecture reveals its potential to revolutionize blockchain interconnectivity, thereby shaping digital commerce and governance. The research highlights the crucial role of the Cosmos blockchain in developing resilient blockchain ecosystems.

[43] framework as PRBFPT (Practical Redactable Blockchain Framework with Public Trapdoor), aims to enhance public blockchain technology by enabling data editing while preserving transparency and accountability. The chameleon hash employed by PRBFPT facilitates the participation of all nodes in the blockchain in editing operations

via a public trapdoor, thereby removing the necessity for predefined node management. An auditing mechanism is also incorporated to validate the authenticity of the modified data. A contract-based, locked voting scheme is also introduced to enhance the voting process. The assessment of the PRBFPT prototype reveals its capability to incorporate modules with negligible time implications. At the same time, the expenses related to specialized transactions are comparable to those of standard Ethereum transactions. This positions it as a feasible approach for addressing misinformation on public blockchains.

[71] examines the challenges and opportunities associated with electronic voting within the realm of public administration, with a particular focus on Italy, where the use of traditional paper ballots remains prevalent. The security concerns that have hindered the global adoption of e-voting are particularly emphasized, especially in the context of the COVID-19 pandemic, which has intensified the demand for secure remote voting solutions. This study presents a methodology that utilizes the ISO/IEC 15408 certification process for assessing e-voting security, incorporating both legal and technical requirements through BPMN processes. The analysis focuses on a Solidity smart contract deployed on the Ethereum blockchain, pinpointing vulnerabilities and limitations while also developing a BPMN representation to harmonize legal and technical dimensions. The primary objective is to improve the security and transparency of e-voting systems, thereby promoting digital citizenship and the effective use of resources.

3.2 Private Blockchain

Permissioned or private blockchains are expressly designed for use by a single organization, where access to the network is allowed to members only by invitation as shown in fig 2. The members undertake specific duties that enable the decentralized upkeep of the blockchain infrastructure [29]. This structural differentiation distinguishes private blockchains from public ones, as access is limited to approved organizations, ensuring

that only verified individuals can participate in the network and maintain the integrity of the blocks [218] .

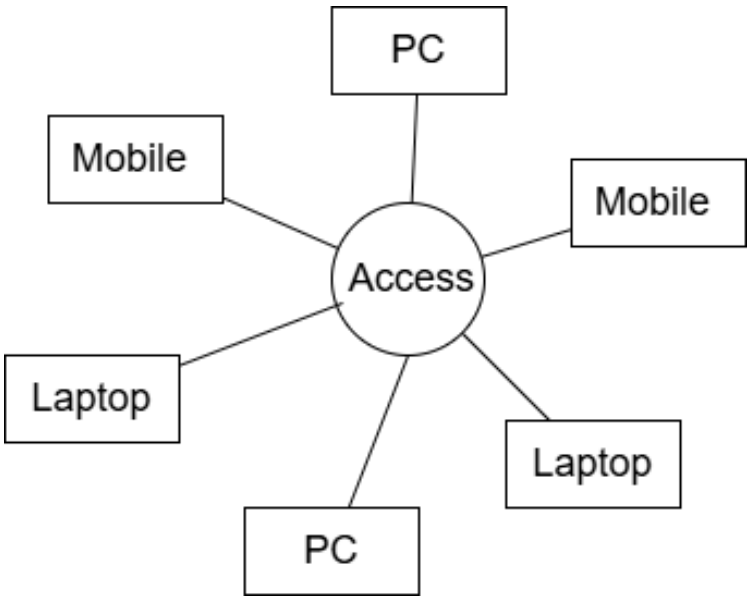

Fig. 2 Private Blockchain

Permissioned blockchains are considered more secure and efficient than public blockchains, as only authorized members have access to the network. Hashes and participant agreements also provide similar safeguards against tampering, similar to those found in public blockchains. Nonetheless, nodes in private blockchains lack anonymity [218]. The unresolved research challenges in private blockchains pertain to block manipulation and network breaches by authorized actors within the network.

3.3 Consortium Blockchain

Consortium blockchains are distinct private blockchains intended to promote cooperation among various groups. These blockchains provide a designated set of organizations to share access and governance of the network, hence augmenting trust and efficiency in their interactions, see Fig 3 . Membership in the network is strictly limited to

invited persons considered trustworthy, guaranteeing a safe and dependable environment for all participants. The consensus process used in these blockchains demonstrates a slower speed than private blockchains, although it functions more rapidly than public blockchains. Consortium blockchains provide superior protection against unauthorized modifications compared to private blockchains in terms of security management. The security against hacking attempts is markedly improved in this specific form of blockchain due to the use of advanced security measures resulting from the participation of individuals from diverse organizations [29, 21].

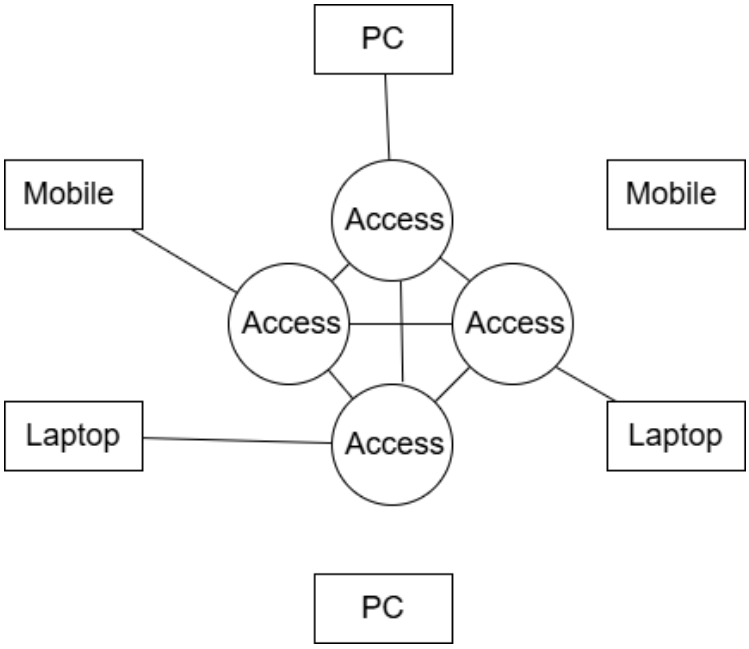

Fig. 3 Consortium Blockchain

Consortium blockchains are often utilized in various domains, including supply chain management, healthcare, and banking, due to their superior efficiency compared to public blockchains. Moreover, they maintain confidence and integrity among the firms participating in the network [27, 15]. Consortium blockchains can reach consensus

via proof-of-work (PoW), proof-of-stake (PoS), proof-of-authority (PoA), or other methods, such as delegated proof-of-stake, among others.

3.4 How does Blockchain technology work?

The key distinction between this technology and others lies in its operation, structure, and method of access. A blockchain is composed of scripts that function as programs, handling database operations such as data input, retrieval, storage, and maintenance. Its distinguishing feature is its decentralized nature, with many copies of the database dispersed across numerous workstations.

The blockchain process adheres to particular procedures [223] established by the leading blockchain network. For example, consider Bitcoin, a prominent cryptocurrency. When you transmit Bitcoin to someone, the transaction is broadcast to the whole network. Special computers, called miners, then compete to validate the transaction, much like solving a complex problem.

Initially, your transaction is placed in a memory pool and waits for a miner or validator to select it. When a transaction is added to a complete block, it is encrypted using an encryption algorithm, therefore initiating the mining process. Miners around the network collaborate to solve the cryptographic hash. Every miner generates a random hash except the nonce (a unique number used only once). Miners begin the process by changing the nonce value to zero and adding it to the randomly generated hash. If the number received does not meet the given hash criteria, the nonce is incremented by one, and a new block hash is generated.

So, whenever a new block of information or data is incorporated into the blockchain, it generates a hash.

The information in the block is hashed, and the hash value is then appended to the end of the chain. Any time the data in the block is updated or changed in any manner, the hash value changes [68, 234].

If a hash value differs from what was initially provided, it is assumed that the data was tampered with. If it matches the original data, it confirms that the information has not been altered or interfered with.

After a block is put to the end of the blockchain, prior blocks cannot be modified. A change in any data affects the hash of the block it was stored in. Because each block carries the preceding block's hash, any modification in one would affect the subsequent blocks. The network would reject a changed block because its hashes did not match.

The blockchain is a collection of information blocks that are validated and chronologically connected using hash values. Each block is like a page in a book, and every page is confirmed with a hash value generated.

3.5 Blockchain Industries

Blockchain technology, a revolutionary innovation, is making significant strides across various industries, transforming traditional processes and enhancing efficiency, security, and transparency. Below is an exploration of the industries where blockchain is being utilized, highlighting its applications, benefits, and challenges.

3.5.1 Financial Sector

Blockchain improves risk management, security, and authenticity in banking and finance. It enables smart contracts, enhances identity verification using blockchain-based IDs, and secures credit reporting, making it a more dependable alternative to conventional ways.

- **Banking and Fund Management:** Blockchain rectifies inefficiencies in fund and account administration, enhancing data integrity and security in financial transactions without dependence on tokens. It has the capacity to revolutionize credit information and payment settlement procedures, especially in areas such as China.

- **Market Clearing and Settlement:** Initiatives such as the ASX CHESS Replacement Project illustrate blockchain's [162] potential to transform post-trade infrastructure, despite acknowledged implementation obstacles.
- **Supply Chain Finance:** Platforms such as BCautoSCF [32] use blockchain technology to improve transparency and efficiency within the automotive retail industry, automating operations and decreasing financing expenses.
- **Crowdfunding:** Blockchain is seen as a mechanism to enhance the effectiveness of crowdfunding for small enterprises, guaranteeing transparency and legality while mitigating issues such as tax fraud and data theft.

The challenges in the adoption of blockchain technology within the financial sector include the complexity of implementation, which has led to significant delays in projects like the ASX CHESS Replacement Project [162] , and the ongoing difficulties related to decentralized system regulation and the establishment of industry standards. Additionally, issues such as tax evasion, data theft, and extortion have emerged in the context of crowdfunding using blockchain, raising concerns about the legal frameworks necessary to support its growth. Furthermore, the need for training and education in blockchain technology is highlighted as essential to enhance its efficiency and effectiveness in financial applications. Although blockchain remains nascent, its capacity to optimize processes and bolster security inside financial industries is gaining recognition.

3.5.2 Healthcare Industries

Blockchain technology is being used in the healthcare industry to tackle substantial security and privacy issues related to the management of sensitive medical information [146]. As healthcare shifts from paper to digital forms, the need for secure data interchange, verification, and connection becomes critical. Blockchain provides decentralized storage, improved security, and authentication, which helps mitigate medical fraud and

foster a patient-centric approach.

Throughout the COVID-19 [183] pandemic, blockchain has shown significant [63] advantages in the safe management of drug supply chains, immunization records, and patient data, hence enhancing monitoring and reaction capabilities for government agencies. It may enhance the safety and security of vaccination records by creating effective data storage infrastructures.

Research underscores several uses of blockchain in healthcare, including electronic health record management, medicine supply chain optimization, and biological research facilitation. Blockchain may improve patient data management by assuring adherence to regulatory standards while allowing access to [146] authorized healthcare professionals. It further facilitates the creation of intelligent apps that provide individualized treatment regimens based on facts rather than conjecture.

Challenges persist, including the need for infrastructure development and integration; nonetheless, blockchain's potential to enhance clinical trials, guarantee transparency, and diminish costs for players in the healthcare ecosystem is substantial. Blockchain technology offers a viable alternative for improving security, privacy, and efficiency within the healthcare sector [182].

3.5.3 Supply Chain

Blockchain technology is progressively used in supply chain sectors to improve efficiency, transparency, and trust among diverse stakeholders. Integrating blockchain with current business information systems enables organizations to securely communicate data and verify its legitimacy, thereby diminishing knowledge asymmetry and preventing corporate malfeasance [122].

In intricate supply chains, such as those in agriculture and aviation, blockchain enhances traceability and protects transactions, thereby reducing the risks of data tampering and fraud. The integration of blockchain with Internet of Things (IoT) [51] technologies, including RFID, enhances data management and security, as shown in the aviation

industry [78], where it facilitates process optimization and communication among stakeholders.

Notwithstanding its promise, the implementation of blockchain in supply chains encounters obstacles in organizational, technological, and policy matters [117]. Existing traceability systems often encounter difficulties in demonstrating provenance and mitigating fraud, challenges that blockchain may resolve via its decentralized architecture and robust data management features.

Blockchain offers a viable alternative to conventional supply chain management by facilitating a synchronized network that improves operational efficiency and promotes novel governance structures. Nevertheless, more study is essential to address the obstacles to its adoption and to investigate its full potential in sustainable supply chains.

3.5.4 Educational Sector

Blockchain technology is progressively being incorporated into the education industry, providing several advantages that improve information management, data security, and confidence in academic procedures [72]. A principal use of blockchain in education is the administration of academic credentials, including degrees and transcripts. Blockchain guarantees the legitimacy of unique digital assets while preserving privacy and trust in the associated procedures.

Numerous projects have arisen to use blockchain for educational objectives. Learning Machine Co. and MIT Media Lab [201] created a blockchain-based system for managing certificates that enables safe viewing, storage, and verification of academic qualifications. Furthermore, services such as Filecoin and SiaCoin [14] aim to reduce cloud storage expenses for institutions by facilitating the exchange of cryptocurrencies for unused storage capacity.

Blockchain enables a more individualized educational experience by providing a unique identification for each learner. This system may retain supplementary data [91] about student performance, location, and project evaluations, thereby improving learning

outcomes and mitigating the possibility of credential tampering.

Smart contracts, functioning on blockchain networks such as Ethereum, are essential for automating many educational activities [159]. They may be used to compensate pupils according to their accomplishments, guaranteeing that disbursements occur without intermediaries. Moreover, smart contracts facilitate equitable assessment in collaborative projects by monitoring individual contributions, therefore mitigating disparities in effort among team members.

Smart contracts enable instructors to submit their teaching plans and gather data on classroom interactions, therefore enhancing feedback and instructional quality [133]. This data may be corroborated by all stakeholders, providing uniformity in educational provision. Educators may also get digital awards for their efforts, promoting high-quality education. Ultimately, blockchain may improve research cooperation between students and supervisors by securely documenting interactions and contributions. This openness promotes active engagement and enhances research results, eventually benefiting society at large. The use of blockchain in education has the capacity to transform the management of academic credentials, the assessment of pupils, and the conduct of educational exchanges.

3.5.5 Agriculture

Blockchain technology is being increasingly applied in agriculture to enhance production, transparency, and efficiency across all aspects of the business. Utilizing blockchain, farmers may enhance supply chain management, guaranteeing stability, security, and fairness in operations. This technology facilitates the monitoring of food sources, therefore enhancing consumer confidence and establishing a reliable food supply chain.

A notable use of blockchain in agriculture is the creation of data-driven agricultural systems that employ smart contracts [173]. These contracts enable swift payments among stakeholders contingent upon alterations documented on the blockchain, therefore improving agricultural yield.

The emergence of platforms such as BIoT seeks [175] to optimize loan disbursement procedures and enhance data quality for lenders, therefore assisting farmers in the successful management of their crops. Blockchain significantly enhances the efficiency and transparency of the food supply chain. It offers a secure, decentralized ledger for tracking food goods, therefore mitigating concerns associated with food safety, fraud, and waste. This technology enhances stakeholder cooperation and streamlines supply chain operations, making it a feasible alternative for the agricultural industry.

Furthermore, blockchain technology is associated with the advancement of renewable energy sources in smart agriculture, facilitating sustainable growth. The benefits include better traceability and transparency throughout food supply chains, resulting in improved peer-to-peer efficiency, consumer satisfaction, and food safety.

Research underscores the promise of blockchain in digital agriculture, emphasizing its applications in food safety [2], quality control, and cost efficiency. By securely tracking agricultural products from production to consumption, blockchain helps mitigate supply chain fraud and improve communication among stakeholders.

The amalgamation of blockchain and IoT [19] in agriculture offers prospects for enhancing supply chain transparency, accountability, and productivity, while also confronting hurdles that must be surmounted for successful deployment. This technology has the capacity to transform the agriculture industry by augmenting food safety, mitigating fraud, and raising the overall efficiency of commodity and transaction monitoring.

3.5.6 IOT

Blockchain technology is increasingly recognized as a crucial solution to the security and privacy challenges posed by the Internet of Things (IoT). The proliferation of IoT devices raises substantial issues about illegal access and data integrity. The interrelated nature of these devices necessitates robust trust mechanisms to ensure that only authorized users can access critical resources. The intrinsic attributes of blockchain, like openness and auditability, make it an optimal choice for augmenting trust in IoT

applications [240].

Blockchain's empowerment of IoT with a decentralized architecture is a game-changer. Traditional centralized IoT [52] systems often grapple with scalability, privacy, and security vulnerabilities due to singular points of failure. By embracing blockchain, IoT can operate on a decentralized framework, reducing operating costs and mitigating security risks. This shift allows for billions of transactions among IoT devices without intermediaries, thereby enhancing efficiency and security.

Research has examined many integration methods between blockchain and IoT, emphasizing blockchain's capacity to establish immutable ledgers and enable safe transactions. This is especially pertinent in contexts like supply chain management, where data integrity and dependability are crucial. The distributed ledger technology of blockchain guarantees that all transactions are recorded immutably, hence substantially improving the security of IoT devices [212].

In the domain of smart homes, several blockchain-based solutions have been suggested to enhance security and privacy. These technologies often obviate the need for conventional mining operations, hence reducing overhead expenses while preserving a secure communication architecture. By designating specific functions to devices in a smart home, such as a miner responsible for monitoring communications, these designs can offer a cost-effective and efficient way to safeguard IoT settings [53].

The concept of Blockchain of Things (BCoT) has emerged as a comprehensive framework that combines the strengths of blockchain with the Internet of Things (IoT). This approach addresses the challenges faced by IoT, such as data privacy and security, while also exploring the potential of blockchain in industrial applications and beyond. The convergence of these technologies opens up new avenues for research and application, particularly in the context of 5G networks [42].

The combination of blockchain and IoT is proving to be a boon for the healthcare sector

[56]. The rise of wearable technology has raised concerns about data privacy and security. However, innovative frameworks using modified blockchain models are stepping in to enhance the confidentiality and anonymity of IoT application data, thereby improving patient care and treatment outcomes. These frameworks often employ advanced cryptography methods to safeguard sensitive data, offering a promising future for healthcare.

Privacy concerns in IoT applications have spurred the exploration of various protective measures, such as anonymization and encryption. These strategies are now recognized as vital for preserving user privacy in blockchain-based IoT systems. The implementation of robust privacy safeguards significantly reduces the risk of data breaches, thereby instilling confidence in the security of IoT applications [75].

Intelligent designs using blockchain have been suggested to augment the security and scalability of IoT systems. These designs prioritize the enhancement of accuracy, centralization, and latency for extensive data analytics, all while maintaining decentralized security. Nevertheless, issues of latency and processing capacity must be resolved to actualize the promise of these systems fully [191].

The development of multi-layered blockchain frameworks for IoT has gained traction. These frameworks typically consist of layers that oversee various aspects of IoT operations, including data processing, secure ledger management, and user interfaces [70]. By leveraging platforms like Ethereum, these systems can execute fast and secure transactions, thereby enhancing the overall utility of IoT applications.

While the use of blockchain technology in IoT applications is expected to significantly enhance security and privacy, researchers must address several challenges and limitations before it can be widely accepted. Overcoming these obstacles will be crucial for the successful deployment of blockchain in IoT, ultimately leading to improved security solutions for interconnected devices. Ongoing research and development in this field

are poised to unveil new opportunities for enhancing the safety and efficiency of IoT systems.

3.5.7 Cryptocurrency

Cryptocurrencies are digital assets that serve as mediums of exchange, using cryptography to secure transactions. Recognizing their role can help you understand a rapidly growing part of the financial world. Bitcoin, created in 2009 by the pseudonymous Satoshi Nakamoto, is the most recognized cryptocurrency, initially gaining traction on the dark web but increasingly used in mainstream commerce. Its uniqueness is secured through blockchain technology, which records transactions transparently and prevents double-spending by maintaining a distributed ledger. The creation of new bitcoins via mining involves updating this ledger, with a cap of 21 million bitcoins; over 17.5 million have been mined so far. While bitcoin remains dominant, the cryptocurrency landscape is expanding, potentially leading to new contenders in the future [205].
Particularly in industries such as FinTech and E-commerce, blockchain technology has emerged as a revolutionary solution to the serious security and privacy problems that afflict the Internet. Even if the Internet is an essential tool for contemporary digital life, its inherent shortcomings—particularly regarding user privacy- highlight the need for robust alternatives. Peer-to-peer (P2P) transactions are enabled by blockchain technology, which underpins cryptocurrencies and eliminates the need for intermediaries such as conventional banks [116]. It carefully verifies transactions and keeps an unchangeable, permanent record, protecting user identities and private data. A cooperative framework that aggregates all transactions into a coded digital ledger provides this protection.
The evolution of Bitcoin [222] from 2010 to 2019 highlights significant milestones in its market and technology. Initially, in 2010, bitcoin had no trade history as it was only mined, leading to a market cap exceeding $1 million in November [195] due to early mining pools. The following years saw the emergence of altcoins and cryptocurrency exchanges, which introduced new features like increased anonymity, sparking curiosity

about the expanding crypto landscape. Between 2014 and 2016, the rise of platforms like Mt. Gox and BitPay, along with new cryptocurrencies such as Ethereum, led to bitcoin's market value reaching $14 billion and to its recognition as a real money currency. However, 2017 marked a peak, followed by a dramatic 80% drop in 2018, with bitcoin values hitting $4,000 [22]. Despite the decline, investor interest persisted, leading to new regulatory frameworks. By 2019, bitcoin remained relevant, attracting new investors despite its lower value. The underlying technology, blockchain, was recognized for its broader applications beyond bitcoin, including e-commerce and banking, though bitcoin's utility as a digital asset remained limited.

3.5.8 IOV

The Internet of Vehicles (IoV) [23] represents a transformative transformation in the transportation sector, propelled by the increasing need for vehicular data and advancements in communication technology. The integration of Smart Vehicle-to-Everything (V2X) [189] connection enhances the efficiency and safety of roadways in smart cities by facilitating communication between vehicles and diverse networks inside the Internet of Vehicles (IoV). The link is vital for improving traffic management, emergency response, and congestion forecasts, making the Internet of Vehicles (IoV) a crucial element of modern Intelligent Transportation Systems (ITS).

The integration of the Internet of Vehicles (IoV) raises significant security concerns, particularly the potential for malicious attacks that could compromise vehicle connectivity. Blockchain technology emerges as a potent solution, ensuring the security of V2X [188] transmission. This is crucial, as the rapid and accurate dissemination of critical event data, such as accident reports, is essential for the safe operation of motor vehicles. By establishing robust security protocols within vehicular ad hoc networks (VANETs), blockchain technology prevents unauthorized alterations of critical communications, addressing these security concerns head-on.

Blockchain technology emerges as a powerful tool in enhancing the reliability and security of the Internet of Vehicles (IoV). With its decentralized and transparent platform, blockchain can gather historical data on traffic and accidents, ensuring that vehicles requesting this information receive precise and reliable data. The intrinsic characteristics of blockchain, namely immutability and public accessibility, enhance the integrity of vehicle communications. These attributes make it impossible for malicious entities to alter the data stored on the blockchain, thereby significantly enhancing the reliability and security of the IoV.

A leading automobile company is exploring blockchain technology to enhance V2X data transfer and streamline the development of autonomous cars [123]. Blockchain technology addresses many challenges associated with the Internet of Vehicles (IoV). These problems include security, trust, transparency, and efficiency. As the Internet of Vehicles (IoV) evolves, these difficulties will become more critical, necessitating the development of innovative solutions that leverage blockchain technology to improve the safety and efficiency of existing transportation systems.

To guarantee the security of vehicular communications and data transmissions, several proposed designs use blockchain technology. A suggested architecture for secure ad hoc networking utilizes blockchain technology to provide trust and transparency in interactions among vehicles. This architecture employs a combination of original data imprints and transaction histories to ensure data integrity and facilitate the recovery of corrupted communications. This framework is particularly advantageous for multimedia files collected via diverse sensors [185].

Mirador et al. [120] suggest using blockchain technology for data authentication and vehicle identification. The research employs SUMO and OMNET++ with a custom-designed software and cryptographic method to ensure secure vehicle connection. The Internet of Vehicles may effectively use blockchain for data authentication and vehicle verification, based on the findings.

Regional blockchains have been explored as a viable method to enhance communication inside VANETs. These regional solutions may reduce message transmission time and enhance data transfer efficiency by limiting the geographic scope of blockchain operations. Moreover, other models have been introduced that integrate blockchain technology at multiple tiers of the Internet of Vehicles architecture. This connection facilitates the seamless storage of data and communication with roadside and vehicle-mounted communication nodes [132, 95].

Novel access authentication methodologies using blockchain technology have been developed to enhance privacy and security in networks, including autonomous cars [238]. These precautions maintain the anonymity of users, prevent the spread of false communications, and deter the surveillance of malicious entities. By decentralizing trust management, these methods reduce reliance on centralized authority, optimizing vehicle identification processes and improving the system's overall resilience. This reassures the audience about the safety of their data in the evolving landscape of network security. To cultivate trust and protect privacy, Lu et al. [139] developed the blockchain-based anonymous reputation system (BARS). BARS include certificates for law enforcement agencies, certifying bodies, automobiles, RSUs, and vehicle privacy. It employs three blockchain architectures: RevSC for revoked public keys, CerBC for all issued certificates, and MesBC for continuous evidence of reputational assessment. BARS achieves transparency, conditional privacy, and resilience via its reputation evaluation system, which considers message credibility.

Blockchain technology, through reputation management systems, has the potential to enhance trust and privacy within the Internet of Things (IoT). These systems can enhance the reliability of data transfers between vehicles by evaluating the legitimacy of communications and maintaining a transparent and accessible record of interactions for all parties involved. The integration of blockchain technology with edge computing can enhance the development of energy-efficient solutions, including Vehicle-to-Grid

(V2G) trading mechanisms [244]. This potential to transform the vehicle industry and address security issues within the Internet of Vehicles ecosystem is inspiring and opens up new possibilities for innovation.

3.5.9 Energy Sector

To address the challenges posed by the integration of renewable energy sources (RES) such as solar and wind, blockchain technology is swiftly empowering the energy sector [240]. Several causes, including privatization, the separation of the energy sector, and government incentives, are propelling the shift towards decentralized energy platforms. Blockchain, a distributed ledger technology, offers a decentralized trust mechanism capable of enhancing energy management and fostering innovative commercial solutions. The following are a few principal benefits that blockchain technology provides to the energy sector:

1. Distributed Energy Trading: Blockchain technology facilitates the establishment of decentralized energy trading systems, enabling direct communication among clients, power providers, and distribution managers. This technique enhances the efficiency and transparency of energy transfers.
2. Smart Contracts: These automated agreements help equilibrate supply and demand by enabling transactions based on real-time data. This facilitates independent regulation of energy flows, resulting in a more equal allocation of energy.
3. Data Security: Blockchain technology ensures that all data related to energy transactions is securely maintained inside a singular network. This ensures decentralized security for energy transactions and business operations.
4. Smart Grids and Devices: This technology facilitates the development of "smart grids" and "smart devices," which are crucial for enhancing awareness of renewable energy sources and optimizing energy efficiency.

The key emphasis of research in the energy blockchain domain is applications using renewable energy sources. In the context of energy systems, Bernd et al. bdenote the integration of blockchain technology as "blockchain energy," including many socio-technical configurations for data storage and energy transactions. Wu et al. [216] suggest a design for a hybrid blockchain aimed at facilitating efficient data management in the energy sector.

The potential of blockchain technology to revolutionize energy trading is immense, with numerous innovative applications being proposed. These include the following:

Park et al. [161] introduced a blockchain-based network using Internet of Things sensors to facilitate efficient power exchange among users. This network, known as 'Peer-to-Peer Trading,' allows users to directly buy and sell excess energy to each other, bypassing traditional energy providers.

Battery Replacement: Duan et al. [**?**]presented a decentralized mechanism to ensure equitable transactions during the replacement of batteries in electric vehicles (EVs).

Knirsch et al. [110] developed a blockchain-based framework for electric car charging that enhances transparency and privacy in the trade process. This platform, known as the 'EV Charging Infrastructure,' ensures that all charging transactions are recorded and verified on the blockchain, providing a secure and transparent record of energy usage.

Khaqqi et al. [104] proposed a blockchain-based system for 'carbon emission trading,' a process where companies can buy and sell permits to emit carbon dioxide, with the blockchain providing a secure and transparent record of these transactions.

Furthermore, Wang et al. [164, 245] and Kang et al. [98] introduced a streamlined blockchain billing framework for electric car charging—the framework aimed to consolidate payment mechanisms across diverse billing suppliers.

Blockchain technology is increasingly attracting attention in the energy sector because of its capacity to address contemporary challenges associated with distributed energy

and information technology. The decentralization aligns with the global trend towards dispersed energy generation, facilitating the integration of diverse energy systems and enhancing user engagement in energy transactions.

3.5.10 Government Services

The global adoption of blockchain technology in governmental services is on the rise, promising enhanced efficiency and a potential transformation of public administration [152, 145]. Among the most significant implementations are:

1. The Chinese government aims to streamline and accelerate financial processes by using blockchain technology for tax collection and invoice issuance.
2. To enhance public access to information, experiments are now underway in Japan for a blockchain-based system designed to manage government contract processes.
3. The United States government is exploring the integration of blockchain technology into its contract bidding process and is seeking contractors for evaluation.
4. The United Kingdom Government is now researching blockchain applications to automate the registration and payment processes related to government grants and benefits.
5. The Estonian government has successfully used blockchain technology across several domains, including official announcements, digital court files, property registrations, succession records, and business registers.
6. The Swedish government is exploring the potential of blockchain technology to streamline real estate transactions.

Blockchain technology has the potential to significantly transform government agency operations, diminish administrative duties, and restructure institutional frameworks. However, it is crucial to analyze the potential ramifications of these advances, which may include employment losses and the digital divide. Academics are urgently needed

to document these repercussions and collect insights from policymakers to facilitate the effective incorporation of blockchain technology into public services.

3.5.11 E-Commerce

A significant transition is occurring in the environment due to the increasing integration of blockchain technology with e-commerce. This transition is marked by enhanced security, efficiency, and transparency. The use of blockchain technology addresses several issues faced by the e-commerce sector, particularly in Indonesia, where challenges such as fraud, elevated commission costs, and insufficient direct interaction between customers and sellers are prevalent [44]. Many authors have acknowledged this observation. The decentralized ledger mechanism, a key feature of blockchain, eliminates the need for third-party validation, thereby enhancing both the speed and security of transactions. This technology safeguards transaction integrity and reduces the potential for manipulation, gaining significance as sectors like energy, banking, and e-commerce expand [226]. Blockchain technology applications provide several benefits, such as enhanced operational synergy, reduced costs, and streamlined business processes, particularly in supply chain management and financial services accounting.

E-commerce has significant obstacles, including client skepticism, especially during the epidemic, which has accelerated digital transformation [180]. Notwithstanding its advantages, e-commerce encounters significant challenges. The topic of transparency has surfaced as a significant concern, with notable figures like President Donald Trump highlighting the need for ethical standards in online trade, particularly with companies such as Amazon. Blockchain technology has the potential to enhance transparency by maintaining immutable transaction records on a communal ledger. This has the power to enhance customer confidence and trust in online transactions.

Businesses like Bitboots are using blockchain technology to provide decentralized platforms for online transactions. Furthermore, major retailers such as Amazon, Alibaba, and eBay are exploring the potential use of blockchain strategies [41]. Moreover,

blockchain technology has the capacity to address supply chain challenges by ensuring the authenticity and traceability of items. This is shown by Maersk's partnership with IBM on the TradeLens system, which provides real-time shipping data and reduces the volume of paperwork required. Other applications include smart contracts for secure and automated transactions, and tokenization for creating digital assets.

International e-commerce is benefiting from blockchain technology, which facilitates smooth transactions and saves costs, both crucial for the growth of global trade [79]. Lai et al. [121] proposes a blockchain-based solution for centralized cross-border logistics, while Jiang et al. [94] examine trust-based management and credibility modeling in e-commerce networks using blockchain technology and data mining.

Blockchain is revolutionizing e-commerce by significantly reducing costs, streamlining processes, and improving efficiency across several industries, including agriculture, banking, and transportation. This reassures us of the economic benefits of blockchain technology. Conversely, further research is necessary to assess its impact in other fields.

3.5.12 Military and Defense

In response to the vulnerabilities exposed by digitalization and cyber threats, blockchain technology is being swiftly explored for applications in the military and defense sectors [5].

Military leaders have been using cyber technology since the 1990s; nevertheless, they are now confronting the risks associated with cyber attacks that have historically penetrated both civilian and military networks. These assaults have raised apprehensions among the military. However, the potential for non-kinetic impacts to induce malfunctions in contemporary weapon systems has also sparked optimism, as it has prompted the armed forces to seek long-term and cost-effective alternatives, with blockchain technology offering a promising solution [112].

This study delves into the transformative potential of blockchain technology in the military, shifting the security vulnerabilities from a single-point failure paradigm to

a majority-compromised vulnerability framework [114]. It also highlights the diverse applications of blockchain technology across various military sectors, including the identification of unauthorized access, the monitoring of critical infrastructure, the orchestration of combat operations, the management of uncrewed aerial vehicles (UAVs), the regulation of supply chains, and the encryption of communications.

A comparative research presents an examination of the blockchain applications used by three major military nations [128].

1. The United States is actively leveraging blockchain technology to enhance data resilience and as a cybersecurity measure to prevent data tampering. This strategic use of blockchain is a significant step in bolstering the country's defense against cyber threats.
2. Russia has launched a research center aimed at developing a blockchain system to detect and mitigate assaults on military digital infrastructure.
3. The Chinese government is concentrating on the use of blockchain technology for equipment management, professional training, transportation, and the integration of commercial technologies with military initiatives.

The potential of blockchain technology lies in its ability to distribute governance and enhance security, despite the contradictory participation of centralized military organizations in decentralized blockchain initiatives. The current focus on defense logistics and data protection underscores the need for a more effective and secure use of blockchain technology in military environments.

Blockchain technology offers substantial benefits across several industries, including enhanced efficiency, security, and transparency. The financial sector optimizes procedures, including fund administration and credit reporting, while enhancing risk management and authenticity. In healthcare, blockchain technology addresses concerns about privacy and security, facilitating the secure management of data and improving patient care. The education sector is seeing advancements in credential management

and customized learning experiences, while the supply chain business is gaining from improved traceability and less fraud. Integrating blockchain technology with the Internet of Things (IoT) may enhance the transparency and efficiency of the agricultural supply chain, while simultaneously improving the security and reliability of connected devices. Nonetheless, challenges persist, including implementation difficulties, legislative hurdles, infrastructure development requirements, and concerns around data privacy and security. Although blockchain technology has the potential for transformative change, it is crucial to resolve these challenges to guarantee its widespread adoption and effectiveness across all industries.

3.6 Blockchain Attacks

Risks that may impact the Blockchain Network:

1. Sybil Attacks: Entail the creation of several fictitious identities to influence the network, impacting both Proof of Work (PoW) and Proof of Stake (PoS) mechanisms [54]. Such assaults may result in selfish mining and double expenditure.
2. Eclipse Attacks: Perpetrators commandeer neighboring nodes to sever a target node from the main network, enabling them to alter blockchain data [174] and abuse the target's resources.
3. DNS Attacks: Upon the addition of a new node, adversaries may use weaknesses in the Domain Name System [108] to provide fraudulent peer lists, therefore isolating victims and providing them with erroneous block information.
4. BGP Hijacks: This routing assault targets clusters of nodes by compromising Internet Service Providers (ISPs) [38], notably impacting networks with concentrated nodes, such as Bitcoin, which is susceptible owing to its dependence on a single ISP.
5. Timejacking: Sybil nodes distort a target node's network time [177], resulting in the rejection of legitimate blocks and subsequent isolation.

6. Denial of Service (DoS) Attacks: These attacks target system availability and may be distributed (DDoS) [30], having previously resulted in substantial problems inside blockchain networks, including a hard split in Ethereum.
7. Mempool Flooding: Malefactors inundate the memory pool with a multitude of low-fee transactions, resulting in escalated transaction backlogs [177] and elevated costs for genuine users.
8. Blockchain Denial of Service (BDoS): A particular DoS [30] assault aimed at PoW systems, whereby attackers mine blocks without disseminating them, hence weakening miners' incentives and possibly disrupting mining operations.

These dangers exemplify the weaknesses in blockchain networks that may be exploited by nefarious entities.

4 Understanding Blockchain Security and Privacy

A thorough risk management solution for a blockchain network is called blockchain security. Combining cybersecurity tools, best practices, and concepts, blockchain security helps to reduce risk and stop hostile attacks on blockchain networks.

4.1 Basic Blockchain security

Data structures created by blockchain technology include built-in security features. Its foundations are consensus, decentralization, and cryptography, all of which promote transaction trust. The data is organized into blocks in the majority of blockchains and distributed ledger technology (DLT), and each block includes a transaction or group of transactions. The following enumerates several significant concepts and principles of security [96]:

- Defense in penetration: This strategy employs a range of corrective measures designed to safeguard the data. The principle posits that safeguarding data through multiple layers is more effective than relying on a singular security layer.

- Minimum privilege principle: This strategy minimizes data access to the lowest possible level in order to strengthen the overall security framework.
- Administer vulnerabilities: This strategy involves assessing vulnerabilities and managing them through the processes of identification, authentication, modification, and patching.
- Administer risk management strategies: This strategy involves systematically processing risks within an environment by identifying, assessing, and controlling potential hazards.
- Oversee the management of patches: This strategy involves addressing the flawed components, such as code, applications, operating systems, and firmware, through the acquisition, testing, and installation of patches.

Every new block forms a cryptographic chain with every previous block [234], making it nearly impossible for any one to tamper with. A consensus mechanism verifies and approves each transaction within the blocks, guaranteeing that every transaction is accurate and true.

Decentralization is made possible by blockchain technology and the involvement of users throughout a dispersed network. The transaction record cannot be altered by a single user, nor is there a single point of failure.

Distributed ledger technology (DLT) powers all blockchains, however not all of them are equally safe or functionally sound. The open versus closed nature of public and private blockchain networks results in profoundly different security mechanisms, despite the fact that each type of blockchain has pros and cons of its own.

4.1.1 Security measures in Public Blockchain

Public blockchains [82], such as Ethereum and Bitcoin, are transparent, permissionless networks that allow anybody to validate a transaction. A developer community regularly checks their open-source existing code for errors and vulnerabilities. These

blockchains' efficiency, functionality, and security are all enhanced by this group's collective knowledge. Still, malevolent organizations and hackers are always scanning the code for weaknesses to take advantage of.

Accountability for Public Blockchain Security

A public blockchain like Ethereum is owned by its founders, who also actively contributed to the network's development and wrote the original source code. Nonetheless, all participants—validators, node operators, and developers—share overall accountability for network security. Users who follow proper security hygiene also help to keep the network secure. Because it is decentralized, it is immune to several types of attacks because no one party can take full responsibility for it [209].

Who Maintains and Develops Public Blockchain?

Public blockchains with organizations devoted to development and community involvement include Bitcoin and the Ethereum Foundation [213]. Satoshi Nakamoto, the person who designed Bitcoin, has a committed group of maintainers who work on it constantly. Since it's a living system, it needs to be maintained on a regular basis to fix problems and adjust to changing conditions. A Bitcoin Improvement Proposal (BIP), which may be made by anybody, is a proposal for a change to the core network that needs to be approved by consensus.

4.1.2 Ensuring security in Private Blockchain

Private blockchains are private networks with restricted access, which increases their centralization and may increase their defense against outside attacks. Since there is only one point of failure in these networks, the operational entity strong security measures are essential. Because consensus methods need less computing power, private blockchains are typically quicker and more efficient than public blockchains, even though they might not benefit from decentralized security to the same extent. Though it is uncommon with public blockchains, the power of the giving entity also raises the possibility of network manipulation or shutdown.

4.2 Securing Transactions in Blockchain

A private key, a cryptographically secure means of access and authentication, protects blockchain transactions. Cryptocurrency transactions are started peer-to-peer, without the need for a middleman, unlike traditional finance. Since there is no need for a middleman when transferring value on-chain, personal accountability becomes essential. Nevertheless, verified transactions cannot be reversed, which makes it challenging to get lost or stolen money back. Cryptocurrency transactions become more efficient and safe as a result.

4.3 Blockchain Privacy

The concept of privacy in the context of blockchain technology [96] refers to the ability of individuals or groups to engage in transactions while maintaining the confidentiality of identifiable information, thereby facilitating discreet transactions and compliance with regulatory frameworks. The enhancement of privacy within blockchain technology seeks to safeguard users' cryptocurrency profiles from unauthorized access and duplication, a concern that is especially relevant considering the varied applications of blockchain systems. The key characteristics that influence privacy encompass several critical aspects. Firstly, the categorization of stored data varies significantly between personal and organizational contexts, with the latter being governed by more rigorous privacy regulations. Secondly, the distribution of storage across full nodes promotes data redundancy, thereby enhancing transparency and verifiability. Additionally, the append-only nature of blockchain technology serves to prevent undetected alterations to historical data, while simultaneously necessitating a careful examination of users' rights to rectify inaccuracies.

Furthermore, the distinction between private and public blockchains enables [247] the encryption of sensitive data, granting access only to authorized individuals. Lastly, the differences between permissioned and non-permissioned blockchains have profound

implications for user control and the function of trusted mediators within the network. The interplay of these factors highlights the complex challenges involved in preserving privacy within blockchain systems while striking a balance between transparency and user rights.

4.3.1 Which techniques are securing blockchain and how?

A distributed digital ledger system made up of nodes that verify and log transactions underpins blockchain technology. Every bitcoin transaction, including sending and receiving, is recorded on a block that has to be confirmed by consensus using the widely used Proof-of-Work and Proof-of-Stake techniques [234] (used to generate new tokens, validate new transactions, and put them to the blockchain). While proof-of-stake involves participants locking up tokens to run nodes in order to confirm transactions, proof-of-work relies on miners solving computationally demanding algorithms. Rewards are given to miners and stakers to encourage them to protect the network.

A block is sealed when it is filled and connected to the one before it by a cryptographic code to create a chain. This guarantees that any effort to manipulate a block would cause the chain as a whole to break. Since everyone can see the ledger, suspect activity may be quickly identified.

Popular cryptocurrencies like bitcoin [178] and ether are built on blockchain technology, which has enormous promise for the development of digital transactions in the future. Its integrity is upheld by each participant, which makes it an essential technology for digital transactions in the future.

5 An Overview of Ether (Eth)

Created in 2015, Ethereum's global virtual computer uses ether as a coin [239]. It serves a number of purposes like other digital currencies: It's employed to compensate network users for their blockchain-related efforts. Traders utilize it to profit from price

changes, and investors use it as a store of value. It may be used by customers to pay for products and services at establishments that accept it. On the Ethereum network, it is also employed to facilitate the creation of apps. The term Ethereum is frequently used to refer to both Ether and the network.

Let's take a closer look at the Ethereum platform in order to comprehend the usage of Ether better.

5.1 Understanding Ethereum

Ethereum is a public [35], transparent, decentralized blockchain-based software infrastructure that helps DApps [143], or decentralized applications, and peer-to-peer contracts, or smart contracts. Investors and developers are primarily familiar with it through its native cryptocurrency, ether (ETH), and its use in blockchain and distributed finance development.

Users can trade value using smart contracts without the need for a middleman. Smart contracts are contracts with explicit terms and enforcement mechanisms built in.

However, smart contracts are written in code that a machine can execute, which removes ambiguity, in contrast to traditional contracts, which are written in human languages and upheld by legal authorities.

The smart contract code is executed by the Ethereum network, which functions as a single decentralized computer. This implies that the decisions made about each smart contract's conclusion will be agreed upon by every computer connected to the Ethereum network.

Conventional software programs require trust since they store and process data through central authority. Using Ethereum smart contracts [147], decentralized apps (DApps) may become decentralized. Without the need for a central, trustworthy party, these contracts maintain data integrity by storing it and enforcing actions in accordance

with the smart contract code.

Key Points to Remember

- Ethereum is a development platform built on the blockchain that is well-known for its cryptocurrency, ether (ETH).
- Ethereum's blockchain technology makes it possible for secure digital ledgers to be made and updated by the public.
- While Ethereum and Bitcoin share many characteristics, their long-term goals and constraints are distinct.
- Ethereum validates transactions with a proof-of-stake system.
- A lot of the newer blockchain-based technology advancements are built on Ethereum.

Transactions in Ethereum [73] can establish new contract accounts or engage with existing ones. When interacting with external accounts, only Ether may be exchanged, whereas contract accounts run corresponding code, which may modify storage or trigger internal transactions. The Ethereum Virtual Machine (EVM) [77] operates as the execution model for contract code, acting as a quasi-Turing complete system. Execution is constrained by a predetermined gas resource, which limits the number of operations. The transaction initiator establishes a maximum gas limit and gas price, remitting payment in advance according to these criteria. If the contract execution does not consume the entire allocated gas, the originator is compensated for the unutilized quantity. The miner obtains the residual wei compensated for the gas as a charge. This framework guarantees the management of transaction costs and promotes the efficient execution of contracts. The EVM's architecture effectively balances adaptability and resource allocation inside the Ethereum network.

EVM Bytecode

The EVM Bytecode [125] language serves as a bytecode implementation for executing Ethereum smart contracts, resembling assembly language. This component comprises standard instructions for stack operations, arithmetic, jumps, and access to local

memory. A variety of opcodes for accessing the contract's execution environment, together with distinct opcodes for performing internal calls and initiating transactions, have been included into the conventional instruction set, which has been augmented with an opcode for the SHA3 hash. Terminate the contract currently in execution only upon the successful conclusion of the external transaction. This exemplifies a specific command included inside the blockchain configuration.

The transaction sender must ascertain the maximum quantity of gas that may be used, since each instruction expends a certain amount of gas. Upon exceeding this limit, an exception is triggered, resulting in the reversion of the current transaction's effects on the global state to their initial condition. An exception in a nested transaction solely nullifies the results of that specific exception, without affecting the outcomes of the calling transaction.

5.1.1 Solidity

In Ethereum, contracts are expressed using Solidity [45], a high-level programming language developed by the Ethereum Foundation. The modeling of contracts is achieved by the use of the class concept, originating from object-oriented programming languages. This enables the delineation of fields and functions for contract instances. Instead of just executing transactions to a recognized contract address that contains a contract of the correct kind, contracts may be generated as new instances or built directly.

Transferring funds to another contract in Solidity [50] presents several idiosyncrasies, especially in a low-level operational context. Employing these functions to initiate a value transfer transforms the transaction into an internal call. Invoking a contract may lead to code execution, which might fail due to insufficient gas availability. Fallback functions may also be implemented for contracts in Solidity. These procedures are executed when a call is made via the function or an address is invoked that fails to accurately specify the concrete function of the contract intended for invocation.

5.2 How does Ethereum work?

Blockchain Technology in Ethereum

The distributed ledger employed by Ethereum is referred to as a blockchain, operating analogously to a database [197]. Data is structured in blocks, with each block comprising the most recent information and encoded details from the prior block, establishing a predetermined sequence of encoded data. A meticulous replica of the blockchain is disseminated throughout the whole blockchain network.

Data is preserved in blocks on the Ethereum blockchain. Each block comprises a sequence of transactions that can be documented on the Ethereum blockchain, encompassing significant data.

- This data in these transactions may include cash transactions or outcomes from executing a smart contract.
- Every node in the Ethereum network retains the data, ensuring its security and accessibility for all users.

To authenticate the data within a block and propose a new one, the validator is rewarded with additional ether tokens for each cell or block generated. The validator's address is allocated to the ether.

An automated algorithmic mechanism achieves consensus on the accuracy of transaction data [219], validating each newly submitted block. Consensus is attained once data and hashes circulate between the consensus layer and the execution layer on the Ethereum blockchain. The block is finalized when a sufficient number of validators verify that their comparison results are consistent.

Contents of Data Transaction

The transaction data on the Ethereum blockchain contains information about the transaction, such as the amount of Ether exchanged, the addresses of the sender and receiver, and the outcomes of the implementation of the smart contract, such as the

contract output and any state modifications.

Ethereum stores the current state of the blockchain, including account balances and the code and data for each smart contract, in a trie data structure. The root node of the trie describes the current state of the Ethereum blockchain, whereas each node in the trie represents a single piece of data. Every time a transaction is carried out on the blockchain, the trie is modified.

The Ethereum Blockchain has three different kinds of trie [92]: state, storage, and transaction. Every account in the Ethereum Account has a key and a value, and State Trie is updated continuously. The smart contract is located in the Storage Trie, where the storageRoot is a 256-bit hash of the root node that is kept in the global state trie. Every Ethereum block is located at a place called Transaction Trie, and each transaction's route is defined by its index within the block.

Retrieving Data. An Ethereum node must receive a request in order to collect stored data from Ethereum. The Ethereum node will then start looking at its local copy of the blockchain for the required data. The data hash determines the precise placement of the data in the trie. The Ethereum blockchain may be queried using Web3.js, a JavaScript tool for communicating with Ethereum nodes.

Proof-of-Stake: How validation happens in Ethereum?

Proof-of-stake is a decentralized system that uses the LMD Ghost and Casper-FFG algorithms in conjunction with Gasper, a consensus mechanism, to provide validation without the need for mining.

- In order to activate their validation capabilities, validators need to invest 32 ETH.
- A validator certifies the authenticity of a newly created block.
- The authenticity of the block is confirmed and voted upon by a committee.
- The network removes and burns the staked Ethereum of dishonest validators as a kind of punishment.

- Ethereum stores data in chunks on a distributed ledger called a blockchain.
- New ether tokens, given to validators as compensation for their labor, are used to generate each block.
- After data and hash are sent between the consensus and execution levels, automated systems validate a new block and come to a consensus.

5.3 Understanding EVM bytecode conversion into opcode

Ethereum is an open-source platform that utilizes blockchain technology to create a decentralized network where developers can build applications known as decentralized applications. Central to its functionality are smart contracts, which are self-executing contracts with the terms of the agreement between buyer and seller directly written into lines of code. These contracts control the transfer of Ethereum's native cryptocurrency, Ether, and can automate a wide range of actions beyond mere fund transfers. Every transaction made on the Ethereum network is a record of an action, such as fund movement, a change in a smart contract's state, or the deployment of a new smart contract. Once a transaction has been confirmed and included in a block, it becomes immutable, meaning it cannot be altered or reversed. This block is then added to the blockchain, further confirming the decentralized and secure nature of the ledger. The permanence and transparency of transactions contribute to the trustworthiness and security that blockchain technology is known for [64].

On the Ethereum blockchain, smart contracts are typically written in Solidity, a high-level programming language specifically designed for creating and implementing smart contracts. Before a smart contract can be deployed and run on the Ethereum network, its Solidity source code must be compiled into bytecode. This is the low-level, machine-readable language that the Ethereum Virtual Machine understands. The EVM is the runtime environment for smart contracts in Ethereum, executing bytecode in a deterministic, isolated, and secure manner. Compiling to bytecode is a crucial step

because it transforms the human-readable Solidity code into a format that can interact with the EVM and, by extension, with the Ethereum blockchain itself.

Ethereum Virtual Machine (EVM) bytecode is the low-level representation of smart contract code that runs on the Ethereum blockchain. It is generated by compiling high-level programming languages like Solidity into a format that the EVM can understand and execute. EVM bytecode consists of a series of opcodes, each representing a specific operation that the EVM can perform. When a transaction is sent to interact with a smart contract, the EVM processes the bytecode to execute the desired function of the contract. This process involves reading the opcodes, interpreting them, and executing the corresponding operations. EVM bytecode is stored on the blockchain along with the transaction history, ensuring that the code and its execution are transparent and immutable. The EVM is designed to be deterministic, meaning that the same bytecode will always produce the same result when executed under the same conditions. This property is essential for ensuring the integrity and predictability of smart contract execution on the Ethereum blockchain. EVM bytecode plays a crucial role in the execution of smart contracts on the Ethereum platform, providing a secure and reliable way to automate transactions and enforce agreements without the need for intermediaries [160].

The translation of Ethereum Virtual Machine (EVM) bytecode into smart contract opcodes is a crucial step in many analyses and tools designed to enhance Ethereum smart contract security. This translation is crucial because opcodes represent the specific instructions that the Ethereum Virtual Machine (EVM) can execute. By decoding these opcodes, we gain a deeper understanding of the smart contract's behavior, which is essential for both manual and automated analysis processes, including the detection of security vulnerabilities.

The conversion from Ethereum Virtual Machine bytecode to opcodes is a process of interpreting the binary code into human-readable instructions. The EVM executes

these opcodes to perform various operations, including arithmetic calculations, stack manipulations, and storage handling. Each bytecode instruction corresponds to an opcode; typically, a byte maps to a single operation, although PUSH instructions also include subsequent bytes for data.

Flow control in EVM bytecode is not function-oriented; instead, it utilizes stack-based jumps. Unconditional jumps use the JUMP opcode, while JUMPI manages conditional ones, with the stack providing the destinations. In place of traditional function calls, EVM contracts use a dispatcher, which routes execution based on function signature hashes matched against the incoming call data, with a fallback function serving as a default.

During compilation in Solidity, function calls are translated into EVM bytecode that sequentially pushes parameters and jump destinations to the stack, followed by JUMP operations to invoke and JUMPDEST operations to mark function boundaries. After the function code executes, the control returns to the calling context, simulating the function call mechanism.

Here is a general overview of our method for the conversion process:

- **Parsing Bytecode**: Start by splitting the bytecode into individual bytes. (1 byte): B_0 = b_0b_1...b_7, B_1 = b_8b_9...b_15, ..., B_(n-1) = b_(4n-8)...b_(4n-1).
- **Identifying Opcodes:** Each byte in the bytecode corresponds to an EVM opcode. Opcodes are the fundamental operations that the EVM can perform, including arithmetic, stack manipulation, and storage operations. Most opcodes are identified by a single byte, but some opcodes in the PUSH family require additional bytes to specify the data to be pushed onto the stack.
- **Decoding Opcodes:** For most opcodes, the byte itself represents the opcode. However, for opcodes in the PUSH family, the opcode specifies the number of bytes to be read as the parameter. For example, if the opcode is PUSH1, the next byte

contains the value to be pushed onto the stack. If the opcode is PUSH2, the next two bytes contain the value, and so on up to PUSH32.

- **Handling Parameters** Extract the parameter for PUSH opcodes by reading the specified number of bytes following the opcode and interpreting them as a value to be pushed onto the stack.
- **Executing Opcodes** Once the opcodes and their parameters have been decoded, you can execute them sequentially to simulate the execution of the smart contract.
- **Special Cases** Some opcodes, like CODECOPY, can be used to treat a portion of the code as data. When encountering such opcodes, the interpreter should handle them appropriately to avoid treating data as executable code.

5.4 Benefits of storing data on Ethereum

Because of its immutability, decentralization, accessibility, and capacity to carry out smart contracts, the Ethereum blockchain is a well-liked option for data storage [101, 8]. Data cannot be changed or removed because of its immutability, which makes it a safe and dependable platform. Decentralization guarantees that data is not under the control of a single party, increasing its security and resistance to censorship. While smart contracts enable self-executing agreements with their conditions encoded directly into the code, accessibility makes it simple to exchange and collaborate on data.

6 Let's dive into Smart Contract

Smart contracts [99] are self-executing contracts that operate on the blockchain platform. They can store complicated logic, guidelines, and terms that control transactions and interrelations between participants. However, smart contracts confront significant issues when it comes to data storage and retrieval, including cost, flexibility, safety, and confidentiality. In this article, you will explore how to store smart contracts and secure them from potential vulnerability attacks.

6.1 Storage options in smart contracts

One of the first concerns you must make when developing a smart contract is where to keep the data it requires or creates. You have two basic storage options: on-chain and off-chain. On-chain storage refers to the actual storage of data on the blockchain utilizing the smart contract's state variables and mappings. This has the virtue of assuring data integrity, authenticity, and accessibility, but it comes at a considerable cost in terms of gas and storage capacity. Off-chain storage refers to storing data beyond the blockchain, utilizing other databases or assistance. This can lower the cost and improve the adaptability terms of information security, the smart contract, but it also poses some dangers and trade-offs in terms of its security, and availability. You must select the storage solution that best matches data size, frequency, and responsiveness.

6.2 What is the structure of Data in Smart Contracts?

Smart contract data management necessitates organizing and optimizing data structures such as arrays, structs, mappings, and enums [8]. These structures have a considerable influence on performance, effectiveness, and legibility. To improve performance, use the smallest data types feasible, utilize mappings whenever possible, eliminate loops and recursive operations, and arrange relevant variables together. Structs are useful for grouping variables, increasing readability and maintainability, and preventing out-of-gas issues. Avoiding loops and recursive calls can save both gas and storage space.

Integrating Oracles and Events in Smart Contracts

Smart contracts can connect or give data that is not yet accessible on the blockchain by leveraging oracles and events to promote communication between the contract and the outside realm. Oracles offer or validate data, whereas events send data from smart contracts into the outside world. To ensure consistent and safe data distribution, use a decentralized Oracle network that collects data from numerous sources and nodes,

such as Chainlink or Band Protocol [61]. Use events to send data that is helpful to users or programs, and use the emit keyword to proclaim and execute events.

Best Practices and Standards for Smart Contracts

Smart contract data management should follow blockchain community and regulatory requirements. These requirements, which include ERC-20 for fungible tokens and ERC-721 for non-fungible tokens [138], describe the design, execution, and interconnectivity of smart contracts. OpenZeppelin [17] for reusable libraries, the Solidity Style Guide for coding norms, and ConsenSys Diligence [215] for security patterns are examples of guidance aimed at improving the quality, security, and functionality of smart contracts. Having followed these principles will ensure that smart contract data is valid, interoperable, and fully comply with the blockchain ecosystem's preconceptions.

Smart contracts on the blockchain are increasingly being used for a number of applications, making their security a crucial concern. Smart contracts support a variety of functionalities, including decentralized voting [16], gaming, and non-fungible token transactions. Nevertheless, as they become more popular, the potential of harmful assaults increases. To avoid this risk, developers must emphasize smart contract security throughout the development, deployment, and usage processes.

6.3 Ensuring Smart Contracts Security in Blockchain

Smart contracts are an important part of the blockchain ecosystem, but unfortunately, correct execution of the smart contract code does not guarantee its entire safety and it is critical to design safe and robust contracts that guard against attacks and limit the chance of events. In reality, an examination of current smart contracts reveals that a large number of them are genuinely insecure.

This section will examine the primary security threats and weaknesses associated with the deployment of smart contracts on the Ethereum platform. Smart contracts possess the capability to manage substantial amounts of virtual currencies [140], perhaps worth

in the thousands of dollars. Consequently, unscrupulous persons frequently attempt to use these contracts for personal gain. The execution of smart contracts occurs on permissionless and decentralized networks, inherently characterized by several vulnerabilities. The failure of smart contract execution may result in substantial losses financially [190, 26]. Once established, smart contracts on a decentralized blockchain cannot be modified or changed without significant effort. This contrasts with traditional systems, which permit the reconstruction or patching of troublesome programs inside a centralized environment. The immutability presents both advantageous and detrimental consequences to the system's security [144]. While it prevents hackers from using contracts for their benefit, it simultaneously limits developers from altering the contracts post-deployment. They can only establish a new contract and terminate the existing one. Therefore, it is imperative to conduct comprehensive testing of smart contracts utilizing diverse test scenarios to ensure their safety and security before deployment. Nevertheless, certain fundamental characteristics of contemporary blockchains may induce security vulnerabilities in smart contracts.

- The decentralized and tamper-resistant characteristics of blockchain technology may lead to vulnerabilities, especially with smart contracts created by pseudonymous individuals.
- The susceptibility of smart contracts: Once deployed, flawed smart contracts are complex to rectify and may become ungovernable. This is due to their susceptibility to specific vulnerabilities [80].
- Public ledgers and open-source software serve as two examples. Generally, smart contracts are open-source, indicating that their transactions and data are publicly accessible. Adversaries may exploit this transparency.
- The underdevelopment of blockchain systems is a significant concern. Blockchain technologies may exhibit design flaws due to their rapid evolution; thus, developers must

possess a thorough understanding of blockchain operations to avert inconsistencies in smart contract implementation.

- Prevalent software vulnerabilities: The unique conditions inherent in the development of decentralized applications (dApps) may amplify conventional software faults inside blockchain environments.
- Misapprehension of Practices: Developers may lack a comprehensive knowledge of the principles behind blockchain technology, particularly in cryptography, thus leading to erroneous assumptions on security.
- Pseudonymous Transactions: The pseudonymous nature of transactions on public blockchains may facilitate illicit activities like money laundering [97] and Ponzi scams [36].

7 Vulnerabilities

Smart contracts can be susceptible owing to a variety of issues, including flawed business logic, insecure code, and external dependencies that cause anomalous behavior. Properly built smart contract languages, such as Solidity, are turned into bytecode that is put on the Ethereum blockchain and comprehends the Ethereum Virtual Machine (EVM). Other languages, such as Vyper **??**, which utilizes Python, address security concerns but provide less developer assistance. Lower-level languages such as Yul **??** enable more optimized and gas-efficient smart contracts, albeit at the cost of ambiguity. The balance between safety and functionality is an important consideration for developers when selecting a smart contract language, making Solidity the preferred choice on Ethereum and other EVM networks. Furthermore, smart contract language is not the sole vector of attack; blockchain execution designs might also be a factor.

7.1 Uncovering the reasons behind Smart Contract hacks

There is a growing presence of vulnerabilities in smart contracts, which can be attributed to a variety of variables that are associated with the operation of blockchains, developer practices, and the architecture of contracts. Several underlying factors contribute to the vulnerability of smart contracts to security breaches, including the following:

1. **The complexity of the code**: The sophisticated nature of the code that is required for the creation of smart contracts allows for the possibility of faults and errors occurring during the development process. Because of this complexity, there is a greater possibility that vulnerabilities will be introduced.
2. **Decentralized Environment**: Smart contracts operate in a context that is decentralized and trustless, which means that there is no central authority to monitor or oversee the execution of these contracts. As a result of this lack of control, contracts may be susceptible to unfair treatment.
3. **A Target for Hackers**: Because smart contracts frequently assist financial transactions and valuable exchanges, they become excellent targets for hackers. There is a potential for weaknesses or vulnerabilities in the contracts to be exploited for nefarious purposes.

As a result, strengthening the security of smart contracts after deployment can be achieved by identifying flaws in smart contracts before they are deployed. Since the code for smart contracts is developed manually, there is a significant possibility that errors or defects may exist, which hackers can exploit.

While diving into the technical aspects of these attacks, let's discuss why hacks occur first!

Smart contract vulnerabilities lead to hacks. They can still be classified as complex concerns, as we discussed, such as incorrect business logic, insecure code, or unattended interactions with external sources. While technology advancement and developer expertise are critical for mitigating attacks, acknowledging the human factor both

users and attackers is equally important. With that being said, the rest of this article will focus on potential vulnerabilities and help you conceptually grasp the existing methodologies of smart contracts vulnerabilities detection tools and how to secure smart contracts with newly innovative methods.

7.2 Types of Vulnerabilities

Reentrancy Attack

Reentrancy **??** has a lengthy history in Ethereum. The DAO Hack, a notable assault on Ethereum in 2016, was ascribed to the vulnerability category of reentrancy, which has been acknowledged by several specifications, including restricting gas transmitted by external guards, reentrancy guards, and adhering to CEI patterns.

Understanding Reentrancy Vulnerability

To comprehend the reentrancy vulnerability, consider the following example:

We have two smart contracts in Fig 4: Contract A and Contract B. Contract A has

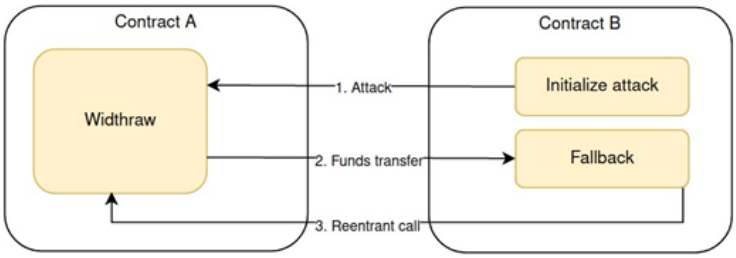

Fig. 4 Reentrancy Vulnerability

a function that allows users to withdraw their funds, while Contract B represents an attacker-controlled contract. The vulnerability arises when Contract B maliciously calls the withdrawal function in Contract A. In a vulnerable smart contract, the following sequence of events can occur:

1. An attacker initiates a withdrawal from Contract A.

2. Contract A starts executing the withdrawal function and transfers funds to the user.
3. Before Contract A can complete the withdrawal, Contract B makes a reentrant call back into Contract A.
4. Inside the reentrant call, Contract B can again call the withdrawal function in Contract A.
5. This process can continue recursively, causing Contract A to lose more funds with each reentrant call.

The result is that Contract B can drain Contract A's balance, leading to a loss of assets and potential contract failure.

Reentrancy is a vulnerability that enables attackers to enter a function several times before the original call is completed. As a result, anytime the contract makes external calls to other locations, there is the risk of a reentrancy attack. This might cause unexpected behavior, such as rearranging transactions, and can be used to drain cash from a contract.

The reentrancy attack in Solidity **??** makes use of fallback functions, which are nameless, externally called structures that are activated in specified scenarios. These functions can be zero or one per contract, and they are automatically activated when an additional contract calls a function in the fallback encompassing smart contract that does not match the function name. It can also be triggered if ETH is provided to the fallback's enclosing contract without any "calldata" or a declareable receive() function. Fallback functions may contain arbitrary logic.

Types of Reentrancy Attacks. This section seeks to explain the types of reentrancy:

- **Single-Function Reentrancy:** Reentrancy vulnerabilities were identified and exploited in a single function, where an external call causes the function to execute numerous times before its first implementation, resulting in a series of incomplete state transitions and possibly serious consequences.

- **Cross-Function Reentrancy:** Although the solution for single-function reentrancy reduces security concerns, cross-function reentrancy, in which many functions share the same state, can still represent a hazard in more complicated circumstances, such as when other code calls withdraw.
- **Cross-Contract Reentrancy:** Cross-contract reentrancy is a recursive loop in smart contracts in which a function contacts another contract before its first execution, which can result in many state transitions and significant difficulties such as fund draining or contract logic deception if not controlled correctly.
- **Read-only Reentrancy:** A specific example of a cross-contract reentrancy attack is read-only reentrancy, which occurs when the behavior of one smart contract is determined by the state of another. Attackers concentrate on state-changing operations, however outdated state information could lead to third-party infrastructure exploitation in circumstances where views provide outdated state data.

Integer Overflow Underflow

An integer overflow or underflow vulnerability **??** occurs when an operation surpasses the maximum or minimum value of an integer variable due to inadequate safeguards against such occurrences. In the POWH Coin Ponzi scam, an assailant exploited a vulnerability by altering the balance of a target contract, resulting in an erroneous reset and enabling the assailant to extract excessive cash. This vulnerability may be avoided by using secure arithmetic libraries such as SafeMath in Solidity, which enforce appropriate bounds checking to prevent similar vulnerabilities.

Attackers can exploit contract weaknesses by intentionally inputting values that surpass or fall short of the permissible range, resulting in undesired consequences such as unlawful financial transfers or unforeseen actions. To alleviate these risks, developers should choose appropriate data formats and meticulously verify boundary conditions throughout mathematical operations. Moreover, implementing input validation and

access control methods is essential to prevent the processing of malicious input data, thereby enhancing the contract's security.

Transaction order Dependency

The Transaction Ordering Dependency **??** issue arises in blockchain systems when the sequence of transaction execution is not guaranteed, resulting in erratic states of smart contracts. The ultimate state of a contract depends on the sequence in which miners execute transactions when multiple users concurrently initiate transactions to the same contract. This may lead to situations when users, including buyers and sellers in a decentralized market, encounter unforeseen results, such as incurring higher costs than planned owing to changes in contract status prior to the execution of their transactions.

Locked Ether

In Locked Ether(LE), **??** funds sent to a smart contract become permanently inaccessible due to bugs or design flaws, resulting in a loss of liquidity and access to funds. This scenario mirrors the Parity Wallet bug in November 2017, where locked ether vulnerability led to the locking of millions of USD worth of Ether.

Unhandled Exception

Unhandled Exceptions (UE) **??** in smart contracts can lead to unpredictable states and potential exploits, resulting in financial losses, erratic behavior, and contract malfunctions.

Tx.Origin

In Ethereum smart contracts, the address of the account that initiated a transaction can be viewed through the global variable Tx.Origin **??**. When developers employ Tx.Origin to verify calls from external accounts, mistakenly assuming that it represents the original originator of the transaction, which is referred to as a Tx.Origin attack.

However, Tx.Origin represents the ultimate initiator of the transaction, which can lead to security vulnerabilities.

Time O

TimeO (TimeOut) **??** is a race condition vulnerability in Ethereum smart contracts that arises when the result of a contract's operation depends on the sequence of transactions submitted to it. Transactions in Ethereum are organized into blocks, with a confirmation time of around 17 seconds. Miners prioritize transactions based on gas price, allowing them to predict transaction order before finalization. This vulnerability can be exploited, for instance, in a reward-based smart contract, where a malicious node can manipulate the transaction order to claim rewards unfairly.

Time M

Contracts often use timestamps to manage time-sensitive operations, such as locking a token sale or unlocking funds at a specific time. However, miners can manipulate the block's timestamp, allowing them to insert their transactions with changes to the contract's state before a victim's transaction in the same block. This manipulation can lead to unauthorized actions or other malicious activities that exploit the TimeM **??** vulnerability, highlighting the importance of secure contract design and mitigating timestamp dependence vulnerabilities.

Parity Multi-Sig Wallet

The Parity Multi-Sig Wallet vulnerability [105] stems from insufficient access control in its public library, enabling unauthorized users to execute critical tasks. An assailant exploited this vulnerability by establishing a multi-signature wallet and asserting ownership, resulting in the loss of more than 15 million Ether from 151 wallets. The assault highlighted problems such as inadequate file access and information leakage, accentuating the need for more stringent access modifiers in smart contract development

to prevent analogous exploitation.

However, these vulnerabilities not only affect the financial system but also have far-reaching implications for the digital world. Ether's substantial losses highlight the need to detect and address these vulnerabilities. The primary objective of this systematic review is to develop an approach that can not only detect reentrancy attacks but also identify other vulnerabilities that have led to significant losses in smart contracts. By understanding and mitigating these vulnerabilities, we can enhance the security and reliability of smart contracts, thereby ensuring the integrity of blockchain transactions and maintaining trust in the blockchain ecosystem.

7.2.1 Vulnerabilities impact on the security of smart contracts

The flaws inherent in smart contracts have a significant impact on their security, particularly in terms of privacy issues. The research results indicate that a primary problem for developers is maintaining the confidentiality of smart contracts. Maintaining the confidentiality of essential operations and data while utilizing cryptographic measures to prevent unauthorized access is a crucial issue that must be addressed [9].

Research has shown that the current privacy standards on platforms like Ethereum are inadequate for supplanting conventional paper contracts [111]. This is because confidential information included in these agreements may be compromised if not properly managed. Due to the significant risks associated with the lack of transactional privacy and inadequate data feed privacy, there is an urgent need for the development of sophisticated techniques that safeguard privacy in smart contract applications.

The vulnerability related to timestamp Dependency [1] has been identified as one of the most critical deficiencies in Ethereum smart contracts. This vulnerability enables malicious actors to exploit transaction timing, potentially allowing them to manipulate the system.

A significant danger pertains to gas expenses [34], particularly in unoptimized smart contracts that need an excessive quantity of gas for optimal operation. Research has

identified some gas-intensive behaviors that may lead to inefficient contract execution. Research indicates that over eighty percent of assessed smart contracts exhibit significant inefficiencies. The recognition of these patterns has been facilitated by the creation of tools like Gasper, which allows developers to optimize their contracts and minimize unnecessary gas use. Addressing this problem would enhance the efficiency of smart contracts and bolster their security by reducing the probability of denial-of-service attacks resulting from elevated gas costs [231].

In conclusion, the challenge of integrating Oracles, which are regarded as external data sources, within smart contracts introduces an additional degree of risk. These vulnerabilities underscore the need for ongoing research and development in smart contract security to ensure the creation of reliable and efficient blockchain applications.

Surveys	**Data Source**	**Models**	**Vulnerability Types**	**PRISMA**	**Tools**	**Static**	**Dynamic**	**Hybrid**
Shiri et al. [187]	✓	✓	✓	✗	✓	✓	✓	✓
Janaka et al. [179]	✓	✓	✗	✓	✓	✓	✓	✓
Kiani et al. [105]	✓	✓	✓	✓	✓	✓	✗	✓
Praitheeshan et al. [165]	✗	✗	✓	✗	✓	✓	✓	✗
Alharby et al. [9]	✓	✗	✓	✗	✓	✓	✗	✗
De et al. [47]	✓	✓	✓	✗	✓	✗	✗	✓
luliano et al. [90]	✓	✗	✓	✗	✗	✓	✓	✗
Kushwaha et al. [118]	✓	✗	✓	✗	✓	✓	✓	✓
Kushwaha et al. [119]	✓	✗	✓	✗	✓	✓	✓	✗
Khan et al. [103]	✓	✗	✓	✗	✓	✓	✓	✗

Table 1 Comparison across existing reviews or surveys

8 Vulnerability Detection Techniques

This section examines prominent security analysis methods used to identify vulnerabilities in smart contracts. Most tools are primarily used for static, dynamic, and hybrid analyses of smart contract codes, categorized under several Deep Learning (DL) and Machine Learning (ML) methodologies.

8.1 Deep Learning Approaches

Current studies on smart contracts have yielded several review articles, each presenting a distinct perspective on security. These articles examine the safety aspects of smart contracts from multiple perspectives, investigating emerging security patterns, evolving threats, and innovative protection methods. The combination of these diverse perspectives enhances our contribution and serves as a significant asset in navigating the intricate realm of smart contract security. These contributions play a crucial role in guiding continuous research and development efforts, which in turn improve smart contracts against potential vulnerabilities in the ever-changing blockchain landscape.

Yang et al. [224] utilize Python to create models of smart contracts and employ SCLMF to detect six different vulnerabilities. The experiments conducted on WScrawlD and Omniglot demonstrate that SCLMF exhibits a notable detection capability, achieving detection accuracies of 96.7% and 98.5% under 5-way 1-shot and 5-way 5-shot conditions, respectively. In general, these methodologies effectively demonstrate their ability to identify vulnerabilities in Ethereum smart contracts.

Deng et al. [48] developed a technique for identifying four different vulnerabilities in smart contracts by employing multimodal feature fusion. The approach retrieves three modal characteristics from the life cycle of intelligent contracts. It employs advanced machine learning models such as Graph Convolutional Networks and bidirectional Long Short-Term Memory networks. The experimental findings improved detection accuracy levels of 85.73% for reentrancy vulnerabilities, 85.41% for timestamp dependence

vulnerabilities, 83.58% for tx.origin vulnerabilities, and 90.96% for integer overflow vulnerabilities. Researchers have evaluated the efficacy of five reentrancy detection tools, including Mythril and Sailfish, in identifying reentrancy vulnerabilities in smart contracts.

In the study, **Zheng et al.** [242] discovered that more than 99.8% of reentrant contracts were identified as false positives. They identified eight specific explanations for this phenomenon. The tools only identified reentrancy vulnerabilities about call.value(), and Ethereum's official IDE, Remix, found 58.8% of these errors. In the study, **Huang et al.** [86] analyzed historical advancements in smart contract security across four development phases: 1) security designing, 2) privacy implementation, 3) testing prior to deployment, and 4) tracking and evaluation. They identified 10 vulnerabilities as a result of their 2019 examination. Lastly, it delineated the forthcoming obstacles and prospects in smart contract security for blockchain developers and researchers.

Tang et al. [199] introduces Lightning Cat, a deep learning framework that employs enhanced deep learning architecture; Optimized-CodeBERT, Optimized-LSTM, and Optimized-CNN to scrutinize and distill critical characteristics from vulnerable smart contract code segments. The proficiency of Lightning Cat was assessed against the SolidiFI benchmark dataset, where it was found that the Optimized-CodeBERT configuration achieved a high F1-score of 93.53%, indicating superior performance compared to alternative methodologies. This outcome underscores the potential of utilizing deep learning for more nuanced and sophisticated intelligent contract vulnerability detection, suggesting that such technologies may eclipse traditional static analysis in effectiveness.

A study by **Qian et al.** [167] which discusses the use of a BLSTM-ATT model to effectively learn and detect vulnerabilities in smart contracts at the code and snippet level. The study focused on the reentrancy vulnerability and demonstrated promising results through extensive experiments on a dataset of real-world smart contracts. This work exemplifies the application of deep learning techniques to the area of smart

contract vulnerability detection and suggests a pathway for future research to extend these techniques to other types of vulnerabilities, such as integer overflow and unhandled exceptions. However, leveraging advanced machine learning techniques, such as BLSTM networks, to detect security vulnerabilities is indeed a direction that has been pursued in related research.

Zheng, et al. developed the CBGRU model in [242] which boasts higher accuracy in identifying a variety of smart contract vulnerabilities than previous models. However, it cannot specify the vulnerability types in contracts with multiple issues and is less accurate when features of vulnerabilities are not prominent.

Yu et al. [227] introduced the TxMirror framework, which adopts a data-driven technique to detect vulnerabilities in smart contracts bytecode by symmetric transaction simulation. It modifies the Ethereum Virtual Machine to trace the interdependencies of stack data using a double-link forest structure. However, TxMirror can only detect attacks and vulnerabilities that occur during the smart contract's execution, excluding those that happen before its initiation, such as the short address attack —a limitation shared by transaction-based detection methods.

Artemis by **Wang et al.**[208] is a refined tool for verifying smart contracts, aimed at mitigating security risks in blockchain applications. In contrast to current tools, Artemis is capable of identifying a broader spectrum of vulnerabilities, such as greedy, block information reliance, gasless send, and perilous delegatecall. The program underwent thorough testing on 12,899 smart contracts, demonstrating its efficacy and speed in identifying vulnerabilities. The findings demonstrate that Artemis is both accurate and economical for practical use.

Deng et al. [33] proposed a symbolic execution-based tool, DefectChecker, which detects contract defects in Ethereum smart contracts, achieving a high F-score of 88.8% on an open-source dataset. It can identify eight defects in the bytecode, providing developers with a way to check the security of smart contracts even without

access to the source code. Furthermore, integrating an SMT Solver can further enhance DefectChecker's capabilities in identifying contract defects, although this may lead to increased analysis time and complexity.

Linh et al. [130] presents a methodology using an Imaging Graph Neural Network with a Defined Pattern to identify vulnerabilities in smart contracts. The model transforms integrated graph characteristics and security patterns into grayscale pictures, training to identify reentrancy, timestamp dependency, and endless loop vulnerabilities. The model's performance is assessed utilizing seventeen techniques, including cutting-edge approaches and neural network-based models.

Srinivas et al. [193] seeks to categorize vulnerabilities in Ethereum smart contracts by feature extraction utilizing machine learning techniques. The dataset was created using pixel values from photos and trigram feature extraction. The Naive Bayes technique surpassed other models in F1-score, attaining values of 99.38% and 99.44% with Binary Relevance and Classifier Chain, respectively. The Random Forest model demonstrated significant performance, with F1-scores of 96.71% and 96.61% with Binary Relevance and Classifier Chain, respectively. In comparison to the inefficient techniques MLkNN and BRkNN, the TriPix dataset surpasses models used in opcode characteristics or image-based detection. This indicates that the TriPix dataset surpasses the models used in earlier studies.

Yuan et al. [228] presents SVChecker, a technique for extracting code snippets from Solidity source code and categorizing them as harmful or benign. It employs a neural network with a transformer encoder and a multi-head self-attention mechanism. The study evaluates the effectiveness of SVChecker relative to conventional methodologies and current tools, concluding that it attains superior accuracy in vulnerability identification. The authors provide VSCL, a system for automated detection via bytecode analysis and a metric learning-based deep neural network. The TCN-based model surpasses previous efforts, achieving notable performance parameters of 95.95% precision,

93.73% recall, 94.83% F1 score, and 94.77% accuracy. EtherSolve, as discussed by the authors **Contro, et al.** [40] is a static analysis algorithm based on the symbolic execution of the Ethereum operand stack that allows the resolution jumps in Ethereum bytecode and the construction of an accurate control-flow graph (CFG) of the compiled smart contracts.

Huang et al. [84] diverge from traditional static and dynamic analyses for detecting bugs in Ethereum smart contracts by not focusing on feature extraction. Instead, they conduct a comprehensive examination of potential attacks, convert Solidity bytecode into an RGB color code, and then represent it as a fixed-size encoded image. These images are then processed by a convolutional neural network, which automates feature learning to detect compiler bugs. However, this innovative approach has its limitations and can detect other main types of vulnerabilities in smart contracts.

Wang et al. [141], as mentioned in this paper, proposed a static method for vulnerability detection based on deep learning, which first disassembles Ethereum smart contracts into opcode sequences and then converts the vulnerability detection problem into a natural language text classification problem.

In 2022, **Hwang et al.** [87] announced a novel approach for detecting vulnerabilities in smart contracts. This method, known as CodeNet, employs a novel convolutional neural network (CNN) design. CodeNet involves a data pre-processing step in which a smart contract is converted into an image while preserving its local characteristics. Based on the assessment findings, the suggested CodeNet demonstrated much higher speed compared to the current leading approaches. CodeNet outperformed state-of-the-art smart contract vulnerability detection solutions in terms of detection performance and detection time across different vulnerability categories. Furthermore, it demonstrated superior performance in terms of accuracy, recall, and F1-score, achieving scores of 98.79%, 98.26%, and 98.75%, respectively, on average. Additionally, the CodeNet operates within a time frame of a few hundred milliseconds, making it far quicker than

other comparable tools by a factor of at least several dozen. The technique yielded excellent outcomes, but it has certain limitations in discovering alternative vulnerabilities and performing bytecode analysis.

To the best of our knowledge, reentrancy attacks have not been thoroughly examined, highlighting an opportunity for researchers to contribute significantly to smart contract security. The related literature suggests generic techniques for identifying smart contract weaknesses. However, as noted in the literature, information loss may occur when transforming sophisticated bytecode into images, and the transformation may capture data relevant only to a particular attack. Therefore, careful preprocessing and feature engineering may be necessary to extract meaningful characteristics from bytecode images and accurately represent the bytecode.

Ref	**Year**	**Methodology**	**DL/ML model**	**Acc and F1 score**
Linh et al. [130]	2023	Source code graph features and security patternsto-Gray image	CNN	95.78% 93.86%
Cheng et al. [243]	2021	Bytecode-to-Gray images	CNN	89.52% 93.96%
Hwang et al. [87]	2022	Bytecode-toimage	non-stride CNN	98.27% 98.24%
Li et al. [124]	2023	Smart contractto-gray image	VGG16 and GRU	F1 score= 95%
Srinivas et al. [193]	2023	Opcode-toimages	MLkNN, BRkNN, RF, NB	F1 score= 99.44%
Huang et al. [85]	2018	Bytecode-to-RGB image	CNN	Acc= 97.39%
Qin et al. [170]	2021	Opcode-toimage	ResNet50	F1 score= 86.04%

8.2 Machine Learning Approaches

In recent years, machine learning (ML) methodologies for vulnerability identification have gained significant prominence, particularly in the domains of smart contracts and Internet of Things (IoT) applications. This study aims to provide an exhaustive review of machine learning-based vulnerability detectors, beginning with an examination of fundamental topics, including the Ethereum blockchain, the Ethereum Virtual Machine,

and the Solidity programming language. We examine many vulnerabilities that may occur in smart contracts, accompanied by illustrative code snippets. Furthermore, we provide substantial machine learning models and methodologies to clarify the operation of these vulnerability detectors. Additionally, we examine vulnerability identification methodologies grounded in formal techniques, which are used either for dataset creation in machine learning or as benchmarks for comparative analysis. We aim to provide readers with the essential knowledge required to assess the methodologies discussed in this survey, encompassing critical elements of the Ethereum ecosystem, common vulnerabilities, prominent machine learning models, and static analysis techniques and datasets cited in the chosen literature.

Abri et al. [3] examine the problem of vulnerabilities in smart contracts, which have led to considerable disturbances in the blockchain domain. It highlights the inconsistency in the definition of these vulnerabilities and suggests an integrated detection technique using a Multi-layer Perceptron (MLP). The authors use feature vectors derived from Opcodes and Control Flow Graphs (CFG) to train the machine learning model. They provide a common pre-processing technique and establish a balanced collection of defective files across various Solidity versions. The research evaluates various machine learning models, including Random Forest, XGBoost, and Support Vector Machine, in comparison to the proposed Multi-Layer Perceptron (MLP) and a hybrid MLP-Long Short-Term Memory (LSTM) model. The findings indicate a remarkable accuracy of 91% with the MLP model, with the lowest average False Positive Rate of 0.0125. This study substantially advances the industry by standardizing vulnerability detection using the Smart Contract Weakness Classification registry.

The manuscript presented by **Ghamry et al.** [58] enhanced machine learning methodology for identifying malware in IoT applications. It utilizes visual representations of network traffic to enhance detection precision. An ant colony optimizer (ACO) is used for feature selection, minimizing the feature set while improving the performance of a

support vector machine (SVM) classifier. The particle swarm optimization (PSO) technique is used to optimize SVM parameters for various kernel functions. Experimental findings indicate that the SVM with a linear kernel attains a high accuracy of 95.56% along with other performance measures. The results indicate that bio-inspired methodologies such as Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) may efficiently create a lightweight malware detection system for the Internet of Things (IoT). The paper emphasizes that bio-inspired methodologies such as Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) may efficiently develop a lightweight machine learning-based malware detection system specifically designed for IoT contexts.

Chen et al. [33] introduces Dynamit, a dynamic system designed to discover vulnerabilities in Ethereum smart contracts. Dynamit identifies insecure smart con- tracts by categorizing malicious transactions inside a blockchain through the use of machine learning on transactional information. The framework achieves a 94% level of accuracy when analyzing a dataset consisting of 105 transactions. By utilizing machine learning, one may discover vulnerabilities without the need to create rules manually. Instead, the machine learning algorithms may depend on their capacity to acquire knowledge and identify vulnerabilities.

Eth2Vec is a machine-learning-based static analysis tool designed by **Ashizawa et al.** [12] to find vulnerabilities in smart contracts, proficient at spotting flaws even in modified code. In contrast to current solutions that depend on manually crafted features, Eth2Vec autonomously derives features from susceptible Ethereum Virtual Machine (EVM) bytecodes via a neural network methodology. It evaluates code similarities between target EVM bytecodes and those it has already acquired. Experimental findings using open datasets, such as Etherscan, indicate that Eth2Vec outperforms a contemporary support vector machine model in critical performance parameters, including accuracy, recall, and F1-score.

[236] has devised a comprehensive feature extraction methodology for vulnerability detection utilizing abstract syntax trees, opcodes, and basic blocks of control flow graphs. The researchers employed the Smartbugs and SolidiFibenchmark datasets alongside six machine learning algorithms to perform comprehensive experiments. Their model, which derived 1,619-dimensional bigram characteristics from simplified operation codes, attained over 96% in predicting recall and precision. The model demonstrated commendable time efficiency, averaging a detection time of 4 seconds per contract. The research presented a classification methodology employing various machine learning models, with the TriPix dataset surpassing all other algorithms, achieving F1-scores of 99.38% and 99.44%.

ContractWard is a machine learning methodology presented by **Wang et al.** [211] that extracts bigram characteristics from reduced operation codes of smart contracts. They use five machine learning algorithms and two sampling methods to construct their models. ContractWard, assessed on 49,502 actual smart contracts, attains over 96% predicted Micro-F1 and Macro-F1 scores, with an average detection time of just 4 seconds per contract using XGBoost and SMOTETomek for model training and data balancing, respectively.

Many current systems utilize various methods to identify different vulnerabilities. Some employ solidity code to detect vulnerabilities, while specific articles have identified multiple vulnerabilities in smart contracts. However, a significant challenge remains in achieving a high level of accuracy in their identification.

Furthermore, the effectiveness of detecting errors is hindered by the limits of current detection methods, which rely on static analysis or have poor test coverage. To address these issues, it is essential to develop advanced detection methods that leverage dynamic analysis, machine learning algorithms, and comprehensive test suites. This will improve the precision and effectiveness of identifying vulnerabilities in smart contracts.

Existing Techniques

The table 2 provides a thorough literature evaluation of multiple studies aimed at identifying vulnerabilities in smart contracts, emphasizing their techniques, detection capabilities, and accuracy. Each entry specifies the authors, the particular approaches utilized spanning Python implementations, deep learning frameworks, symbolic execution, and multimodal feature fusion and the categories of vulnerabilities they seek to detect. The table succinctly encapsulates the cutting-edge methodologies in smart contract vulnerability identification, highlighting the variety of techniques and their corresponding performance results.

Vulnerabilities in existing tools

Our investigation delves into the domain of smart contract vulnerabilities, highlighting the need to enhance the security of blockchain technology. According to the results of our exhaustive investigation, we have discovered a wide range of vulnerabilities. Among these vulnerabilities, the reentrancy and timestamp dependence vulnerabilities have emerged as the most prevalent as represented in table 3, accounting for approximately half of our findings. These findings provide evidence that our detection approach is successful in identifying a wide variety of serious vulnerabilities in smart contracts.

Tools	**Vulnerabilities**
Hwang et al. [87]	Reentrancy, Unchecked low-level calls, Tx.origin and Timestamp dependency
Li et al. [124]	Reentrancy, Timestamp dependency, and Infinite loop
ke et al. [243]	Reentrancy
Qian et al. [169]	Reentrancy, Integer overflow underflow, TOD, Timestamp dependency
Tahir et al. [198]	Reentrancy

Tools	Vulnerabilities
Chen et al. [33]	Transaction State Dependency, DoS Under External Influence, Strict Balance Equality, Reentrancy, Nested Call, Greedy Contract, Unchecked External Calls, Block Info Dependency
Filippo et al. [40]	Reentrancy
Deng et al. [48]	ARTHM, Reentrancy, locked ether, TimeO
Kim et al. [107]	Reentrancy
Yang et al. [224]	Reentrancy, Unchecked LL calls, Tx.Origin, Timestamp dependency
Rongwei et al. [227]	transaction-ordering dependence, timestamp dependence, reentrancy, integer overflow, suicidal, unchecked call, unsecured balance
Hang et al. [232]	Callstack Depth Attack, Integer Overflow, Integer Underflow, Reentrancy, Timestamp Dependency, Infinite Loop
Ashizawa et al. [12]	Reentrancy, Time Dependency, ERC-20 Transfer, Gas Consumption, Implicit Visibility, Integer Overflow, Integer Underflow
Chinen et al. [37]	Reentrancy
Fu et al. [66]	Reentrancy, Integer overflow-underflow, Suicide, TOD, predictable variable dependency, mishandled exception, unchecked return values
Bo et al. [93]	Gasless Send, Exception Disorder, Reentrancy, Timestamp Dependency, Block Number Dependency, Dangerous Delegatecall, Freezing Ether
Qian et al. [167]	Reentrancy

Tools	Vulnerabilities
Singhal et al. [192]	Reentrancy
Sun et al. [196]	Reentrancy, Arithmetic Issues, Time manipulation
Torres et al. [204]	Reentrancy
Yuan et al. [228]	Reentrancy, Arithmetic, Time manipulation, Access control , Unchecked Low-level call , Front running
Wang et al. [208]	Freezing Ether, Block Info Dependency, Gasless Send, Dangerous Delegatecall
Zhang et al. [235]	Reentrancy, Timestamp dependency, Block info dependency, Tx.Origin

Table 3: Vulnerabilities detected in recent studies

Currently, the proposed techniques for detecting vulnerabilities in smart contracts include static, dynamic, and hybrid analyses, based on four distinct types of tools.

8.3 Static analysis

Static analysis, a technique used to examine the source code or bytecode of smart contracts without executing them, may be utilized to detect vulnerabilities within these contracts. Code review, model building representation, symbolic execution, and data flow analysis are prevalent methodologies used to identify issues such as reentrancy vulnerabilities and uninitialized variables. The dynamic characteristics of smart contracts render static analysis incapable of identifying vulnerabilities that arise during execution, particularly those triggered by external input. Although static analysis [217] can be automated and effectively identify vulnerabilities in static code, it is limited by its inability to rectify these errors.

Chinen et al. [37] presents RA (Reentrancy Analyzer), a static analysis tool intended

Study	Methodology	Detection Capability	Detection Accuracies
Yang et al. [224]	Employ Python and SCLMF for detecting six vulnerabilities	Detecting six vulnerabilities	96.7% under 5-way 1-shot and 98.5% under 5-way 5-shot
Deng et al. [48]	Use multimodal feature fusion with Graph Convolutional Networks and bidirectional LSTM networks	Detecting four different vulnerabilities	85.73% reentrancy, 85.41% timestamp dependence, 83.58% tx.origin vulnerabilities, and 90.96% integer overflow
Zheng et al. [242]	Evaluate reentrancy detection tools including Mythril and Sailfish	only identify reentrancy vulnerabilities	99.8%
Huang et al. [86]	Analyze historical advancements in smart contract security	Identify 10 vulnerabilities	N/A
Abri et al. [3]	Dynamit, a dynamic system	Analyzing a data set consisting of 105 transactions.	Attains a 94% level of accuracy
Tang et al. [199]	Lightning Cat, a deep learning framework		Achieved a high f1-score of 93.53%
Qian et al. [167]	Use a BLSTM-ATT model to learn and detect vulnerabilities in smart contracts at the code level	Demonstrated promising results	Extensive experiments on a dataset of real-world smart contracts
Wang et al. [141]	Propose a static method for vulnerability detection based on deep learning	Converts bytecode into opcodes and then images	N/A
Zhang et al. [242]	Develop the CBGRU model for identifying smart contract vulnerabilities	Boasts higher accuracy than previous models	N/A
Yu et al. [227]	Introduce the TxMirror framework for detecting vulnerabilities in smart contract bytecode	Uses symmetric transaction simulation	N/A
Chen et al. [33]	Propose DefectChecker, a symbolic execution-based tool for detecting contract defects	Achieves a high F-score of 88.8%	On an open-source dataset
Filippo et al. [40]	Develop EtherSolve, a static analysis algorithm based on symbolic execution of Ethereum bytecode	Resolves jumps in Ethereum bytecode	Constructs an accurate control-flow graph
Huang et al. [84]	Utilize an image-based approach with CNNs to detect compiler bugs in Solidity bytecode	Automates feature learning to detect bugs	N/A
Hwang et al. [87]	Propose CodeNet, a CNN design for detecting vulnerabilities in smart contracts	Converts smart contracts into images	Outperformed state-of-the-art solutions in terms of performance and detection time
Tahir et al. [198]	Introduce an approach for identifying reentrancy vulnerabilities using deep learning and images	Converts smart contracts into opcodes and images	Achieves a precision rate of 98.10%

Table 2 Existing approaches detail

to evaluate the susceptibility of Ethereum smart contracts to re-entrancy attacks. In contrast to current dynamic analysis techniques that necessitate Ether expenditure and prior familiarity with attack patterns, RA employs symbolic execution and equivalence checking via a satisfiability modulo theories solver, enabling the examination of inter-contract interactions solely through Ethereum Virtual Machine bytecodes. RA may detect vulnerabilities without executing the contracts and ensure the absence of false positives or negatives. The execution of RA illustrates its efficacy in precisely identifying smart contracts vulnerable to re-entrancy attacks.

Eth2Vec [12] is a machine-learning static analysis tool that autonomously derives features from susceptible Ethereum Virtual Machine (EVM) bytecodes, outperforming contemporary support vector machine models in accuracy, recall, and F1-score, even in modified code.

Slither is a static analysis framework presented by **Feist at al.** [62] that transforms Solidity smart contracts into an intermediate representation known as SlithIR, maintaining semantic information and enabling the use of program analysis methods such as dataflow and taint tracking. The primary applications include automatic vulnerability detection, identification of code optimization possibilities, user comprehension of contracts, and facilitation of code reviews. The framework's bug identification is rapid, precise, and surpasses existing static analysis tools in speed, robustness, and the equilibrium between detection and false positives. The research evaluated instruments using an extensive dataset of smart contracts.

Securify [206] is a scalable static analysis tool and fully automated security analyzer for Ethereum smart contracts that assesses whether contract behaviors are safe or unsafe based on specified properties. Its analysis involves two main steps: first, it symbolically analyzes the contract's dependency graph to extract detailed semantic information, and second, it checks compliance and violation patterns using a domain-specific language to determine if a property holds. Securify has been publicly released and has

analyzed over 18,000 contracts, making it a valuable tool for security audits by experts. Its effectiveness in proving the correctness of smart contracts and identifying critical violations has been extensively evaluated using real-world contracts.

Oyente [7] is a static analysis tool that employs symbolic execution to examine EVM bytecode without necessitating high-level languages such as Solidity. It can detect multiple vulnerabilities, including Time of Death (TOD) weaknesses, predictable random number generation, reentrancy vulnerabilities, and exception-handling deficiencies, while accommodating the majority of EVM opcodes. Nevertheless, Oyente encounters difficulties in deducing developers' intentions from bytecode owing to insufficient contextual information, like variable types, and the redundancy of bytecode between function invocations. As a result, it is unable to adequately address fairness and accuracy concerns, such as integer overflow.

Traditional static analysis methods primarily focus on examining source code and bytecode. Consequently, they often generate a considerable number of false positives, as they fail to consider the numerous complexities inherent in the implementation of smart contracts. The principal methodologies used in static analysis are formal verification, symbolic execution, and intermediate representation. Prominent tools used in this domain are Scurify, SmartCheck, Eth2Vec [12], ContractWard [211], RA [37], Mythril, and Slither [62].

Dynamic analysis

Dynamic analysis is a method of runtime monitoring and evaluation used to detect security flaws and vulnerabilities in systems, particularly in smart contracts. Fuzz testing is an effective method used in dynamic analysis. [126] generates several normal and abnormal test scenarios to assess the application under evaluation. It may discover vulnerabilities that static analysis may miss, including unusual state changes and time-dependent attacks. This approach demonstrates superior scalability and can

identify deficiencies.

The teEther [115] dynamic analysis tool evaluates Ethereum EVM contracts through symbolic execution and result verification. It detects the absence of permission verification, permitting uncontrolled access. The analytical procedure entails generating a Centrifuge token, scrutinizing essential instructions such as DELEGATECALL, SELFDESTRUCT, and SSTORE, exploring pathways to these instructions, and addressing constraints through diverse methodologies to identify vulnerabilities. This instrument is vital for guaranteeing secure and effective contract administration.

Ashouri et al. [13] presents Etherolic, a resilient, scalable, and effective fuzzing tool for security assessment of smart contracts, namely those created for the Ethereum platform. The tool employs dynamic taint tracking and concolic testing to examine the bytecode of smart contracts operating on the Ethereum Virtual Machine. Etherolic can detect vulnerabilities and create exploits to activate unidentified code problems. The article assessed Etherolic using a custom benchmark suite, uncovering 204 security breaches, hence demonstrating its efficacy in smart contract security analysis.

Mythril [184] employs tactics that are emblematic [83]. Upon decompiling the EVM bytecode, Mythril then initializes the state of the contract account. It subsequently utilizes a substantial volume of transactions to examine the state space of the contract. On the occasion that an unfavorable condition is uncovered. Mythril identifies the transactions necessary to achieve a vulnerability state when one is detected, hence validating the existence of the vulnerability.

Fuzz testing utilizes application binary interfaces (ABIs) to provide test inputs and observe the execution of smart contracts on systems such as Ethereum. This is exceptionally accurate when testing is conducted using tools like ContractFuzzer [93]. One of ContractFuzzer's most notable advantages is its capacity to detect a wide range of vulnerabilities while maintaining a lower false positive rate compared to other tools. The complexity of the contract's state space and execution pathways may provide

challenges for dynamic analysis. Dynamic analysis, especially via fuzz testing, serves as an adjunct to static analysis, augmenting the identification of security vulnerabilities in software systems. Fuzz testing such as Artemis [208] exemplifies this concept.

Hybrid analysis

The integration of multiple network models into a hybrid analysis enables the utilization of each model's unique strengths to achieve superior outcomes. An exemplary hybrid learning model is the CBGRU model [232], which utilizes diverse network capabilities to provide enhanced results. Hybrid models have shown considerable success across several areas. Conkas [214] is a tool that identifies vulnerabilities in Solidity contracts by examining the bytecode or Solidity code of the smart contract. It employs dynamic and static analysis techniques to detect potential security vulnerabilities like as re-entrancy and integer overflow. Conkas can function autonomously or be included into an existing development pipeline, enabling smart contract developers to detect and rectify any errors through a comprehensible report format.

To provide a precise air quality prediction, Du et al. [55] integrated one-dimensional convolutional neural networks (1D-CNNs) with bidirectional long short-term memory networks (Bi-LSTMs). This enabled the use of CNNs for feature extraction and Bi-LSTMs for learning feature correlations. Furthermore, Yue W et al. [229] proved that their hybrid model surpassed single-structured networks by integrating Word2vec, BiLSTM, and CNN for short text classification. Hybrid approaches are often demonstrating more success than traditional single-network methods across many applications.

8.3.1 Tools

Researchers and developers are increasingly focusing on the security concerns associated with smart contracts, employing various methods to rigorously analyze Ether smart contracts and EVM bytecodes to prevent exploitation by malicious actors [194]. We

categorized the analysis into four distinct types, see fig 5: code analysis tools, model-building tools, data tools, and symbolic analysis tools.

1. **Code Analysis Tools:** A variety of tools for code analysis in Ethereum smart contracts have been created to improve vulnerability identification and code similarity assessment. Eth2Vec, a static analysis approach [12] utilizes the PV-DM model and EVM Extractor to detect code clones and vulnerabilities via the analysis of EVM bytecodes and the use of existing vulnerability data. Another approach incorporates deep learning and multimodal decision fusion based on hybrid analysis [48] to enhance detection rates for various vulnerabilities by amalgamating diverse code representations. CodeNet [87] employs a novel methodology by converting smart contracts into images using a code-specific CNN, enabling vulnerability detection while preserving semantic integrity. A systematic approach based on static analysis [107] automates the evaluation of innovative contract analyzers using varied test scenarios, revealing false positives in current tools. HGAT [141] employs a hierarchical graph attention network to transform functions into code graphs, hence improving feature extraction for vulnerability identification. Finally, improvements to Mythril's as a hybrid analysis tool internal module have augmented its operating efficiency and streamlined opcode analysis for automated vulnerability discovery.
2. **Model Building Tools:** Model analysis for smart contracts employs diverse, specialized techniques to detect weaknesses and enhance security. One method includes dynamic analysis [194] creating finite state machine models that depict contract behavior, which are then evaluated using model testing tools to identify defects. Advanced approaches, including deep learning technologies such as LSTM networks, enhance the precision of vulnerability identification via the analysis of operational code. BLSTM-ATT deep learning model based on static analysis [167] proficiently identify reentrancy flaws proficiently, exhibiting enhanced performance through the examination of contract snippets from an extensive dataset exceeding

42,000 contracts. These solutions collectively aim to enhance security, analyze runtime execution and protect user assets within the blockchain ecosystem .

3. **Data Tools:** Diverse technologies have been developed for data analysis related to Ethereum smart contracts, each employing unique approaches to enhance vulnerability identification and code optimization. SlithIR [62] employs Static Single Assignment (SSA) form for program analysis, concentrating on dataflow and taint tracking to enhance vulnerability identification and code optimization. Colored Petri nets are used by one tool [76] to model smart contracts, focusing on data and control flow to identify reentrancy vulnerabilities via hierarchical modeling and simulations. An approach using deep learning for data preparation [192] has achieved excellent accuracy in identifying reentrancy attacks. ÆGIS [204] is a dynamic analysis tool that safeguards smart contracts against runtime exploitation by detecting vulnerabilities using customized attack patterns and data flow analysis. TxMirror [227] dynamically operates at the bytecode level, detecting vulnerabilities through transaction simulation and an efficient dependency storage mechanism. Finally, ASGVulDetector and BASGVulDetector [235] use abstract semantic graphs and graph neural networks for enhanced static analysis, exhibiting higher efficacy in vulnerability detection relative to conventional techniques.
4. **Symbolic Analysis Tools:** A variety of tools have been developed for the symbolic analysis of Ethereum smart contracts, each presenting distinct characteristics and functionalities. A machine learning technique, as emphasized in reference [196], augments conventional symbolic analysis by increasing precision and reducing human involvement in vulnerability identification. Artemis [208] is a sophisticated symbolic analysis fuzzing tool that accurately detects various vulnerabilities in 12,899 contracts with remarkable accuracy and cost efficiency. Another technique improves dynamic symbolic execution [66] by including data flow and taint analysis, highlighting crucial channels for enhanced vulnerability detection. ContractWard

[211] statically utilizes machine learning to attain over 96% accuracy in vulnerability discovery within only four seconds, exceeding traditional symbolic analysis methods. MAIAN [156] is a tool that employs symbolic execution to examine Ethereum EVM contracts, detecting security vulnerabilities such as suicidal contracts and frozen tokens by examining execution paths and testing contracts with real transactions. Furthermore, RA [37] based on static analysis employs symbolic control flow graphs (CFGs) and the Z3 SMT solver, comprising three integrated modules: CFManager, VM, and Verifier, for analysis.

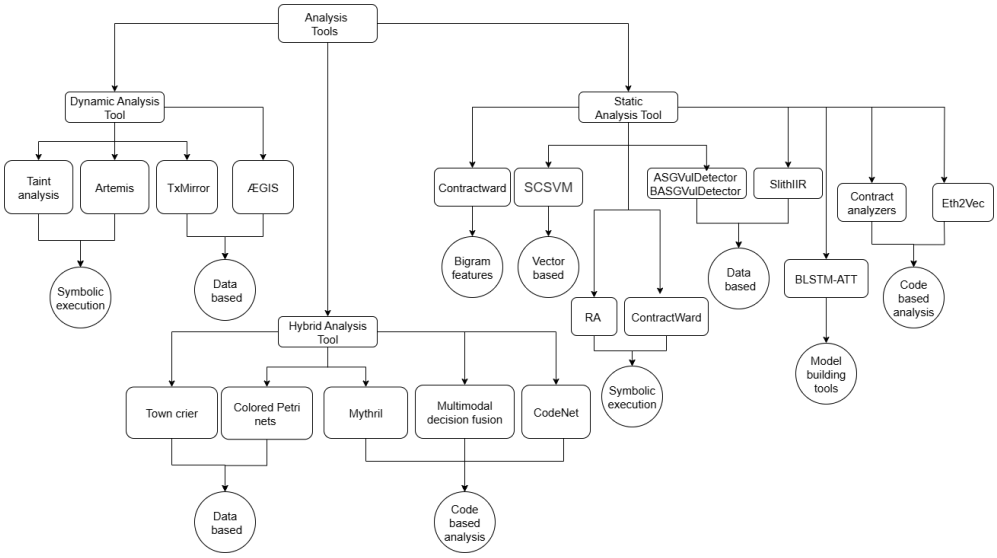

Fig. 5 Tools

9 Artificial Intelligence

Artificial intelligence (AI) is a discipline dedicated to developing intelligent computers that can perform tasks typically requiring human intelligence [25]. Machine learning (ML) is a subset of artificial intelligence (AI) that concentrates on creating systems capable of learning from data and making decisions autonomously, without explicit

programming. Conventional programming methodologies necessitate the manual coding of solutions for individual issues, demanding considerable human effort and constrained by the programmer's capacity to anticipate all potential scenarios. Conversely, machine learning and artificial intelligence instruct computers using extensive datasets, enabling them to discern patterns and generate predictions, hence enhancing their adaptability to novel data and circumstances.

The development of AI and ML models encompasses several essential stages:

- Data Collection: Assembling an extensive dataset pertinent to the issue at hand.
- Data Preprocessing: Refining and converting the data for optimal usage.
- Model Selection and Training: Selecting and training a suitable machine learning model on the dataset.
- Model Evaluation: Evaluating the model's efficacy through a distinct dataset to ascertain accuracy.
- Deployment: Executing the model within a production setting and assimilating it with other systems.
- Monitoring and Maintenance: Persistently assessing the model's efficacy and implementing requisite modifications.

AI and ML are potent instruments capable of automating numerous tasks and producing very precise forecasts and decisions. They flourish in domains where traditional programming techniques would be excessively time-consuming or inefficient, particularly those characterized by complexity.

9.1 AI for Smart Contract

Recent prominent hacking events concerning smart contracts (SCs) have resulted in substantial financial losses. Nonetheless, the application of AI technology can augment the security of supply chains and reduce the likelihood of such breaches [113]. One method is to utilize AI algorithms to analyze the code for vulnerabilities by

detecting patterns and abnormalities that signify security deficiencies. These algorithms may analyze extensive code bases to identify prevalent vulnerabilities, facilitating enhancements in subsequent iterations of smart contracts.

Natural language processing (NLP) techniques are employed to improve the security of smart contracts (SCs) by detecting ambiguities and inconsistencies in their terminology, hence aiding in the prevention of hacking. A vulnerability detection methodology utilizing hierarchical attention and BiLSTM [167] neural networks was created to enhance contract safety. A larger dataset was developed to identify Ponzi schemes throughout the creation of social networks using a multi-view cascading ensemble model (MulCas) [241].

AI markedly improves the security of smart contracts (SCs) through code analysis, identification of linguistic ambiguities, and facilitation of real-time monitoring. It assists in alleviating hacking vulnerabilities and exploitation. Moreover, blockchain technology can attain enhanced security and transparency via decentralized management systems. With the growing utilization of smart contracts, the significance of AI in safeguarding their security and mitigating fraud will become progressively essential.

AI has emerged as a crucial tool for modeling complex code semantics and enhancing vulnerability detection in smart contracts. Further, it reviews the latest and most significant AI-driven techniques, focusing on machine learning (ML) and deep learning (DL) approaches that utilize various analysis types to improve security measures in

10 Machine Learning

Machine Learning (ML) algorithms enhance their efficacy by examining extensive datasets to identify patterns and trends. This empowers people to make decisions and predict results, so enhancing their performance. They may be categorized into three principal types see table 4: supervised learning, unsupervised learning, and reinforcement learning . Each style of learning possesses distinct advantages and

Type	Description
Supervised Learning	Supervised learning entails using a training dataset including input samples and their matching labeled outputs. For example, forecasting the quantity of active users on a platform may rely on input variables such as sold items or user evaluations. The training set links these inputs with outputs to adjust the parameters of the machine learning model. Upon completion of training, the model is capable of forecasting the target variable for novel input data. Supervised learning is divided into regression problems, which forecast numerical values, and classification tasks, which allocate categorical labels. This differentiation helps in choosing the suitable model for certain tasks.
Unsupervised Learning	Unsupervised learning entails identifying patterns in data devoid of pre-existing classifications. The training data comprises only variables, with the objective of revealing structural information. Essential strategies include clustering, which aggregates pieces with common characteristics, and dimensionality reduction, which streamlines high-dimensional data. An example of unsupervised learning is the use of clustering in electronic marketplaces to categorize clients or markets. This division facilitates more precise communication techniques. Unsupervised learning is essential for uncovering latent patterns inside data.
Reinforcement Learning	Reinforcement learning entails delineating the present state, establishing a goal, and specifying permissible behaviors within limitations, enabling the model to learn by trial and error to optimize rewards. This method has shown efficacy in confined settings such as games. Moreover, reinforcement learning is pertinent to multi-agent systems, including electronic marketplaces. Its success across several sectors underscores its adaptability and capacity for intricate problem-solving.

Table 4 Types of Machine Learning

disadvantages, and their behavior evolves according to the experiences acquired via learning. Machine learning algorithms are utilized throughout several areas, including virus detection, image recognition, and blockchain technology.

In the realm of securing computer systems and networks, viruses and vulnerabilities are critically significant; nonetheless, conventional methods are both laborious and resource-demanding. Machine learning has emerged as a viable solution for discovering malware and vulnerabilities at the opcode level. By analyzing opcode sequences of executable files, machine learning algorithms may identify trends and behaviors indicative of possible vulnerability exploitation or malware. One advantage of machine learning is its ability to rapidly scan extensive datasets, detect threats in real time, and respond to emerging vulnerabilities and infections. This renders it a more robust and scalable security method.

Machine learning has been demonstrated to effectively identify vulnerabilities in several programming languages, including Java, SQL, PHP, and C/C++. It provides real-time identification of several vulnerabilities, which is advantageous for Ethereum smart contracts. This detection method's accuracy and efficiency are highly contingent upon an effective data representation that captures the essential aspects of smart contracts. Insufficient representation may hinder model training, leading to suboptimal generalization across diverse vulnerabilities and contract types. Moreover, to get elevated precision and generalizability in vulnerability detection, it is essential to select appropriate architectures for the models.

The use of machine learning approaches in smart contract vulnerability detection has gained attention in recent years summarized in 5.

The predominant method of ML with DL for analyzing smart contracts involves converting them into Control Flow Graphs (CFG) [40], which visually depict the control flow and different execution paths of programs. Mi et al. [149] provide a method of static analysis by encoding smart contract opcodes inside a control flow graph (CFG) and then converting them into vectors using a depth-first search approach. Subsequently, the vectors are organized into a feature matrix using techniques such as n-gram and term frequency-inverse document frequency (TF-IDF) methodologies. In the last phase, a Deep Neural Network (DNN) functions as a binary classifier to identify weak contracts. Nevertheless, the specific vulnerabilities found by this method remain undisclosed [141].

The generation of code graphs from smart contracts may also be achieved using Control Flow Graphs (CFG) and Abstract Syntax Trees (AST), as shown by the Hierarchical Graph Attention Network (HGAT) model [141]. The inventors of this methodology abstract the nodes of the code subgraph and extract features. They then use a graph attention method to amalgamate these properties into those pertinent to each line of code. Although HGAT is intended to discover four specific kinds of vulnerabilities, it

fails to provide any details on the nature of these vulnerabilities. Instead, it depends on labels from the Smartbugs Wild dataset, which focuses primarily on problematic contracts.

Graph Neural Networks (GNNs) have gained popularity as a machine learning model for vulnerability discovery due to their ability to maintain the structural integrity and interactions inside smart contracts. This is due to their superior ability to spot weaknesses. Research indicates that Graph Neural Networks (GNNs) outperform traditional Natural Language Processing (NLP) methods, especially in contracts characterized by complex logic and structure. Cai et al. [28] present a bidirectional gated graph neural network that integrates a hybrid attention pooling layer. This neural network is designed to gather vulnerability-related information from the graph effectively. The efficacy of GNNs in this domain is shown by their model operating as a binary classifier, trained on several datasets. It particularly tackles nine main categories of vulnerabilities.

ContractWard [211] utilizes the SMOTETomek API to attain equilibrium in training datasets. This is achieved by extracting bigram characteristics from simplified opcodes and using XGBoost to validate the accuracy of various machine learning approaches. To reduce feature dimensionality, they consolidate functionally similar opcodes into a single category. Their approach can predict six distinct types of vulnerabilities with an F1-score of 96%.

xFuzz [221] illustrates that machine learning may be coupled with verification techniques like as fuzzing. This strategy significantly narrows the search region for vulnerable paths. The technique emphasizes reentrancy vulnerabilities while also identifying cross-contract attacks when the number of contracts exceeds two.

Machine Learning effectively identifies weaknesses in smart contracts, whereas Deep Learning, a specialized subset of multi-layered neural networks, amplifies this power.

Deep Learning reveals intricate patterns with great precision by analyzing extensive datasets, facilitated by its architecture of three or more layers.

Author	Methodology	Findings	Limitations
Abri et al. [3]	Tests machine learning and deep learning classifiers for zero-day malware detection. It compares techniques like random forests against deep learning methods and finds that certain conventional methods work effectively.	Achieves 91% accuracy with MLP and a low False Positive Rate of 0.0125	Future research could explore ensemble-based deep learning classifiers and alternative approaches to enhance detection capabilities.
Ghamry et al. [58]	IoT malware detection method utilizes the Ant Colony Optimizer (ACO) for feature selection and Particle Swarm Optimization (PSO) for parameter tuning of the Support Vector Machines (SVM) classifier, which employs various kernel functions	SVM with a linear kernel achieved an accuracy of 95.56%, recall of 96.43%, precision of 94.12%, and F1 score of 95.26%, also ACO and PSO can effectively create a lightweight malware detection system for IoT environments.	Experiments are conducted on a single dataset and do not utilize a deep learning approach for feature selection, also it could involve various image-based datasets and the use of deep learning for feature extraction.

Author	Methodology	Findings	Limitations
Chen et al. [33]	DefectChecker, symbolic execution based approach, analyzes smart contracts' bytecode to identify eight specific defects that can lead to unwanted behaviors.	Analyzes smart contracts' bytecode, has a high F-score, Recall and Precision of 88.8% , 90.9% and 88.3%, also it was applied to 165,621 contracts, revealing 25,815 contained defects classified as impact level 1-3.	DefectChecker has limitations, including false positives and negatives in defect detection, and increasing analysis time. It assumes all paths are reachable, leading to inaccuracies, especially with false conditional expressions. Future work is needed to balance efficiency and accuracy in defect detection.

Author	Methodology	Findings	Limitations
Ashizawa et al. [12]	Detects vulnerabilities in altered code by autonomously learning characteristics from EVM bytecodes using a neural network, instead than depending on manually crafted features, also evaluates code similarity between target bytecodes and previously acquired ones.	Eth2Vec outperforms the SVM-based method in terms of precision, recall, and F1-score.	Due to its training and assessment on open databases, Eth2Vec may not work with different datasets. While it is resilient to code rewrites, its vulnerability detection may be affected by training data quality and variety.
Wang et al. [211]	ContractWard using machine learning extracts bigram features from simplified operation codes and employs various algorithms	achieved over 96% predictive accuracy and an average detection time of 4 seconds per contract	Will explore more effective features to describe the characteristics of smart contracts. Designing anomaly detection models to detect novel vulnerabilities in smart contracts is also being investigated.

Table 5: ML Existing Approaches

11 Deep Learning

The exploration of clustering problems in deep learning builds upon traditional machine learning algorithms by uncovering hidden relationships among dataset variables and creating hierarchical models. Although constructing these models is time-consuming, thorough data analysis enhances categorization accuracy.

In contrast, machine learning techniques provide quick and straightforward models for anomaly detection and rule-based classification. Deep learning approaches, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), can autonomously identify contextual and sequential patterns in code. Additionally, Graph Neural Networks (GNNs) utilize the graph structure of smart contract code to detect vulnerabilities by examining the connections between nodes and edges. Transformer models, particularly those adapted from Natural Language Processing (NLP).

Deep learning (DL) is essential for vulnerability identification in blockchain security because it can scan extensive datasets and discern patterns that may not be apparent using conventional approaches. Deep learning techniques improve threat detection precision, reduce false positives, and adapt to changing attack patterns, making them essential for evaluating the security of smart contracts and cross-chain bridges. The automation of vulnerability identification using deep learning enhances auditing procedures for security professionals, facilitating more efficient detection of possible threats.

Deep learning (DL) methods have gained prominence in the discovery of smart contract vulnerabilities due to their ability to learn hierarchical representations from raw code without requiring human feature engineering. Employing architectures such as convolutional neural networks (CNNs), long short-term memory (LSTM) networks, and attention mechanisms, they effectively capture sequential and contextual dependencies in contract logic. Deep learning models can capture the syntax and semantics of tokenized source code, opcodes, or bytecode via their processing capabilities. Due to their scalability and versatility, they are effective in identifying previously undiscovered

weaknesses and vulnerabilities.

Improved vulnerability assessment in smart contracts can be achieved by utilizing deep learning attention models. These models focus on essential program components, understand context, and adeptly handle complex control flows. By emphasizing code patterns and relationships, they may identify vulnerabilities such as reentrancy and integer overflow issues. Moreover, attention mechanisms improve interpretability, allowing auditors to more effectively understand the rationale for assessing specific code as potentially dangerous. Sequential models, like recurrent neural networks (RNNs) and long short-term memory (LSTMs), effectively capture complex control flows and linkages inside code and execution traces, making them valuable for vulnerability detection. Due to their capacity to evaluate inputs of diverse lengths, track temporal state changes, and predict future security threats based on historical data, they serve as invaluable tools for safeguarding complex contracts [151].

Graph-based deep learning models, such as graph neural networks (GNNs), enhance risk assessment by representing intricate interconnections, control flows, and data flows as graphs. These models facilitate the identification of vulnerabilities, management of inter-contract relationships, and efficient evaluation of complex contracts. Due to the structural connections inherent in smart contracts, graph-based algorithms may detect vulnerabilities and provide comprehensible insights that may aid developers in resolving these issues.

Author	Methodology	Findings	Limitations
Yang et al. [224]	SCSVM, using support vector machine technology constructs representations based on word-to-vector techniques and extracts features for vulnerability identification, also SCLMF, a basic learner-meta-learner framework constructed via Python on Omniglot.	SCSVM, achieved 87.51% accuracy and SCLMF framework demonstrates high detection rates of 96.7% and 98.5% in different conditions, outperforming existing methods.	limited to specific vulnerabilities
Deng et al. [48]	Incorporates code semantics and control structure information using deep learning and multimodal decision fusion, additionally integrates various data types and extracts five features to enhance detection accuracy.	The detection accuracy for arithmetic vulnerability, re-entrant vulnerability, transaction order dependency, and Ethernet locking vulnerability is 91.6%, 90.9%, 94.8%, and 89.5%, respectively, with corresponding AUC values of 0.834, 0.852, 0.886, and 0.825.	Future investigations should examine tiny samples and unsupervised learning to solve this problem, while fulfilling the pressing need for a standardized, unified dataset of smart contract vulnerabilities.

Author	Methodology	Findings	Limitations
Zheng et al. [242]	analyzes 230,548 Etherscan contracts to evaluate smart contract reentrancy detection methods. It shows that five technologies can identify reentrant functions, with 34 true positives and 21,178 false positives out of 21,212 identified contracts.	A study of 230,548 smart contracts reveals existing reentrancy detection tools have over 99.8% false positive rates, identifying only 34 true positives.	ocus on new reentrancy issues beyond traditional patterns like call.value().
Tang et al. [199]	The Lighting Cat solution utilizes deep learning models, specifically Optimized-CodeBERT, Optimized-LSTM, and Optimized-CNN, to detect vulnerabilities without predefined rules.	Optimized-CodeBERT model demonstrated superior performance, achieving an f1-score of 93.53% on the SolidiFI-benchmark dataset.	Enhance the efficacy of the three models in LightingCat and broaden the applicability of our proposed LightingCat to other domains of code security beyond the identification of smart contract vulnerabilities.

Author	Methodology	Findings	Limitations
Qian et al. [167]	Employs a deep learning model, BLSTM-ATT, to effectively detect reentrancy bugs in smart contracts over 42,000 using contract snippet representations.	BLSTM-ATT significantly outperforms state-of-the-art smart contract reentrancyvulnerability detection tools, achieving an FPR of 8.57% and an F1 score of 88.26%. While other tools like Smartcheck and Mythril show much lower TPR and PRE, highlighting the effectiveness of BLSTM-ATT in vulnerability detection.	Limited to detecting reentrancy vulnerabilities in smart contracts.
Zheng et al. [242]	Evaluates five reentrancy detection tools by analyzing 139,424 smart contracts, identifying 21,212 with potential reentrancy issues.	Assesses the efficacy of five reentrancy detection techniques on 139,424 smart contracts, indicating that more than 99.8% of identified reentrant contracts are false positives.	Need for new detection methods beyond those related to call.value(), as existing tools fail to identify real-world reentrancy issues.

Author	Methodology	Findings	Limitations
Yu et al. [227]	TxMirror is a pioneering data-centric solution that symmetrically replicates transactions to identify flaws in smart contract bytecode. It employs a double-linked forest for data storage and user-configurable detection mechanisms to enhance vulnerability identification in the EVM.	TxMirror demonstrated superior efficiency in analyzing transactions, taking only 4.5 hours for 357,293 transactions, significantly faster than TxSpector and Perez's tool. In a direct test, TxMirror processed 10,000 transactions in just 21 seconds, achieving over 400 transactions per second.	TxMirror's limitation is that it can only detect attacks and vulnerabilities occurring during the execution of a smart contract, missing those that occur prior to execution.
Wang et al.[208]	Artemis, identified weaknesses such as greedy, block information reliance, gasless send, and unsafe delegatecall, validated on 12899 smart contracts for vulnerability detection efficacy and efficiency.	Artemis tool is accurate and economical for practical use.	Enhance Artemis to detect novel vulnerabilities, also investigate the integration of verification and fuzzing methodologies to improve the discovery of vulnerabilities in smart contracts.

Author	Methodology	Findings	Limitations
Deng et al. [33]	DefectChecker identified eight contract faults in Ethereum smart contracts, categorized into five levels of damage. It use symbolic execution to detect errors in bytecode and is verified using an open-source dataset.	DefectChecker exhibits robust performance with an F-score of 88.8% and an average analysis duration of 0.15 seconds per smart contract. It detected flaws in 25,815 of 165,621 Ethereum smart contracts, including vulnerabilities associated with actual attacks.	An SMT approach may decrease false positives and negatives in flaw detection using DefectChecker; nevertheless, it prolongs analysis time. Future research should identify coding trends that contribute to these issues in order to enhance accuracy and efficiency.
Contro et al. [40]	EtherSolve employs a technique that improves the precision of control-flow graphs (CFGs) for Ethereum smart contracts via symbolic execution of the operand stack.	EtherSolve analysis accurately identified all 42 vulnerabilities, achieving an accuracy of 100%.	Limited to reentrancy only

Author	Methodology	Findings	Limitations
Huang et al. [84]	Transforms Solidity bytecode into RGB color codes and a fixed-sized encoded picture. A convolutional neural network (CNN) extracts features from this image to find Ethereum smart contract compiler problems.	Powerful hardware setup on Ubuntu 14.04 to train CNN models on Ethereum smart contract data, achieving notable accuracies with various architectures. Results from Inception-v3 showed high accuracy and recall metrics, particularly with a learning rate of 0.001 over 500 epochs.	Require high computational resources
Wang et al. [141]	HGAT abstracts functions into code graphs utilizing Abstract Syntax Tree (AST) and Control Flow Graph to improve smart contract vulnerability detection, also uses graph attention to extract node characteristics, identifying vulnerabilities quicker and more accurately than previous approaches.	The HGAT model outperforms previous detection approaches for Reentrancy, Timestamp, Integer Overflow, and Integer Underflow vulnerabilities with an accuracy rate of 85.64%. HGAT also has the highest AUC value in ROC analysis, outperforming standard tools and deep learning models.	Enhance the model's accuracy, expand it to new vulnerability detection types, and increase detection efficiency without decreasing accuracy.

Author	Methodology	Findings	Limitations
Hwang et al. [87]	CodeNet uses CNN to convert smart contracts into images while keeping semantics and context to identify them as invulnerable and to improves vulnerability detection performance and efficiency by pre-processing data.	CodeNet has remarkable accuracy (98.79%), recall (98.26%), and F1-score (98.75%). It also works in hundreds of milliseconds, faster than competitors.	Provide more accurate and useful smart contract vulnerability detection
Yuan et al. [228]	SVChecker is a neural network-based method for classifying Solidity code snippets as malicious or benign, demonstrating higher precision in vulnerability detection compared to traditional methods, then also presents VSCL for automatic detection through bytecode analysis and deep learning.	Outperforms existing work, with high-performance metrics including 95.95% precision, 93.73% recall, 94.83% F1 score, and 94.77% accuracy.	Provide more accurate and useful smart contract vulnerability detection with more vulnerabilities

Table 6: DL Existing Approaches

Current deep learning methodology for detecting vulnerabilities in smart contracts use several techniques, see Table 6to improve accuracy and efficiency. Methods like support vector machines, advanced deep learning models such as BLSTM-ATT and Optimized-CodeBERT, together with novel frameworks like Lighting Cat and TxMirror, have shown encouraging outcomes, attaining elevated detection rates and minimal false positive rates. Nonetheless, several solutions are limited to certain categories of vulnerabilities or need substantial computing resources. Moreover, there exists an urgent want for standardized datasets and additional investigation into unsupervised learning to enhance detection skills across various vulnerabilities. Despite progress, obstacles persist in enhancing the application and efficacy of these deep learning models.

11.1 Deep Learning Models

11.1.1 VGG16 Classifier

A convolutional neural network (CNN) utilizes the VGG16 architecture for image classification. Despite having 21 layers, including convolutional, max pooling, and dense layers, VGG16 is named for its 16 weight layers, which are learnable parameters. The network takes input tensors of size 224 $\times$ 224 with three RGB channels. VGG16 is a continuous use of 3x3 convolutional filters with a progression of 1 and 2 with 2x2 maximum pooling layers with the same padding. This uniformity in layer design is a key feature of VGG16, making it easy to understand and implement. The architecture includes five convolutional layers, each followed by a max pooling layer. Conv-1 has 64 filters, Conv-2 has 128 filters, Conv-3 has 256 filters, and Conv-4 and Conv-5 each have 512 filters. The network concludes with three fully connected layers, the first two with 4096 channels each and the third performing 1000-way ILSVRC classification with 1000 channels for each class, followed by a softmax layer for classification.

11.1.2 ResNet50

ResNet50 is a modified version of the ResNet structure, comprising 50 layers that include convolutional layers, pooling layers, fully connected layers, and shortcut connections. The input image should have dimensions of 224 x 224 pixels and consist of three color channels (red, green, and blue). The first layer consists of a 7x7 convolutional layer with 64 filters and a stride of 2. Then, it is followed by a 3x3 max pooling layer with a stride of 2. ResNet50 consists of several residual blocks, each including multiple convolutional layers, which are fundamental advancements of ResNet. These blocks feature shortcut connections that bypass one or more layers, thereby addressing the vanishing gradient problem and facilitating the training of deeper networks. ResNet50 incorporates identity blocks that preserve the spatial and channel characteristics of the input, hence aiding the learning process. Following the convolutional layers, ResNet50 employs global average pooling to decrease the spatial dimensions of the features to a 1x1 vector for each channel. The last layer of ResNet50 uses the softmax activation function to provide a probability distribution across the output classes.

11.1.3 Inception V3

The Inception V3 architecture is a powerful tool for image classification tasks, particularly in vulnerability detection using an RGB image-based approach. Our proposed solution is used to capture features at different scales for identifying potential vulnerabilities in smart contracts represented as RGB images. In this system, where vulnerabilities are displayed as images, Inception V3 analyzes these images and classifies them based on the presence of specific vulnerabilities.

As the field of AI advances and high-performance computing becomes more accessible, the exploration of DL applications in blockchain security is increasingly important, underscoring the need for ongoing research in this area.

Current vulnerability detection methods often suffer from high false positives and negatives, limited generalization and lack of interpretability. These challenges hinder their practical adoption in real-world blockchain environments. Therefore, new methods are needed to achieve greater accuracy, scalability, and explainability in detecting smart contract vulnerabilities.

12 Proposed Methodology and Experimental Results

Traditional approaches to detecting smart contract vulnerabilities mostly depend on static analysis and dynamic execution strategies, as demonstrated by tools like Oyente [140], Mythril [184], Osiris [203], SmartCheck [202], and Slither [62], Manticore [154], Maian [156], and Securify [206]. Although these methods have aided security initiatives, their scalability and efficiency are constrained by their labor-intensive nature and reliance on manual rule development [59].

Natural Language Processing (NLP) word embedding algorithms, including word2vec, FastText, n-gram, and TF-IDF [220] [148] [18] [149], are frequently used in the field of machine learning-based vulnerability identification for feature generation. These techniques, however, struggle with long bytecodes, which could result in memory-intensive training and erroneous predictions [127] [233].

To the best of our knowledge, reentrancy attacks are not thoroughly examined in related research, which instead suggests broad techniques for identifying smart contract weaknesses. However, as noted in the literature, information loss may occur when sophisticated bytecode is converted into images. The transformation, however, has the potential to capture characteristics that are only pertinent for a particular attack. As a result, a comprehensive approach involving meticulous preprocessing and feature engineering is necessary to efficiently represent the bytecode, allowing for the extraction of functional characteristics from bytecode images.

Additionally, the call.The value() pattern is the primary focus of current reentrancy

detection attempts, leaving space for investigating new detection patterns and methods. Delving into the realm of deep learning, where Convolutional Neural Networks (CNNs) have demonstrated remarkable capabilities in image analysis, we aim to overcome these challenges and constraints. Motivated by this, we propose a new method that enhances reentrancy detection in smart contracts by leveraging deep learning and RGB image analysis. Our method is designed to be more effective, precise, and scalable than existing methods, instilling confidence in its ability to detect reentrancy issues accurately by translating Ethereum Virtual Machine (EVM) opcodes into RGB images. Our study aims to provide a novel, automated, and reliable approach to identifying smart contract vulnerabilities. We aim to make a substantial contribution to the ongoing efforts to secure blockchain applications by leveraging deep learning, which will ultimately enhance dependability and confidence in this game-changing technology.

12.1 Detecting Reentrancy Vulnerability using Bytecode

Our study significantly advances blockchain security by introducing a novel image-based methodology for detecting reentrancy vulnerabilities [198], using a VGG16 CNN model. This technique uses deep learning to assess RGB representations of opcodes efficiently, therefore surmounting the limitations inherent in traditional methods. The approach effectively mitigates several false positives and negatives associated with complex code analysis. Furthermore, it obviates the need for source code access, hence enhancing its relevance to practical applications via opcode analysis.

How do opcodes convert into RGB images to detect vulnerability in smart contracts?

Opcode analysis in smart contracts involves examining the bytecode instructions that the Ethereum Virtual Machine (EVM) executes. These opcodes represent the fundamental operations performed by a smart contract, such as arithmetic, storage, and

control flow. By analyzing the sequence and patterns of these opcodes, researchers can identify potential vulnerabilities and security threats.

Converting opcodes into RGB images involves transforming the bytecode instructions into a visual representation of the data. Each opcode is mapped to a unique RGB color value, creating a visual pattern that represents the entire smart contract code. This transformation enables researchers to utilize image processing techniques to identify vulnerabilities.

The use of RGB images provides a new perspective on smart contract analysis, allowing researchers to apply image-based machine learning algorithms for vulnerability detection. These algorithms can extract features from the images and identify patterns that indicate potential vulnerabilities, enhancing the accuracy and efficiency of smart contract security analysis.

The process of converting opcodes into RGB images to detect vulnerabilities in smart contracts typically involves a few steps. This method utilizes the concept of visually representing bytecodes and then applying image processing techniques to detect patterns that indicate vulnerabilities. Here is a general overview of the steps that we follow to convert the opcode into an RGB image for vulnerability detection 6:

1. **Opcode Extraction:** First, the opcode sequence is extracted from the smart contract's compiled bytecode. Opcodes are the operational codes that the Ethereum Virtual Machine executes.
2. **Color Encoding**: Each distinct opcode is then mapped to a unique color in the RGB color space. This mapping translates the opcodes into a sequence of RGB values, effectively converting code into colored pixels.
3. **Image Construction**: The RGB values are placed in an image grid to create a visual representation of the contract's bytecode. The arrangement is typically done linearly, and the image may be scaled to a fixed size for consistency.

4. **Apply Image Processing Techniques***:* The constructed image is fed into a Convolutional Neural Network. CNNs are powerful in image recognition tasks as they can automate the feature extraction process through convolutional layers that detect patterns and features within images.
5. **Vulnerability Detection**: The CNN models VGG16, ResNet50, and Inception V3 were utilized for training to analyze the RGB images representing smart contracts bytecode. These models were used to identify patterns characteristic of vulnerabilities within the smart contract code. This approach leverages the ability of CNNs to extract features from images and detect patterns that indicate potential vulnerabilities, enhancing the accuracy and efficiency of smart contract security analysis.

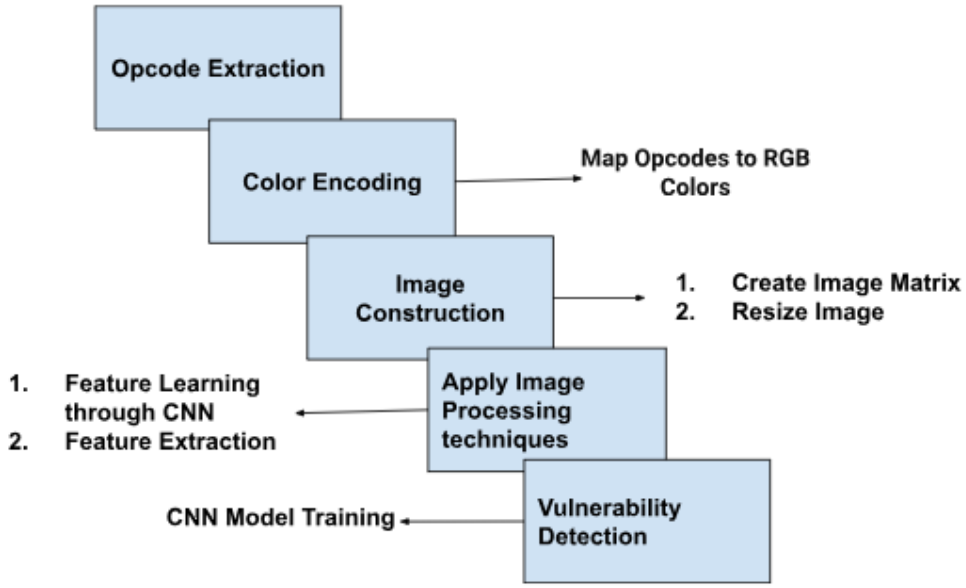

Fig. 6 Opcode conversion to Images

12.1.1 Experimetal Evaluation

In this section, we provide a thorough examination of our experimental method for identifying reentrancy attacks in Ethereum smart contracts. We provide detailed information on preprocessing procedures, model training, performance measures, data collection, and results presentation.

1. ***Data Collection***

   Our dataset comprises two crucial types of smart contracts: those vulnerable to reentrancy attacks, which form the first category, and those immune to such vulnerabilities. The meticulous collection process involved the following steps:
   We initiated our data collection from the SCrawlID repository, a source of tagged samples of insecure contracts. To further enrich our dataset, we meticulously conducted web scraping of Ethereum smart contract data from Etherscan.
   The low-level programming language known as Ethereum Virtual Machine (EVM) bytecode, compiled from a high-level programming language like Solidity, acts as a virtual bridge layer between the application and operating system layers, reducing reliance on the underlying operating system. The EVM enables Ethereum smart contracts to run on a variety of computer platforms. While EVM bytecode is machine-readable, it remains non-human-readable. The evmdasm tool plays a crucial role in this process, successfully converting EVM bytecode into a human-readable format and extracting the corresponding opcodes. This stage is essential for additional processing and analysis, providing reassurance in the process. After processing the opcode set, 145 unique opcodes were extracted. A total of 145 randomly generated RGB codes were produced by assigning a unique RGB color code to each distinct opcode. These RGB-coded opcode sequences were transformed into pictures by us. The pictures' width varied based on the number of opcodes in each contract, but their height was consistent at 224 pixels. Figure 7 displays an example picture.

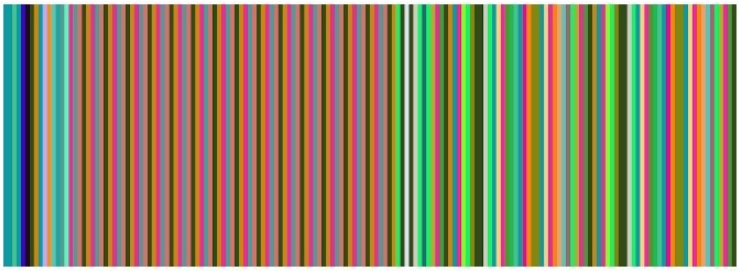

Fig. 7 An image representation of the opcodes of a smart contract

2. ***Model Training***

In our quest to uncover reentrancy vulnerabilities, we harnessed the power of a VGG16 Convolutional Neural Network (CNN) to construct an efficient and effective model. The training process was comprehensive, with a particular emphasis on diversifying our training data through the use of data augmentation techniques. We initially identified 323 SCrawlID contracts that were vulnerable and then created an additional 4,730 samples to fortify this dataset, resulting in a total of 5,053 examples of vulnerable contracts. Importantly, we ensured fairness by treating the non-vulnerable data with the same level of scrutiny and augmentation as the vulnerable data. Our final dataset consisted of 5053 images of vulnerable contracts and 5053 images of non-vulnerable contracts. The dataset was divided into training and testing sets, with each class in the test set receiving 7.5% of the total.

3. ***Performance Metrics***

Precision, recall, F1 score, and support were among the standard classification measures used to assess our model's performance. These metrics provide in-depth insights into how effectively the model can distinguish between vulnerable and non-vulnerable contracts.

4. ***Results***

Our experimental findings clearly demonstrate the effectiveness of our method in identifying reentrancy attacks in Ethereum smart contracts. Table 7 shows the exact

numerical values for precision, recall, and F1-score as well as the support for each. Our findings consistently demonstrate the model's effectiveness in distinguishing

Class	Precision	Recall	F1-Score	Support
Vulnerable	0.91	0.87	0.89	380
Non vulnerable	0.88	0.92	0.90	380

Table 7 Performance Metrics

between vulnerable and non-vulnerable smart contracts, highlighting its reliability in identifying reentrancy issues.

Comparison of our Deep Learning Models with others

Combining these models enhances vulnerability detection by leveraging their strengths. Inception V3 captures the high-level context in RGB images, while VGG16 and ResNet-50 focus on detecting specific vulnerability-indicating patterns. This collaborative approach enhances RGB-based system's ability to identify a wide range of vulnerabilities.

12.1.2 Conclusion

Our image-based method for identifying reentrancy issues in smart contracts, with an impressive accuracy rate of 99.07%, outperforms existing approaches. This method is not just a theoretical concept, but a practical and reliable solution that enhances blockchain security without the need for source code access or complex code analysis criteria.

Our study demonstrates the effectiveness of deep learning and image-based representations in reentrancy detection. The obtained accuracy rate of 99.07% provides compelling evidence of our method's reliability and practicality, making it a valuable tool for blockchain security professionals and developers.

While our method has delivered exceptional results, it also opens the door to further

FEATURE	ResNet-50	VGG16	InceptionV3	Xception	MobileNetV2
Architecture	Residual Network, Contain 50 layers	Very Deep Convolutional Neural Network, Contain 16 weight layers	Inception architecture with residual connections	Depth wise separable 2 convolutional layers	Lightweight architecture with depth wise separable convolutions
Parameters (Millions)	25.6	138	23.8	36.9	3.4
Accuracy (ImageNet)	77.3%	76.7%	77.9%	79.0%	72.2%
Speed	Moderate	Slow	Slow	Moderate	Fast
Memory Usage	High	High	High	Moderate	Low
Strengths	Good balance of accuracy and complexity, handles vanishing gradient problem	High accuracy	High accuracy	High accuracy, efficient use of filters	Efficient for mobile and embedded devices
Weaknesses	More complex architecture, higher memory usage	Very deep architecture	Complex architecture, requires more training data	Higher memory usage	Lower accuracy compared to other models
Use Cases	Image classification, object detection, image segmentation	Image classification	Image classification	Image classification	Mobile and embedded applications

study and development. The potential for extending our methodology to identify more types of smart contract vulnerabilities is an exciting prospect for future research.

As many current systems utilize various methods to identify different vulnerabilities, some employ solidity code to detect vulnerabilities, while specific articles have identified multiple vulnerabilities in smart contracts. However, a significant challenge remains in achieving a high level of accuracy in their identification.

Comparison of existing studies

The comparison of existing studies on smart contract vulnerability detection, as shown in 8, reveals diverse approaches across data sources, representations, models, vulnerability types, and supporting tools. Most studies, such as those by Hwang et al. (2022), Deng

et al. (20230, and Tahir et al. (2023), adopt a comprehensive strategy by integrating all major components, including data representation and model development, for effective vulnerability detection. Others, including Contro et al. (2021), Torres et al. (2020), and Rongwei et al. (2023), rely more on tools and vulnerability identification while limiting emphasis on representation or modeling. Notably, several works such as Kim et al. (2020) and Wang et al. (2020) focus primarily on tool-based detection without addressing data representation or modeling. Overall, the comparison highlights that while some studies attempt to balance multiple dimensions, a large proportion still emphasize tool support and vulnerability identification, leaving room for improvement in integrating robust modeling and data representation techniques for enhanced smart contract security.

Study	Data Source	Representation	Models	Vulnerability types	Tools
Hwang et al. [87]	✓	✓	✓	✓	✓
Contro et al. [40]	✓	✗	✓	✓	✓
Torres et al. [204]	✓	✗	✗	✓	✓
Singhal et al. [192]	✓	✓	✓	✓	✗
Zhang et al. [232]	✓	✓	✓	✓	✗
Chen et al. [33]	✓	✓	✗	✓	✓
Tahir et al. [198]	✓	✓	✓	✓	✗
Deng et al. [48]	✓	✓	✓	✓	✓
Rongwei et al. [227]	✓	✗	✗	✓	✓
Sun et al. [196]	✓	✓	✓	✗	✓
Kim et al. [107]	✗	✗	✗	✓	✓
Wang et al. [208]	✗	✗	✗	✓	✓
Qian et al. [167]	✓	✗	✓	✓	✓
Chinen et al. [37]	✓	✗	✗	✓	✓
Qian et al. [169]	✓	✓	✓	✓	✗

Table 8 Comparison of existing studies

Existing Review papers

Table 9 reveals that majority of existing surveys are inconclusive, relying heavily on hybrid analysis methods while lacking depth in comparative evaluation and definitive conclusions.

Authors	Survey Type
Shiri et al. [187]	ML Models
Janaka et al. [179]	ML Models
Kiani et al. [105]	Static and Hybrid Analysis
Praitheeshan et al. [165]	Static and Dynamic Analysis
Alharby et al. [9]	Blockchain and Static Analysis
De et al. [47]	Static and Hybrid Analysis
luliano et al. [90]	Static and Dynamic Analysis
Kushwaha et al. [118]	Hybrid Analysis
Kushwaha et al. [119]	Static and Dynamic Analysis
Khan et al. [103]	Static and Dynamic Analysis

Table 9: Type of Existing surveys

Furthermore, the effectiveness of detecting errors is hindered by the limits of current detection methods, which rely on static analysis or have poor test coverage. To address these issues, it is essential to develop advanced detection methods that leverage dynamic analysis, machine learning algorithms, and comprehensive test suites on multiple vulnerabilities rather than reentrancy only. Scalability is a concern as the number of contracts grows, and resource-intensive methods like deep learning strain computational resources. Interpreting results from deep learning models is challenging, and real-time detection is difficult in the fast-paced blockchain environment. Validating new methods and benchmarking against existing ones is also challenging due to the lack of standardized datasets and evaluation criteria. Further research should aim to enhance the complexity and comprehensiveness of vulnerability identification within a single system while improving the overall system's robustness and efficacy. This will improve the precision and effectiveness of identifying vulnerabilities in smart contracts.

12.2 Identification of vulnerabilities via Bytecode Image Analysis

In contemporary supply chain systems, smart contracts are gaining traction as a means to transition away from conventional centralized models and toward automated, decentralized, and trustless procedures. These self-executing contracts, with code stored on blockchain networks, have decreased supply chain operating expenses, increased transparency and confidence among supply chain participants, and offered a promising array of benefits such as traceability, decentralization, and immutable code. Leading companies in the sector, such as Walmart, Maersk, and Amazon, have been at the forefront of deploying blockchain-based solutions to optimize logistics and use IoT devices to monitor environmental conditions.

Automated reactions based on real-time data from IoT devices are a significant area of focus. Maersk uses blockchain-based TradeLens for secure documentation, AI for route optimization, and IoT for real-time cargo tracking; Amazon uses blockchain for supply chain transparency, IoT for automated warehouses, and AI for inventory management; and Walmart uses blockchain for food supply chain traceability, IoT for temperature monitoring, and AI for demand forecasting.

Ethereum is unique because it is widely used and programmable. It provides a decentralized framework for deploying smart contracts using the Ethereum Virtual Machine (EVM). The behavior of the contract is controlled by this bytecode, which also serves as the foundation for its communication with the blockchain network. However, due to its shortcomings and examples of malicious attacks in blockchain applications, the rapid adoption of blockchain has made it vulnerable to potential exploitation by attackers. Conventional vulnerability detection approaches, including static and dynamic analysis, are often resource-intensive and time consuming, particularly when addressing complex smart contract flaws. Although tools such as Slither [62], Mythril [184], Oyente [140], and SmartCheck [202] demonstrate effectiveness in uncovering vulnerabilities, they

frequently demand contract execution or expose source code during runtime, raising efficiency and security concerns. Similarly, machine learning and deep learning models especially those employing Natural language Processing (NLP) word embeddings struggle with long bytecodes, which can result in reduced accuracy and computationally expensive training processes.

To mitigate potential risks to the application wallet, this article presents a novel method for securing Ethereum smart contracts by detecting flaws before the contract is deployed on the network. The primary contributions include a new technique for identifying smart contract vulnerabilities while converting opcode-RGB images, a highly successful data transformation procedure that eliminates the need for source code, enhances security, and streamlines the system, as well as comprehensive experimental results demonstrating adequate performance in comparison to the most advanced image-based vulnerability detection techniques.

The study's results demonstrate the success and resilience of this strategy against various vulnerabilities, including those caused by logical errors, unforeseen interactions between contracts, or incorrect handling of user input.

12.2.1 Research Questions

To validate the proposed methodology, we formulate specific research questions that guide our evaluation process. These questions ensure a thorough examination of the effectiveness of the model in detecting vulnerabilities, the role of its architectural components, and its adaptability to real-world scenarios such as IoT environments. By addressing them, we aim to assess the robustness, efficiency, and practicality of our approach. This structured evaluation not only benchmarks our method against existing techniques but also demonstrates its potential for broader applicability.

RQ1

How well does the novel methodology identify vulnerabilities in smart contracts? Investigating how well our innovative method works to identify smart contract vulnerabilities in various complex scenarios is the goal of this research question. To evaluate the generality and efficacy of our technique, we first tested it on publicly accessible datasets that included samples of several vulnerability types. We also compare our results with current vulnerability detection techniques, which serve as benchmarks to demonstrate whether we consistently deliver reliable results.

RQ2

How do the many components in our strategy affect the final result? To fully evaluate the effects of different architectural components on the functionality of our suggested convolutional neural network, we performed a comprehensive set of ablation experiments. Additionally, we examine how well our CNN model performs in comparison to the more widely used deep learning architectures for image-based detection techniques.

RQ3

Is our method reliable for Internet of Things applications? The purpose of this study is to determine whether our method is effective in resource-constrained scenarios, which are common in real-world IoT applications. Our model employs several strategies, including global average pooling and convolution, to strike a balance between a deeper architecture and a lightweight design. We compared our model with MobileNets, the most widely used tiny CNN architectures, which can be easily implemented in real-time utilizing embedded devices such as smartphones and drones, in order to address this research question.

12.2.2 Proposal

This section outlines our proposed methodology for vulnerability detection in smart contracts, emphasizing the transformation of bytecode into visual representations. the approach leverages CNNs to capture opcode patterns, enabling accuracte classification across multiple vulnerability types.

1. ***Approach***

   The core idea of our suggested method is to utilize convolutional neural networks (CNNs) to identify vulnerabilities by leveraging the visual encoding of smart contract bytecode. Fig 8 gives a summary of our methodology. The process begins by gathering smart contracts from publicly accessible datasets, specifically SCrawlD and Messi-Q, ensuring the transparency and reproducibility of our research. An EVM disassembler is used to break down the raw bytecode after these contracts have been extracted using a custom crawler and translated to bytecode format. This process transforms the binary instructions into opcodes that humans, such as CALLVALUE, PUSH, JUMP, and STOP, can understand. The data preparation section contains information on this stage. The transformation phase at this stage creates a visual RGB image representation of the smart contract's instruction sequence by mapping each opcode to a unique color using a predetermined color mapping scheme. The CNN model uses these pictures as input. Within these color-encoded images, patterns of opcodes linked to specific vulnerabilities will appear as recognizable visual characteristics (for further information, see the section "Opcode to Images"). A CNN model is then meticulously trained using the resulting dataset of opcode RGB pictures that have been tagged by vulnerability type (such as Reentrancy, Timestamp Dependency, Locked Ether, Arithmetic Errors, etc.), ensuring its accuracy. A fresh RGB image is fed into the trained model

during inference, and it produces a classification output that identifies the kind of vulnerability in the contract (such as Reentrancy, etc.).

Fig. 8 Overall operation of the proposed methodology

2. ***Data Preparation***

The following procedures were employed to carefully control the data preparation process, as depicted in Fig 8 . We utilized the Etherscan API to obtain the contract source code or bytecode for each dataset, and we developed a web crawler to extract the matching contract bytecodes from etherscan.io. Next, we processed the Ethereum Virtual Machine (EVM) bytecode and extracted the corresponding opcodes using the advanced evmdasm tool. This use of cutting-edge tools ensured the technical rigor of our study. We gathered a total of 7,355 contracts and 10 distinct vulnerabilities after eliminating duplicates and live contracts on etherscan.io that lacked valid bytecode. In our tests, we integrated the publicly available Ethereum Smart Contract dataset with four other publicly available datasets: ScrawID3, Messi-Q4, SmartBug5, and SoliFI.

Yashavant et al. [225] collaborated to create the SCrawlD dataset in 2022, which includes 6,780 tagged contracts with eight distinct vulnerabilities. This collaborative effort ensures the dataset's comprehensiveness and reliability. The authors utilized five different tools: Slither, Mythril, Smartcheck, Oyente, and Osiris, each with distinct versions and dependencies, to label this dataset. They used a majority

vote criterion to determine the appropriate label. The dataset repository, initially provided in .csv format, lists contract addresses with identified vulnerabilities in the dataset. Using the addresses included in a .csv file, we created a Python-based web crawler, which involved [specific steps or methods], to obtain the contract bytecode code from etherscan.io. A Python script implementing a Python-based EVM disassembler library was then used to compile the recovered data into bytecode files and disassemble them into opcodes.

The Messi-Q Dataset [135, 136, 168, 137] comprises 40,000 unlabeled contracts, which include smart contracts from multiple sources. The authors describe a system that ensures accurate and consistent categorization across different vulnerability types by combining automated analysis with expert-defined patterns. We then obtained 12,000 contracts with four distinct vulnerabilities and used the authors' publicly available scripts and code. The solidity code is kept in columns as text. To create distinct .sol files, we extracted each piece of code. These Solidity files were transformed into EVM bytecode using the py-solc.x compiler, and the opcode representations were then obtained using the EVM disassembler.

A dataset of buggy contracts injected by 9369 bugs from seven distinct bug types: reentrancy, timestamp dependency, unchecked send, Transaction Ordering Dependency, integer overflow/underflow (Arithmetic), and use of tx.Origin (DofS) can be found in the public repository SolidiFI-benchmark [67]. To ensure correct compilation, we cloned the SolidiFi repository and addressed version compatibility issues. We introduced vulnerabilities into the base contracts that were accessible within the repository, adhering to the dataset's requirements. We construct programs to divide contracts into independent contract files and create a dataset because many.sol files contain numerous contracts. Files with serious syntax and compatibility problems were excluded from the preparation pipeline.

A set of vulnerable Solidity smart contracts taken from the Ethereum network is

available in the SmartBugs dataset. The dataset has a folder structure containing pre-curated smart contracts. Sol files are arranged class-wise based on the dataset's nine distinct vulnerabilities. After immediately compiling Solidity files into bytecode, we transformed them into opcodes using the EVM disassembler. The public repository devoted to this experimental data contains the scripts and code.

3. ***RGB-encoding of the opcodes into images***

A total of 145 distinct opcodes were identified in our dataset, and each one was assigned its own unique RGB color code. Figure 9 illustrates the color assignment for several opcodes. This image clearly illustrates how each opcode has a unique color associated with it. For instance, the color blue is used to visually represent the opcode CHAINID, whereas the color pink is used to indicate CALLCODE. It's important to note that for every one of the 145 opcodes, 145 distinct color representations were created in a random and unbiased manner, ensuring the objectivity of our research.

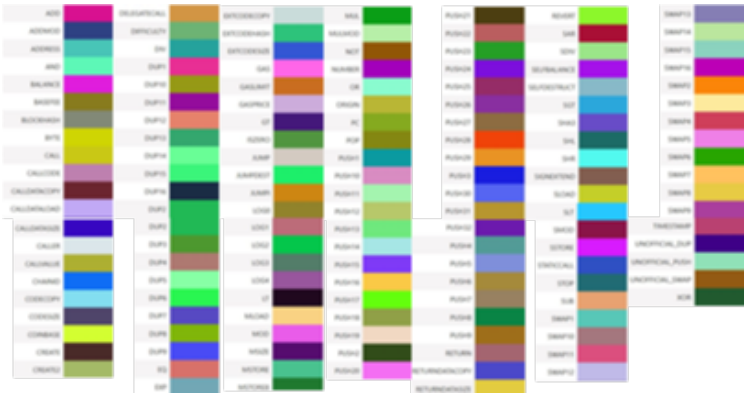

Fig. 9 Color representation for each opcode identified in our dataset

In order to turn each smart contract into a visual representation, we convert the entire opcode sequence into a list of vertical RGB channels. Each opcode is represented by a 10px rectangle with a set height of 224px. To ensure that all opcodes are maintained, the image's full width is determined by the size of the

opcode series in the contract list. Next, we scale every image to a fixed size of 224×224 pixels before feeding the CNN model for training. In particular, when

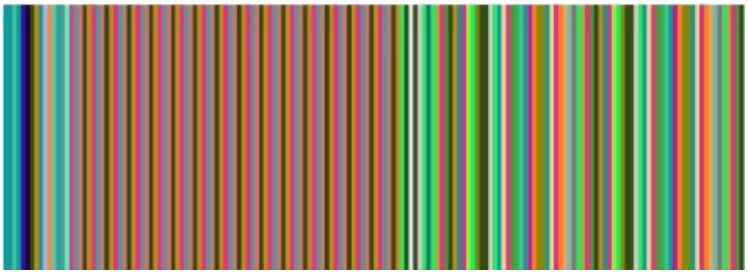

Fig. 10 An example of image obtained from a our RGB-enconding on a smarct contract

maintaining fine-grained spatial relationships and overall image structure is essential for model performance, we employ binary interpolation for image resizing, which is typically more successful than padding or cropping. Since images in our method lack relevant aspect ratios and real-world objects, straight resizing using bilinear interpolation does not compromise the semantic integrity of the patterns. The bar patterns scale linearly when resized, but CNN can still reliably understand them 10. By avoiding center cropping or padding, this method preserves the entire opcode sequence in a fixed-size input that works with all models, including VGG16.

12.2.3 Experimental Evaluation

In this section, we present the experimental setup used to validate our proposed model. We first describe the data composition and distribution of the vulnerabilities, followed by the details of the CNN architectures employed and their baseline comparisons. Finally, we summarize the experimental settings and hardware specifications to ensure reproducibility and fair evaluation.

1. ***Dataset***

   The dataset, which comprises a comprehensive 10,428 contracts with 12 vulnerabilities, encompasses both typical and uncommon security issues in smart contracts.

Classes	Train	Test	Validation	Total
Arithmetic Error	3321	293	293	3907
Block Dependency	146	12	12	170
Block Number Dependency	230	20	20	270
Integer Overflow	352	30	30	412
Locked Ether	1218	107	107	1432
Reentrancy	1609	141	141	1891
Timestamp Overuse	414	36	36	486
Timestamp Dependency	797	70	70	937
Unchecked External Call	507	44	44	595
Total	9594	753	753	10100

Table 10 DATASET DISTRIBUTION ACROSS TRAIN, TEST, AND VALIDATION SETS FOR EACH VULNERABILITY CLASS

The data set has been divided into training, testing and validation sets (7.5%, 7.5%, and 7.5%, respectively. To ensure that relative class frequencies are roughly maintained in each fold, 10-fold cross-validation was used in conjunction with stratified sampling. The table 10 displays the details of the distribution of vulnerabilities.

2. ***Model Architectures Used***

Our proposed CNN architecture is designed to classify vulnerable smart contracts in a practical and efficient manner. It creates a compact model that is suitable for real-time operation on embedded devices such as smartphones and drones. This architecture achieves a deeper structure by utilizing a variety of unique techniques, such as 1×1 convolution and global average pooling. Figures 11 12 illustrate our CNN model's implementation and structural components.

To ensure the robustness of our model, we compared it with three cutting-edge CNN architectures: VGG16, ResNet-50, and Inception-V3. These models were selected because of their complementary architectural features (see figure 12), broad use, and demonstrated efficacy in vision-based tasks across a range of application domains, including binary visualization [210], malware classification [102, 131], and particularly for smart contract bytecode analysis [124, 170, 87], demonstrating their applicability and usefulness as reliable baseline architectures for this field. These models represent three different families of convolutional neural networks (CNN).

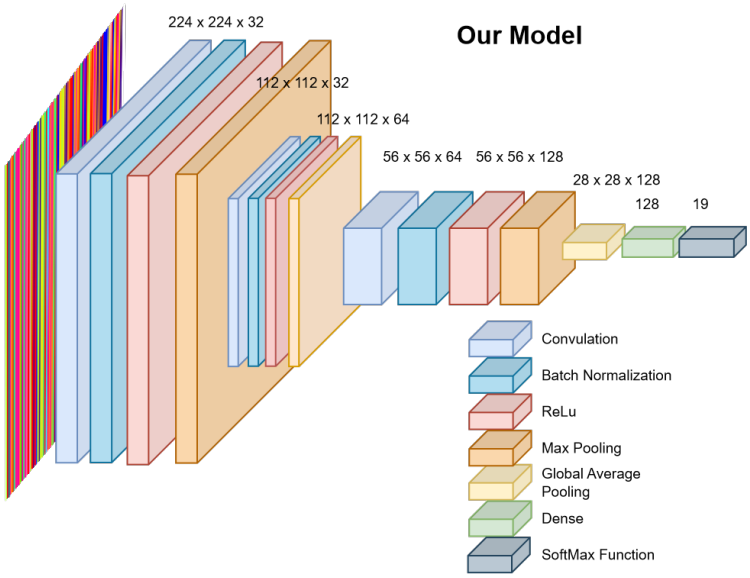

Fig. 11 An overview of our proposed CNN model layer wise

VGG16 is a member of the VGG network family, which employs a simple sequential design that efficiently captures local patterns and utilizes uniform 3x3 convolutional layers.

Despite its simplicity, it is a powerful baseline model due to its consistency in feature extraction, particularly in controlled settings with constrained GPU resources. The most commonly used ResNet design is ResNet50, which incorporates residual connections into deeper networks to address vanishing gradient problems and facilitate the learning of intricate features. Lastly, InceptionV3 is a member of the Inception architecture family, which introduced the concept of "inception modules," or parallel convolutional procedures that concurrently extract features at multiple scales.

Furthermore, we compared our model with MobileNets, a CNN that is part of the lightweight architectural family designed for embedded and mobile vision applications. CNNs integrated into Android phones to run Google's Mobile Vision API, which can automatically recognize the labels of well-known items in photos, are real-world instances of the MobileNet CNN architecture.

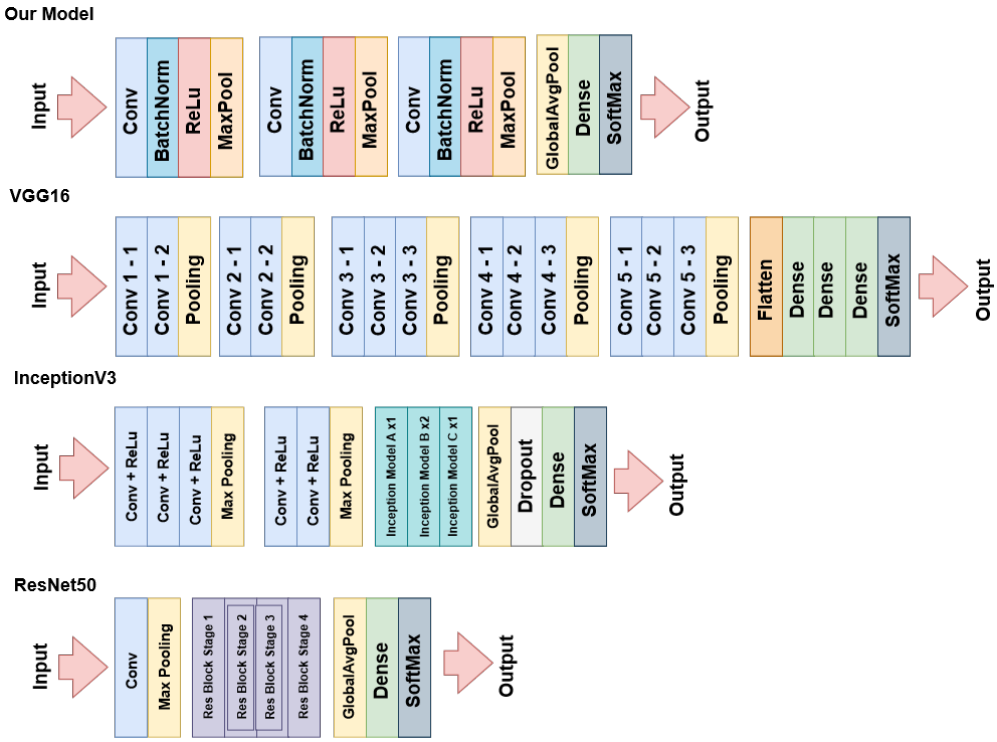

Fig. 12 Architectures of the different implemented CNN models for imagebased vulnerabilities detection

3. *Hardware specifications and experimental settings*

All experiment was carried out on a server running at a base clock speed of 2.50 GHz with 24 GB of RAM, 12 physical CPU cores, and 16 logical processors. It is crucial to remember that all models were trained solely on the CPU, with no GPU acceleration used. To evaluate the viability and performance of each architecture in resource-constrained environments, such as IoT devices with limited computational resources, where GPU access may not be available, we conducted training on a CPU-based system.

All models were trained using a standardized configuration, as indicated in Table 11, to provide uniformity and comparability across experiments. In particular, we use a batch size of 32, which is a widely used number that strikes a balance between

Table 11 MODEL'S HYPERPARAMETERS

Configuration	Value
Batch size	32
Learning rate	0.0001
Weight decay	0.001
Loss function	Categorical Cross-Entropy
Activation function	Softmax

computational efficiency and gradient estimation quality, especially in CPU-bound training situations [186]. Due to its demonstrated stability and effectiveness in training deep neural networks, particularly when dealing with noisy input and sparse gradients—the Adam optimizer was chosen [109]. The initial learning rate was set at 0.001, a commonly used substantial default value suggested in the original Adam study. A learning rate decay strategy, which has been demonstrated to enhance generalization and avoid premature convergence, was employed to promote convergence during training by lowering the learning rate by a factor of 0.1 every 10 epochs. Weight decay was set to 1e-4, in accordance with best practices [186] for L2 regularization in classification tasks, to regularize the model and reduce overfitting. The categorical cross-entropy loss, which is suitable for multi-class classification tasks and closely relates to the probabilistic interpretation of softmax outputs, was the training objective. To ensure interpretable class predictions and compatibility with the cross-entropy loss, the output layer was designed to employ a softmax activation function. This function, by converting logits into normalized probability distributions, plays a crucial role in making the model's predictions more understandable and reliable, thereby enhancing our confidence in the model's performance.

12.2.4 Results and Discussions

In addition to training setup information, including the number of epochs, layer depth, and model size, Table 12 highlights the following important performance metrics: precision, precision, recall, and F1 score. Notably, the bespoke architecture (Our

Model) outperformed all baseline models in every category, achieving the best overall performance. With an F1 score of 94.41% and an accuracy of 94.12%, it demonstrated strong dependability and balanced predictive power between classes. It is important to note that this achievement was achieved while maintaining a modest depth (only 14 layers) and a relatively compact footprint (just 380 KB), making it ideal for deployment in settings with limited resources.

With an F1 score of 89.83% and comparatively quick convergence, which means the model reached a stable performance, in just 25 epochs, VGG16 outperformed the other conventional architectures. However, compared to the bespoke model, it required significantly more memory.

ResNet50 and InceptionV3, with their comparable performance (F1 scores of 86.42% and 87.57%, respectively), provide a reassuring choice. However, it's important to consider that they required more computation time, particularly ResNet50, which utilized over 76 MB of memory and trained for more than 60 epochs.

Despite being the lightest model (464 KB), MobileNetV2's resource efficiency is truly commendable. However, it's important to note that it exhibited low accuracy and recall, suggesting that its depthwise separable convolution approach may not be the optimal choice for identifying patterns associated with vulnerabilities in binary image data.

1. ***Ablation Study***

   We conducted several ablation experiments to thoroughly evaluate the impact of various architectural components on the functionality of our proposed convolutional neural network. To evaluate their unique impact on model correctness, we methodically removed or changed key layers and hyperparameters. Figure 13 14 displays the results. The last Conv2D layer is removed: By removing the final Conv2D layer and its matching MaxPooling layer, we examined the effects of a lower feature extraction depth. Prior to the dense layers, the updated design consisted of only two Conv2D blocks. The classification accuracy decreased significantly to 78.6%, a result that

Model Name	Accuracy	Precision	Recall	F1 Score	Inference Time
ResNet50	88.68 % ± 1.79	89.39%	85.00%	86.42%	1.86 sec
InceptionV3	90.73 % ± 0.38	93.19%	89.12%	90.16%	2.87 sec
VGG16	86.77% ± 1.04	92.14%	83.44%	85.87%	2.99 sec
MobileNetV2	64.37% ± 1.6	60.146%	62.193%	63.572%	1.11 sec
Our Model	92.24% ± 1.2	90.33%	89.44%	89.06%	1.65 sec

Table 12 Performance comparison of CNN architecture used for smart contract vulnerability detection

underscores the urgency for further research. This decrease highlights the necessity for more complex hierarchical feature representations to identify discriminative patterns in the input images, a task that remains a challenge in the field of machine learning. *Elimination of Dropout Layers:* We removed both Dropout layers from the fully connected portion of the network to examine the regularization function. The model exhibited significant overfitting, as indicated by a decrease in test accuracy to 80.3%, despite an improvement in training accuracy. This underscores the crucial role of Dropout layers in reducing overfitting and improving generalization, a key aspect of model performance.

Reduction in the number of filters (capacity): We decreased the number of filters in each convolutional layer by 50% (for example, from 32, 64, and 128 filters to 16, 32, and 64 filters) to assess the effect of model capacity. The accuracy dropped to 76.5% as a result of this simplification, highlighting the challenge of maintaining strong performance on this task, which necessitates a substantial amount of representational capacity. *Modification of the activation function:* By substituting Tanh for ReLU, we examined the effect of the activation function in the dense layers. Tanh outperforms ReLU in the final dense layers in capturing nonlinear correlations in our dataset, as seen by the configuration's 81.7% accuracy. As a result of the modification, training instability increased and convergence speed reduced.

Our Model: Our final architecture, which included three Conv2D layers with ReLU activation, appropriate Dropout regularization (0.5 and 0.3), and completely connected layers, outperformed all ablated alternatives with an astounding accuracy

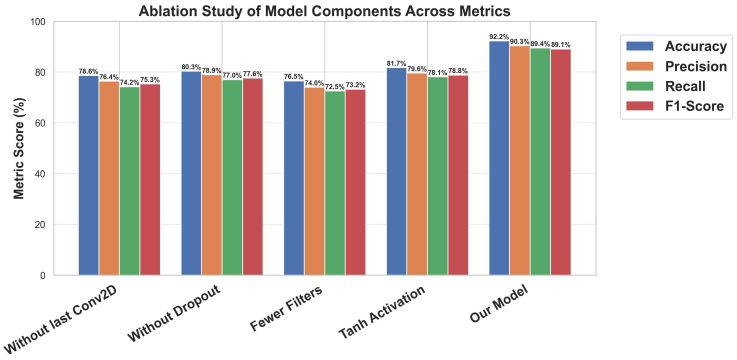

Fig. 13 Impact on test accuracy at varying of components in ablation study

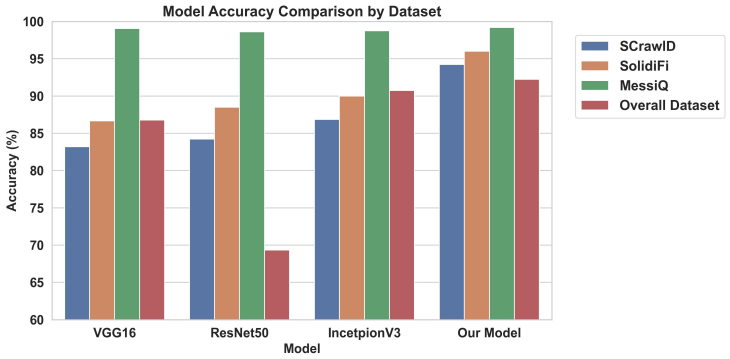

Fig. 14 Model's performance on different datasets

Model Name	Time Consumed	Model Size
ResNet50	20 min	76.6 MB
InceptionV3	20 min	39.1 MB
VGG16	18 min	19.2 MB
MobileNetV2	30 min	464 KB
Our Model	22 min	380 KB

Table 13 Performance Comparison of CNN architecture with time and size

of 94.12%. Furthermore, it remained tiny at 380 KB, see Table 13 allowing for deployment in contexts with limited resources.

2. ***Class Imbalance Analysis***

Analysis of class imbalance: With 3944 samples in the Arithmetic Error class, 1989 samples in Reentrancy, and only five samples in Transaction State Dependency, the

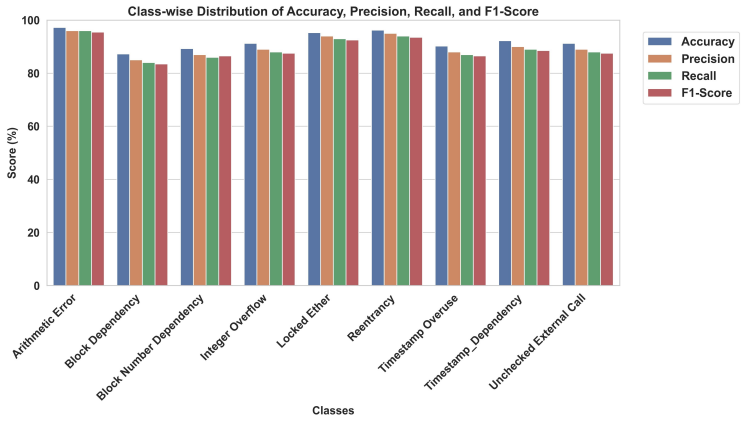

Fig. 15 Analysis of performances metrics over each classes

final dataset shows a significant class imbalance. We recognize that the distribution of vulnerabilities in deployed smart contracts in the actual world has an impact. Our data relies on several series, making our classification task unique and more complex than usual. Contextual dependency and instruction order are crucial elements for vulnerability detection in our task. The sequence may be broken, and invalid instances may result from the introduction of synthetic or oversampled samples. Despite the substantial data imbalance and the lack of mitigation strategies, our model 15 consistently performed well across the majority of classes. We maintained the integrity of the actual data and did not employ any imbalance mitigation strategies.

12.2.5 Conclusion

In this study, we introduce a novel and portable image-based method for identifying various vulnerabilities in Ethereum smart contracts. Our approach eliminates reliance on source code, eliminates the requirement for intricate rule-based systems, and facilitates opcode-level analysis by transforming bytecode into RGB-encoded images and using a customized convolutional neural network (CNN). With a small model size of only 380 KB, the experimental findings across four benchmark datasets

demonstrated exceptional performance, achieving an accuracy of 94.12% and an F1-score of 94.41%. This high level of accuracy instills confidence in the effectiveness of our method, making it suitable for deployment in IoT scenarios with limited resources.

Unlike current techniques, our solution ensures that no semantic information is lost during image production, maintaining the entire opcode structure. This unique feature, combined with the approach's flexibility in responding to new or changing threats, sets it apart from other methods. By not relying on pre-established security practices, our approach is well-suited for dynamic environments.

Despite the encouraging outcomes, there is limitation for improvement. By incorporating explainable AI (XAI) tools to identify which image regions contribute most to vulnerability detection, we aim to address the current interpretability issue of the model in future work. Finally, to verify scalability and resilience in a new real-world setting, we aim to enhance the handling of class imbalance and expand the assessment to datasets with novel vulnerabilities.

12.3 Opcode-Based Reentrancy Vulnerability Detection in Smart Contracts Using Explainable Deep Learning

Ethereum-based decentralized applications in governance, supply chain management, and finance have demonstrated the importance of securing smart contracts. However, the most destructive attacks are still reentrancy attacks, which not only deplete resources but also result in significant losses and necessitate robust security measures.

The current machine learning and deep learning tools, as well as static and dynamic analytics tools, are valuable in identifying patterns of known vulnerabilities. However, their reliance on run-time settings or demand processing capacity often leads

Ref	Methodology	DL/ML model	Accuracy	F1 score	Dataset	Limitations
[237]	Source code graph features and security patterns-to-Gray image	Customized CNN	95.78%	93.86%	Ether smart contracts (ESC)	Source code only / strict rulesbased
[87]	Bytecode-to-image	Customized CNN	98.27%	98.24%	Smartbugs	fixed size argument/computational overhead/only 4 vulnerabilities
[124]	Source code-to-gray image	VGG16 and GRU	100%	100%	Ether smart contracts (ESC)	Source code only/strict rulesbased
[193]	Opcode-to-images	MLkNN, BRkNN, RF, NB	not mentioned	99.41%	SmartBugs	limited window size in N-gram embedding/ memory intensive
[85]	Bytecode-to-RGB image	Customized CNN	97.39%	not mentioned	Crawled from etherscan.io	pixled based fixed size images
[170]	Opcode-to-image	ResNet-50, Lenet-5	82.44%	86.04%	Awesome Buggy-ERC20Tokens	strict rules-based
[176]	Opcode-to-RGB image	ResNet-50, ResNeXt-50, Inception v3, EfficientNet-v2, SwinV2-T	65.93%	77.82	Slither Audited Smart Contracts Dataset	truncated opcode patterns
[176]	Bytecode-to-RGB images	Resnet-18, Inception, EfficientNetV2, MobileNetV3	96%	0.78	Slither Audited Smart Contracts Dataset	fixed vulnerability pattern
Our work	Opcode-to-RGB image	VGG16	99.07%	90%	Messi-Q, ScrawlD SmartBugs, SolidiF	lack of interpretability

to scaling issues. Moreover, these techniques act like 'black boxes,' lacking interpretability. This lack of interpretability in high-stakes applications, such as finance, compromises their credibility, accountability, and ability to be legally defended, highlighting the need for scalable and interpretable vulnerability detection tools. The need to bridge the crucial gap between interpretability and detection is what spurred this study. Our goal is to develop a robust, environment-independent deep

learning model that can interpret smart contract opcode patterns and utilize XAI approaches to enhance the model's transparency and interpretability, because it not only improves the interpretability of model decisions but also build trust by revealing the reasoning behind predictions as highlighted in the recent studies:

Grad-CAM has been successfully applied in various medical diagnostics, including Alzheimer's disease (AD), malaria parasite detection, and UC diagnosis. **Mahmud et al.** [142] used ensemble modeling and deep transfer learning to provide an explainable AI (XAI)-based method for diagnosing AD, enhancing the interpretability of deep learning models. Ensembles, Ensemble-1 and Ensemble-2, showed better accuracy, precision, recall, and F1 scores compared to individual models.

Islam et al. [89] proposed a multiheaded attention-based transformer model for detecting malaria parasites from blood cell images using the gradient-weighted class activation map (Grad-CAM) method. The model achieved high testing accuracy, precision, recall, f1-score, and AUC scores. **Zou et al.** [246] introduced ensemble AI explainability (XAI), a novel image explainability technique based on Grad-CAM++ and SHAP methods.

Dhole et al. [49] presented a novel AI framework for UC diagnosis that uses Grad-CAM for interpretability and Sparse Autoencoders (SA) for feature extraction. SA effectively encodes critical information for diagnosis by reducing the dimensionality of medical images. A machine learning classifier receives the retrieved features to detect UC, with impressive results of 98.5%, 97.5%, 96.4%, and 95.5% in accuracy, precision, recall, and F1 score, respectively.

Rahman et al. [171] developed a bespoke convolutional neural network model called Glioma-CNN, which achieved an impressive accuracy of 99.1569% using the BraTS 2020 dataset. Grad-CAM improved pneumonia identification accuracy to 95.6% by visualizing key regions in X-ray images.

Ahmad et al. [4] introduced a novel framework for skin lesion detection using

explainable artificial intelligence, deep learning, and data augmentation. The framework outperformed state-of-the-art techniques in terms of accuracy and computational time.

LIME and SHAP, which utilize Explainable Artificial Intelligence (XAI), have been applied across various fields to enhance model transparency and interpretability. For instance, **Raval et al.** [172] presented a deep learning model for real-world leaf disease diagnosis, outperforming 94% accuracy on unbalanced data and 99.9% accuracy on high-quality, balanced data. **Ahmed et al. [6]** explored the application of explainable AI in healthcare, constructing a machine-learning model using 253,680 survey responses and elucidating predictions generated by Random Forest and logistic regression-based models.

LIME proved to be the most effective method for clarifying decision-making processes in deepfake detection by **Tsigos et al.** [207]. The SHapley Additive exPlanations technique by **Morgan et al.** [153] was used to combine the RFC model with XAI, achieving 94.4% accuracy in identifying multispectral characteristics for lithological discrimination in dry crystalline regions.

The DeepSHAP framework [46] integrates elastic net-based feature selection, a fine-tuned deep neural network, and dual explanation mechanism, contributing to its superior interpretability in predicting complex conditions. The SP-RISA algorithm [81] aligns human retionales with model inferences through feature attribution, significantly lowering calibration errors across datasets. In agricultural diagnostics, feature extraction using the Hough Transform (HT) and Discreate Wavelet Transform (DWT) has improved classification, with logistic regression achieving an accuracy of 0.99 in the YCbCr color space.

Medical imaging demonstrates the strength of XAI, with the pediapulmoDx [166] framework achieving high diagnostic accuracy and hybrid methods for COVID-19 detection achieving 100% precision \cite{akbar2024contemporary}.

Evirgen's goal [60] is to provide explanation strategies for inexperienced users to improve their comprehension of Text-to-Image (T2I) models, examining their opaque nature.

Gomez et al. [69] explore the transformative potential of AI in primary eye care, demonstrating that AI can improve referral accuracy and outperform human-AI partnerships. It improves the interpretability of a Convolutional Neural Network (CNN) for fetal ultrasound picture classification using local interpretable model-agnostic explanations (LIME). This approach aims to foster confidence in AI applications used for prenatal diagnosis.

To combat confounding effects in AI-based medical image analysis, [134] presents a novel strategy that aims for a confounder-free representation while preserving confounder-related information in latent representations.

The DeepSHAP [46] methodology by **Davagdorj et al.** enhances the interpretability of deep neural networks in predicting non-communicable diseases (NCDs) by using Shapley Additive Explanations. This model, consisting of a deep neural network classifier trained with specified features, two types of explanations provided by DeepSHAP, and an elastic net approach for feature selection, has shown superior performance in experimental data [20]. It improves medical professionals' understanding of NCDs by shedding light on differences in disease risk.

Automatic labeling [106] addresses the drawbacks of human annotation by automatically labeling large medical imaging datasets using a verified explainable AI model. This technique can retain or enhance the original model's performance, offering a hopeful outlook for the future of medical imaging.

The 'What I Know (WIK)' method by **ishikawa et al.** [88] enhances the reliability of deep learning models by providing instances from the training dataset that closely match the input data. This approach emphasizes the importance of the training dataset and model architecture while validating model conclusions within applications.

Chanda et al.[31] found a strong correlation between Explainable AI (XAI) systems for diagnosing melanoma, boosting diagnostic confidence and trust in the AI support system. The Semantic Encoder is a CNN model developed to enhance explainability of autonomous systems in industrial contexts. PollenNet [181], a deep learning framework, has successfully improved the categorization of pollen grains with minimal mistake rates.

Borys at al. (2023) [24] highlights the need for a comprehensive understanding of AI systems in healthcare by exploring various explainable AI techniques in medical imaging. The primary goal is to enhance the integration of AI in medical imaging and personalized clinical treatment, promoting collaboration between Deep Learning engineers and healthcare professionals.

The existing literature demonstrates that XAI techniques, such as Grad-CAM, SHAP, LIME, and ensemble models, have significantly improved interpretability in diverse domains, including medical diagnostic, agriculture, and security. These approaches emphasize the importance of feature attribution, visualization, and model transparency to enhance trust and usability of AI systems. Reported results often achieve high accuracy rates, up to 99%, while providing information on critical characteristics that influence predictions. Taken together, these findings emphasize the potential of XAI in bridging the gap between performance and interpretability. When applied to smart contract vulnerability detection, such methods can help uncover code-level risk factors while offering clear explanations for developers, paving the way for reliable and explainable blockchain security solutions.

In particular, we experimented with Integrated Gradients to generate token-level attributions that translated RGB color codes into the opcode bars most influential in the classification decision.

12.3.1 Methodology

In this section, we describe our methodology for smart contract vulnerability detection. we outline the data preprocessing pipeline, explain the rationale behind model selection, and present the use of XAI techniques for interpretability. Finally, we report experimental results demonstrating the effectiveness of the proposed framework.

(a) ***Data Pre-processing***

Our process began with obtaining the Messi-Q dataset **??**, which includes tagged instances of Ethereum smart contracts. This dataset served as the foundation for our training dataset for vulnerability identification. The data, which included metadata and the solidity source code of the contract, was initially made public on Kaggle in CSV format. We then transformed this data, converting it from CSV files to Solidity source files in .sol format. Subsequently, we rebuilt every row into a legitimate, working smart contract to facilitate compilation and analysis. For the compilation of each contract, we relied on the Solidity compiler Python library (solcx) **??**. This library encapsulates the Solidity compiler, enabling us to compile smart contracts into EVM bytecode. After the data was converted to .sol format, we used solcx to compile each contract. This step allowed us to effectively deploy and examine the contracts within the same Ethereum Virtual Machine. In this pahse, we verified that the remaining contracts are valid and executable by identifying those with syntax issues, such as incorrect variable declarations or function calls, and those that cannot compile due to errors in the code structure or dependencies.

The compiled bytecode was then meticulously broken down into opcodes using evmdasm. This process converted the raw bytecode into human-readable opcode instructions, such as CALL, CALLVALUE, and JUMPDEST, ensuring the clarity

and comprehensibility of our analysis. The dataset was previously labeled because the data distribution for the reentracy classification was already labeled.

We created a data transformation pipeline that maps each opcode to a distinct RGB color using a predefined dictionary, thereby preparing the opcode data for use in deep learning. The previously created opcode-to-RGB mapping is then used to convert these opcodes into visually engaging 1D RGB strips, providing a unique and intuitive way to interpret the data.

The data preprocessing pipeline is illustrated in 16. We assigned each opcode

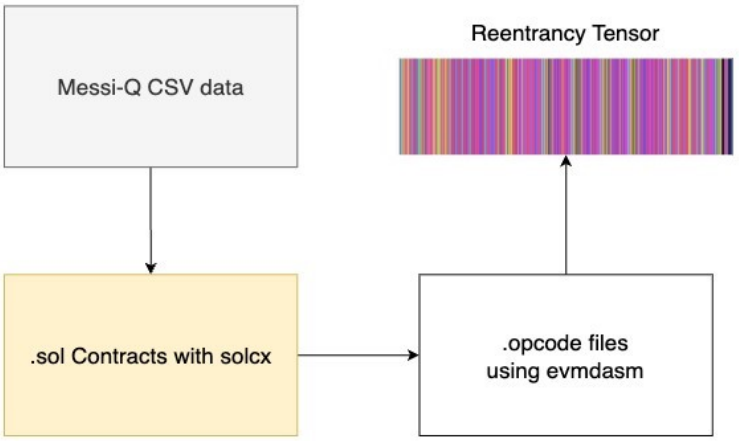

Fig. 16 Data Pre-processing Pipeline

its own RGB color code, taking into account collapsed categories such as DUP1-16, SWAP1-16, and PUSH1-32. The length dimension of the imagelike tensors corresponds to the sequential execution of opcodes, while the depth dimension represents the RGB channels. Each opcode in the image represents a bar of 10px width, fixed with a 1000px high bar. This enables us to transform a contract into a horizontal image in a structured format, maintaining the execution order of the opcode sequence while representing it spatially.

This method preserves important elements of the execution flow and contract semantics while converting contracts into a structured image-like format that

Category	Count
Reentrancy Contracts	1219
Non-Reentrancy Contracts	1300
Total	2519

Table 14 Contracts category distribution

CNN can process. This approach is highly effective, enabling the model to learn patterns corresponding to vulnerable behavior, making it suitable for convolutional processing. The following number of reentrancy and non-reentrancy contracts were included in the data conversion; the non-reentrancy contract was chosen at random from the same dataset, ensuring fairness and objectivity. The quantity of contracts used in model training and experiments is displayed in 14.

(b) ***Model Selection***

We empirically evaluated a number of deep learning architectures in our initial experiments, including VGG16, ResNet50, and Inception V3. These models performed well in the image classification task, but when applied to our dataset—which lacks intricate spatial hierarchies—they showed less than ideal generalization. Moreover, its use is restricted by computational cost, memory expense, and incompatibility with the 1D sequential nature of our data. We decided to use a lightweight 1D CNN design in light of these findings. Since our data is not real images with features, 1D CNN is more appropriate from a methodological and practical standpoint. Additionally, the model is computationally efficient for this task. Local patterns in opcodes are being captured by 1D CNN with small kernel widths while preserving model simplicity, interpretability, and training effectiveness.

(c) ***XAI Selection***

While Grad-CAM is a widely used technique in recent research on deep learning image-based techniques, this technique is specifically designed for visual saliency

in 2D spatial feature maps, making it less effective in the context of 1D strip-like images. In these images, where each opcode is represented as a bar and each vertical bar encodes a distinct opcode token, Grad-CAM's relevance is limited. Furthermore, Grad-CAM produces diffuse heatmaps that indicate various areas but fail to link the activations to specific opcodes. This limitation restricts its use for debugging or forensic purposes, particularly in our setting, as illustrated in 17 **??**. These limitations underscore the need for alternative techniques. Integrated Gradients (IG), which contribute to the prediction by

Fig. 17 Grad-CAM diffused heatmap

adding gradients along a linear path from the baseline input to the actual input, were employed to lessen this **??**. Comparatively, IG offers Grad-CAM with feature-level attributions, which enable us to determine which RGB bar opcode is influencing the model's decision. This made it possible to isolate important opcode bars that are closely linked to reentrancy vulnerability using interpretability.

(d) ***Experimental Results***

Using an opcode-based image representation, our proposed 1D CNN model yields encouraging results in both interpretability and vulnerability detection for Ethereum-based smart contracts. The framework eliminates the requirements for source code, bytecode, high-dimensional feature engineering, and manual static analysis by working directly on classified image representations, including both model and image representations. The outcomes shown in Table 15 indicate that our lightweight architecture is capable of accurately differentiating between vulnerable contracts and those that are not. The model's excellent accuracy was remarkable.

This approach's primary advantage is its lightweight design. Unlike traditional models, it leverages the inherent sequential nature of opcode data rather than complex 2D convolutional structures, which are often employed in computer vision tasks involving image-like inputs. This approach not only lowers computational overhead but also demonstrates its strong suitability for opcode-based strip representations, maintaining semantic alignment with the linear execution flow of smart contract opcodes. Despite its simplicity, the model achieves competitive results when compared to complex architectures examined in the related literature.

Beyond performance, the framework's goal is to use XAI approaches to guarantee decision-making transparency. The Integrated Gradients, a gradient-based attribution technique, quantify and illustrate the portion of the sequence flow's opcode that has the most impact on the model forecast, shown in Figure 18.

Accuracy	**Precision**	**Recall**	**F1 Score**
97.62%	96.84%	96.55%	96.69%

Table 15 Model performance on dataset

The top ten RGB opcode representations that affect the categorization choice

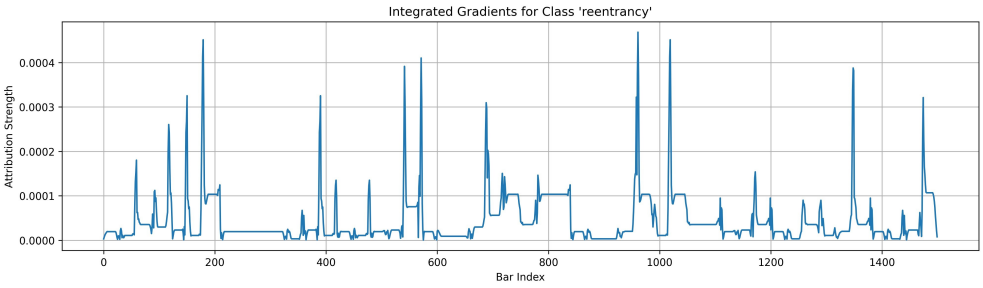

Fig. 18 Integrated Gradients Plot

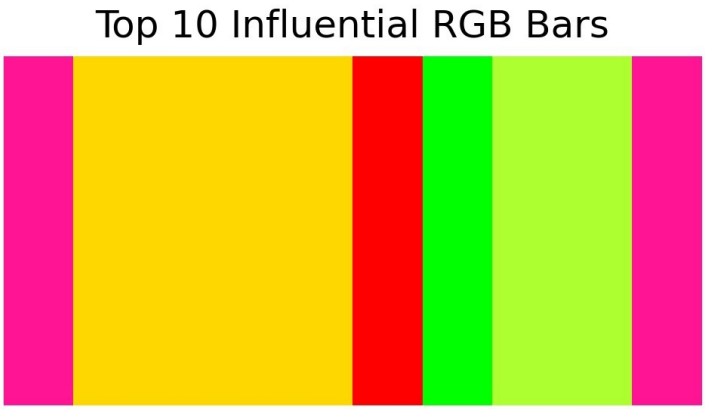

Fig. 19 Top 10 influential RGB Bars

are also shown in Figure 19 20, which makes it easy to comprehend which opcodes are impacted and where in the sequential series they are located. Because of its interpretability, auditors can link the prediction to particular opcodes, providing transparency and useful information about which instructions went into the classification.

Overall, our findings conclusively demonstrate that a comprehensive model can produce a cutting-edge detection framework with transparency and interoperability when combined with efficient attribution techniques. This makes it ideal for practical uses in smart contract security pipelines.

Fig. 20 Influential Bars Positions

12.3.2 Conclusion

In order to uncover reentrancy vulnerabilities in Ethereum smart contracts without utilizing the smart contract source code, we presented an explainable DL framework in our third paper, by utilizing the sequence-based RGB visual representation generated by smart contract bytecode and the lightweight 1D CNN. We classify smart contract vulnerabilities and identify the key trends that let the model classify data intelligently. The model is suitable for practical blockchain security applications, as it exhibits excellent accuracy while maintaining a low computational cost.

We particularly utilized XAI techniques, specifically Integrated Gradients, to assign model predictions to the individual input tokens in order to overcome the interpretability challenge with typical deep learning approaches. This makes it possible to interpret code at the opcode level, which helps us find important opcodes that affect the vulnerability assessments. By providing a cohesive solution that improves interpretability and efficacy, the work closes a significant gap between vulnerability identification and explainability in smart contract analysis. Additionally, it opens up new avenues for software security technologies that are reliable, auditable, and accurate.

We want to expand the framework in future improvements to support and handle multiple vulnerability detection for additional significant vulnerabilities. Furthermore, in order to enable the model to suggest context-aware patches for smart contract vulnerabilities, we aim to investigate not only their detection but also their correction through autonomous code repair techniques. This will facilitate a secure smart contract development pipeline from start to finish for blockchain platforms and make a significant contribution to artificial intelligence-based smart contract security.

13 Conclusion

In Industry 4.0, blockchain technology is crucial for creating decentralized and efficient industrial ecosystems. Its application in smart manufacturing, IoT, and digital supply chain improves trust and operational efficiency. This paper analyzes blockchain, covering its concepts, components, applications, and specific dfocus on Ethereum-based smart contracts, addressing security and privacy challenges and common vulnerabilities. It critiques traditional detection approaches and compares DL and ML solutions for identifying smart contract vulnerabilities. An intelligent detection framework is proposed, demonstrating significantly improved detection accuracy and adaptability for real-world applications.

We have begun by introducing a novel image-based methodology for detecting reentrancy vulnerabilities in blockchain security, leveraging a VGG16 CNN. This deep learning approach evaluates RGB representations of opcodes, overcoming limitations of traditional methods and reducing false positives and negatives. Notably, it does not require source code access, making it highly applicable in practical scenarios. Achieving an impressive 99.07% accuracy in detecting reentrancy vulnerabilities, this method offers a reliable solution for smart contract analysis, demonstrating its effectiveness as a valuable tool for blockchain security professionals and developers.

Current error-detection methods for smart contracts face challenges due to their reliance on static analysis and limited test coverage. To enhance effectiveness, advanced techniques such as dynamic analysis, machine learning, and comprehensive test suites are necessary to address a range of vulnerabilities. Scalability issues arise from the increasing number of contracts, as resource-intensive methods such as deep learning strain computational resources. Additionally, interpreting deep learning results complicates real-time detection in the fast-paced blockchain environment. The absence of standardized datasets hinders the validation of new methods, suggesting that future research should prioritize improving vulnerability detection to enhance the robustness of smart contract identification techniques.

Furthermore, to address the limitations, we present a novel image-based method for identifying multiple vulnerabilities in Ethereum smart contracts, rather than relying solely on reentrancy, thereby bypassing reliance on source code and complex rule-based systems. This approach converts bytecode into RGB-encoded images for analysis with a customized convolutional neural network (CNN), achieving a model size of 380 KB. Experimental results show high accuracy (94.12%) and F1-score (94.41%) across four benchmark datasets, indicating its effectiveness for constrained IoT environments. A key advantage of this method is that it preserves all opcode structure during image generation, ensuring no loss of semantic information. This flexibility to adapt to evolving threats distinguishes it from existing techniques, making it suitable for dynamic environments without relying on outdated security practices.

Deep Learning (DL) and Machine Learning (ML) techniques have improved current detection methods, including static and dynamic analytic tools, though they often yield high false-positive rates and lack interpretability. An interpretable One-dimensional Convolutional Neural Network (1D CNN) using integrated Gradients within an Explainable AI (XAI) framework has been proposed to address these issues.

To efficiently extract features while preserving contract semantics, this method interprets smart contract opcodes as RGB-encoded sequences. Its opcode-level attributions enhance transparency, making smart contract analysis more understandable and trustworthy, which is vital for researchers and AI developers seeking reliable tools.

Multiple vulnerability detection for additional significant vulnerabilities is essential for comprehensive security. Future work will focus on expanding this framework to support and handle multiple vulnerabilities simultaneously, which is crucial for cybersecurity professionals and blockchain experts aiming for robust smart contract security.

Overall, this work contributes to strengthening the security and resilience of blockchain-based systems by bridging the gap between automation and trust. While blockchain and smart contracts offer transformative potential, their safe deployment requires continous advancements in intelligent security mechanisms. Future research should focus on improving explainability, real-time detection, and cross-platform compatibility to further enhance the robustness of decentralized industrial systems.

References

[1] (2000) Timed Commitments. In: Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg, p 236–254, https://doi.org/10.1007/3-540-44598-6_15, URL http://link.springer.com/10.1007/3-540-44598-6_15, iSSN: 0302-9743

[2] Abiri R, Rizan N, Balasundram SK, et al (2023) Application of digital technologies for ensuring agricultural productivity. Heliyon 9(12)

[3] Abri F, Siami-Namini S, Khanghah MA, et al (2019) Can Machine/Deep Learning Classifiers Detect Zero-Day Malware with High Accuracy? In: 2019 IEEE International Conference on Big Data (Big Data). IEEE, Los Angeles, CA, USA, pp 3252–3259, https://doi.org/10.1109/bigdata47090.2019.9006514, URL https://ieeexplore.ieee.org/document/9006514/

[4] Ahmad N, Shah JH, Khan MA, et al (2023) A novel framework of multi-class skin lesion recognition from dermoscopic images using deep learning and explainable AI. Frontiers in Oncology 13. https://doi.org/10.3389/fonc.2023.1151257, URL https://www.frontiersin.org/articles/10.3389/fonc.2023.1151257/full, publisher: Frontiers Media SA

[5] Ahmad RW, Hasan H, Yaqoob I, et al (2021) Blockchain for aerospace and defense: Opportunities and open research challenges. Computers & Industrial Engineering 151:106982

[6] Ahmed S, Kaiser MS, Shahadat Hossain M, et al (2025) A Comparative Analysis of LIME and SHAP Interpreters With Explainable ML-Based Diabetes Predictions. IEEE Access 13:37370–37388. https://doi.org/10.1109/access.2024.3422319, URL https://ieeexplore.ieee.org/document/10583856/,

publisher: Institute of Electrical and Electronics Engineers (IEEE)

[7] Albert E, Gordillo P, Livshits B, et al (2018) Ethir: A framework for high-level analysis of ethereum bytecode. In: International symposium on automated technology for verification and analysis, Springer, pp 513–520

[8] Aldyaflah IM, Zhao W, Upadhyay H, et al (2023) The design and implementation of a secure datastore based on ethereum smart contract. applied sciences 13(9):5282

[9] Alharby M, van Moorsel A (2017) Blockchain-based Smart Contracts: A Systematic Mapping Study https://doi.org/10.48550/ARXIV.1710.06372, URL https://arxiv.org/abs/1710.06372, publisher: arXiv Version Number: 1

[10] Ali MS, Vecchio M, Pincheira M, et al (2018) Applications of blockchains in the internet of things: A comprehensive survey. IEEE Communications Surveys & Tutorials 21(2):1676–1717

[11] Amann M, K. Roehrich J, Eßig M, et al (2014) Driving sustainable supply chain management in the public sector: The importance of public procurement in the european union. Supply Chain Management: An International Journal 19(3):351–366

[12] Ashizawa N, Yanai N, Cruz JP, et al (2021) Eth2Vec: Learning Contract-Wide Code Representations for Vulnerability Detection on Ethereum Smart Contracts. In: Proceedings of the 3rd ACM International Symposium on Blockchain and Secure Critical Infrastructure. ACM, Virtual Event Hong Kong, pp 47–59, https://doi.org/10.1145/3457337.3457841, URL https://dl.acm.org/doi/10.1145/3457337.3457841

[13] Ashouri M (2020) Etherolic: a practical security analyzer for smart contracts. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing, pp 353–356

[14] Ayare AA, Jadhav VA, Banatwala MK, et al (2025) A systematic review on blockchain-based framework for storing educational records using interplanetary file system. Cureus Journals 2(1)

[15] Bai Y, Lee S, Seo SH (2025) A survey on directed acyclic graph-based blockchain in smart mobility. Sensors 25(4):1108

[16] Balcerzak AP, Nica E, Rogalska E, et al (2022) Blockchain technology and smart contracts in decentralized governance systems. Administrative Sciences 12(3):96

[17] Baralla G, Ibba G, Tonelli R (2024) Assessing github copilot in solidity development: Capabilities, testing, and bug fixing. IEEE Access

[18] BEKTAŞ BC (2023) Vulnerability detection on solidity smart contracts by using convolutional neural networks

[19] Bhat SA, Huang NF, Sofi IB, et al (2021) Agriculture-food supply chain management based on blockchain and iot: A narrative on enterprise blockchain interoperability. Agriculture 12(1):40

[20] Bhattarai P, Thakuri DS, Nie Y, et al (2024) Explainable ai-based deep-shap for mapping the multivariate relationships between regional neuroimaging biomarkers and cognition. European journal of radiology 174:111403

[21] Bhutta MNM, Khwaja AA, Nadeem A, et al (2021) A survey on blockchain technology: Evolution, architecture and security. Ieee Access 9:61048–61073

[22] Biswas K, Muthukkumarasamy V (2016) Securing smart cities using blockchain technology. In: 2016 IEEE 18th international conference on high performance computing and communications; IEEE 14th international conference on smart city; IEEE 2nd international conference on data science and systems (HPCC/SmartCity/DSS), IEEE, pp 1392–1393

[23] Biswas S, Sharif K, Li F, et al (2018) A scalable blockchain framework for secure transactions in iot. IEEE Internet of Things Journal 6(3):4650–4659

[24] Borys K, Schmitt YA, Nauta M, et al (2023) Explainable ai in medical imaging: An overview for clinical practitioners–beyond saliency-based xai approaches. European journal of radiology 162:110786

[25] Boulila W, Driss M, Alshanqiti E, et al (2022) Weight initialization techniques for deep learning algorithms in remote sensing: Recent trends and future perspectives. Advances on Smart and Soft Computing pp 477–484

[26] Breidenbach L, Daian P, Juels A, et al (2017) An in-depth look at the parity multisig bug. Hacking, Distributed, July

[27] Brown RG, Carlyle J, Grigg I, et al (2016) Corda: an introduction. R3 CEV, August 1(15):14

[28] Cai J, Li B, Zhang J, et al (2023) Combine sliced joint graph with graph neural networks for smart contract vulnerability detection. Journal of Systems and Software 195:111550

[29] Cai W, Wang Z, Ernst JB, et al (2018) Decentralized applications: The blockchain-empowered software system. IEEE access 6:53019–53033

[30] Chaganti R, Boppana RV, Ravi V, et al (2022) A comprehensive review of denial of service attacks in blockchain ecosystem and open challenges. IEEE Access 10:96538–96555

[31] Chanda T, Hauser K, Hobelsberger S, et al (2024) Dermatologist-like explainable ai enhances trust and confidence in diagnosing melanoma. Nature Communications 15(1):524

[32] Chen J, Cai T, He W, et al (2020) A blockchain-driven supply chain finance application for auto retail industry. Entropy 22(1):95

[33] Chen J, Xia X, Lo D, et al (2022) DefectChecker: Automated Smart Contract Defect Detection by Analyzing EVM Bytecode. IEEE Transactions on Software Engineering 48(7):2189–2207. https://doi.org/10.1109/tse.2021.3054928, URL https://ieeexplore.ieee.org/document/9337195/, publisher: Institute of Electrical and Electronics Engineers (IEEE)

[34] Chen T, Li X, Luo X, et al (2017) Under-Optimized Smart Contracts Devour Your Money. https://doi.org/10.48550/ARXIV.1703.03994, URL https://arxiv.org/abs/1703.03994, version Number: 2

[35] Chen T, Li Z, Zhu Y, et al (2020) Understanding ethereum via graph analysis. ACM Transactions on Internet Technology (TOIT) 20(2):1–32

[36] Chen W, Zheng Z, Cui J, et al (2018) Detecting ponzi schemes on ethereum: Towards healthier blockchain technology. In: Proceedings of the 2018 world wide web conference, pp 1409–1418

[37] Chinen Y, Yanai N, Cruz JP, et al (2021) RA: A Static Analysis Tool for Analyzing Re-Entrancy Attacks in Ethereum Smart Contracts. Journal of Information Processing 29(0):537–547. https://doi.org/10.2197/ipsjjip.29.537, URL

https://www.jstage.jst.go.jp/article/ipsjjip/29/0/29_537/_article, publisher: Information Processing Society of Japan

[38] Cho S, Fontugne R, Cho K, et al (2019) Bgp hijacking classification. In: 2019 Network Traffic Measurement and Analysis Conference (TMA), IEEE, pp 25–32

[39] Chu S, Wang S (2018) The curses of blockchain decentralization. arXiv preprint arXiv:181002937

[40] Contro F, Crosara M, Ceccato M, et al (2021) EtherSolve: Computing an Accurate Control-Flow Graph from Ethereum Bytecode. https://doi.org/10.48550/arXiv.2103.09113, URL http://arxiv.org/abs/2103.09113, arXiv:2103.09113 [cs]

[41] Correa Tavares E, Meirelles FdS, Tavares EC, et al (2021) Blockchain in the amazon: creating public value and promoting sustainability. Information Technology for Development 27(3):579–598

[42] Dai HN, Zheng Z, Zhang Y (2019) Blockchain for internet of things: A survey. IEEE internet of things journal 6(5):8076–8094

[43] Dai W, Liu J, Zhou Y, et al (2024) Prbfpt: A practical redactable blockchain framework with a public trapdoor. IEEE Transactions on Information Forensics and Security 19:2425–2437

[44] Daniel E, Wilson H (2002) Adoption intentions and benefits realised: a study of e-commerce in uk smes. Journal of Small Business and Enterprise Development 9(4):331–348

[45] Dannen C (2017) Solidity programming. In: Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners. Springer, p 69–88

[46] Davagdorj K, Bae JW, Pham VH, et al (2021) Explainable artificial intelligence based framework for non-communicable diseases prediction. Ieee Access 9:123672–123688

[47] De Baets C, Suleiman B, Chitizadeh A, et al (2024) Vulnerability detection in smart contracts: A comprehensive survey. arXiv preprint arXiv:240707922

[48] Deng W, Wei H, Huang T, et al (2023) Smart Contract Vulnerability Detection Based on Deep Learning and Multimodal Decision Fusion. Sensors 23(16):7246. https://doi.org/10.3390/s23167246, URL https://www.mdpi.com/1424-8220/23/16/7246, publisher: MDPI AG

[49] Dhole SV, Chougule SR (2025) FastColitisDetector-XAI: An efficient AI model utilizing sparse Autoencoder with explainable AI for ulcerative colitis diagnosis. MethodsX 14:103356. https://doi.org/10.1016/j.mex.2025.103356, URL https://linkinghub.elsevier.com/retrieve/pii/S221501612500202X, publisher: Elsevier BV

[50] Di Pirro M (2018) How solid is solidity? an in-dept study of solidity's type safety.

[51] Di Vaio A, Varriale L (2020) Blockchain technology in supply chain management for sustainable performance: Evidence from the airport industry. International Journal of Information Management 52:102014

[52] Dorri A, Kanhere SS, Jurdak R (2016) Blockchain in internet of things: challenges and solutions. arXiv preprint arXiv:160805187

[53] Dorri A, Kanhere SS, Jurdak R, et al (2017) Blockchain for iot security and privacy: The case study of a smart home. In: 2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops), IEEE, pp 618–623

[54] Douceur JR (2002) The sybil attack. In: International workshop on peer-to-peer systems, Springer, pp 251–260

[55] Du S, Li T, Yang Y, et al (2019) Deep air quality forecasting using hybrid deep learning framework. IEEE Transactions on Knowledge and Data Engineering 33(6):2412–2424

[56] Dwivedi AD, Srivastava G, Dhar S, et al (2019) A decentralized privacy-preserving healthcare blockchain for iot. Sensors 19(2):326

[57] Efanov D, Roschin P (2018) The all-pervasiveness of the blockchain technology. Procedia computer science 123:116–121

[58] El-Ghamry A, Gaber T, Mohammed KK, et al (2023) Optimized and Efficient Image-Based IoT Malware Detection Method. Electronics 12(3):708. https://doi.org/10.3390/electronics12030708, URL https://www.mdpi.com/2079-9292/12/3/708, publisher: MDPI AG

[59] Eshghie M, Artho C, Gurov D (2021) Dynamic vulnerability detection on smart contracts using machine learning. In: Proceedings of the 25th international conference on evaluation and assessment in software engineering, pp 305–312

[60] Evirgen N, Wang R, Chen X (2024) From text to pixels: Enhancing user understanding through text-to-image model explanations. In: Proceedings of the 29th International Conference on Intelligent User Interfaces, pp 74–87

[61] Ezzat SK, Saleh YN, Abdel-Hamid AA (2022) Blockchain oracles: State-of-the-art and research directions. IEEE Access 10:67551–67572

[62] Feist J, Grieco G, Groce A (2019) Slither: A Static Analysis Framework for Smart Contracts. In: 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB). IEEE, Montreal, QC, Canada, pp 8–15, https://doi.org/10.1109/wetseb.2019.00008, URL https://ieeexplore.ieee.org/document/8823898/

[63] Fiquaro MA, Zahilah R, Othman SH, et al (2021) Vaccination system using blockchain technology: a prototype development. In: 2021 3rd International Cyber Resilience Conference (CRC), IEEE, pp 1–6

[64] Forissier T (2024) Eveilm: Evm bytecode obfuscation

[65] Fortino G, Greco C, Guzzo A, et al (2026) Detecting ethereum smart contract vulnerabilities via bytecode image analysis. IEEE Internet of Things Journal pp 1–1. https://doi.org/10.1109/JIOT.2026.3652734

[66] Fu M, Wu L, Hong Z, et al (2019) A Critical-Path-Coverage-Based Vulnerability Detection Method for Smart Contracts. IEEE Access 7:147327–147344. https://doi.org/10.1109/access.2019.2947146, URL https://ieeexplore.ieee.org/document/8867880/, publisher: Institute of Electrical and Electronics Engineers (IEEE)

[67] Ghaleb A, Pattabiraman K (2020) How effective are smart contract analysis tools? evaluating smart contract static analysis tools using bug injection. In: Proceedings of the 29th ACM SIGSOFT international symposium on software testing and analysis, pp 415–427

[68] Ghorbian M, Ghobaei-Arani M (2025) Key concepts and principles of blockchain technology. arXiv preprint arXiv:250111707

[69] Gomez C, Wang R, Breininger K, et al (2024) Explainable ai enhances glaucoma referrals, yet the human-ai team still falls short of the ai alone. arXiv preprint arXiv:240711974

[70] Gong X, Liu E, Wang R (2020) Blockchain-based iot application using smart contracts: Case study of m2m autonomous trading. In: 2020 5th international conference on computer and communication systems (ICCCS), IEEE, pp 781–785

[71] Granata D, Rak M, Palmiero P, et al (2024) A methodology for vulnerability assessment and threat modelling of an e-voting platform based on ethereum blockchain. IEEE Access

[72] Grech A, Camilleri AF (2017) Blockchain in education. Publications Office of the European Union

[73] Grishchenko I, Maffei M, Schneidewind C (2018) A semantic framework for the security analysis of ethereum smart contracts. In: International conference on principles of security and trust, Springer, pp 243–269

[74] Guidi B, Michienzi A, Ricci L (2020) Steem blockchain: Mining the inner structure of the graph. IEEE Access 8:210251–210266

[75] Hassan MU, Rehmani MH, Chen J (2019) Privacy preservation in blockchain based iot systems: Integration issues, prospects, challenges, and future research directions. Future Generation Computer Systems 97:512–529

[76] He Y, Dong H, Wu H, et al (2023) Formal Analysis of Reentrancy Vulnerabilities in Smart Contract Based on CPN. Electronics 12(10):2152. https://doi.org/10.3390/electronics12102152, URL https://www.mdpi.com/2079-9292/12/10/2152, publisher: MDPI AG

[77] Hirai Y (2017) Defining the ethereum virtual machine for interactive theorem provers. In: International conference on financial cryptography and data security, Springer, pp 520–535

[78] Ho GT, Tang YM, Tsang KY, et al (2021) A blockchain-based system to enhance aircraft parts traceability and trackability for inventory management. Expert Systems with Applications 179:115101

[79] Hongmei Z (2021) A cross-border e-commerce approach based on blockchain technology. Mobile Information Systems 2021(1):2006082

[80] Honkaranta A, Leppänen T, Costin A (2021) Towards practical cybersecurity mapping of stride and cwe—a multi-perspective approach. In: 2021 29th Conference of Open Innovations Association (FRUCT), IEEE, pp 150–159

[81] Hu H, Lei S, Sun D, et al (2024) Dual-channel reliable breast ultrasound image classification based on explainable attribution and uncertainty quantification. In: Proceedings of the 2024 6th International Conference on Image Processing and Machine Vision, pp 53–61

[82] Huang H, Chen X, Wang J (2020) Blockchain-based multiple groups data sharing with anonymity and traceability. Science China Information Sciences 63(3):130101

[83] Huang J, Kong L, Chen G, et al (2019) Towards secure industrial iot: Blockchain system with credit-based consensus mechanism. IEEE Transactions

on Industrial Informatics 15(6):3680–3689

[84] Huang THD (2018) Hunting the Ethereum Smart Contract: Color-inspired Inspection of Potential Attacks. https://doi.org/10.48550/arXiv.1807.01868, URL http://arxiv.org/abs/1807.01868, arXiv:1807.01868 [cs]

[85] Huang THD (2018) Hunting the ethereum smart contract: Color-inspired inspection of potential attacks. arXiv preprint arXiv:180701868

[86] Huang Y, Bian Y, Li R, et al (2019) Smart Contract Security: A Software Lifecycle Perspective. IEEE Access 7:150184–150202. https://doi.org/10.1109/access.2019.2946988, URL https://ieeexplore.ieee.org/document/8864988/, publisher: Institute of Electrical and Electronics Engineers (IEEE)

[87] Hwang SJ, Choi SH, Shin J, et al (2022) CodeNet: Code-Targeted Convolutional Neural Network Architecture for Smart Contract Vulnerability Detection. IEEE Access 10:32595–32607. https://doi.org/10.1109/access.2022.3162065, URL https://ieeexplore.ieee.org/document/9740682/, publisher: Institute of Electrical and Electronics Engineers (IEEE)

[88] Ishikawa Sn, Todo M, Taki M, et al (2023) Example-based explainable ai and its application for remote sensing image classification. International Journal of Applied Earth Observation and Geoinformation 118:103215

[89] Islam MR, Nahiduzzaman M, Goni MOF, et al (2022) Explainable Transformer-Based Deep Learning Model for the Detection of Malaria Parasites from Blood Cell Images. Sensors 22(12):4358. https://doi.org/10.3390/s22124358, URL https://www.mdpi.com/1424-8220/22/12/4358, publisher: MDPI AG

[90] Iuliano G, Di Nucci D (2024) Smart contract vulnerabilities, tools, and benchmarks: An updated systematic literature review. arXiv preprint

arXiv:241201719

[91] Jamsheed SB, Suhailam P (????) Blockchain and ai: Enhancing personalized learning. In: Blockchain and AI in Shaping the Modern Education System. CRC Press, p 1–25

[92] Jezek K (2021) Ethereum data structures. arXiv preprint arXiv:210805513

[93] Jiang B, Liu Y, Chan WK (2018) Contractfuzzer: Fuzzing smart contracts for vulnerability detection. In: Proceedings of the 33rd ACM/IEEE international conference on automated software engineering, pp 259–269

[94] Jiang L, Dong K (2020) Credibility modelling of e-commerce networks based on block-chain and massive data mining. In: 2020 Fourth International Conference on Inventive Systems and Control (ICISC), IEEE, pp 441–444

[95] Jiang T, Fang H, Wang H (2018) Blockchain-based internet of vehicles: Distributed network architecture and performance analysis. IEEE Internet of Things Journal 6(3):4640–4649

[96] Joshi AP, Han M, Wang Y (2018) A survey on security and privacy issues of blockchain technology. Mathematical foundations of computing 1(2)

[97] Juels A, Kosba A, Shi E (2016) The ring of gyges: Investigating the future of criminal smart contracts. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp 283–295

[98] Kang J, Yu R, Huang X, et al (2017) Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains. IEEE transactions on industrial informatics 13(6):3154–3164

[99] Kaur R, Ali A, Faisal M (2022) Smart contracts: the self-executing contracts. In: Blockchain. Chapman and Hall/CRC, p 35–49

[100] Kawaguchi N (2019) Application of blockchain to supply chain: Flexible blockchain technology. Procedia Computer Science 164:143–148

[101] Khan N, Aljoaey H, Tabassum M, et al (2022) Proposed model for secured data storage in decentralized cloud by blockchain ethereum. Electronics 11(22):3686

[102] Khan SH, Alahmadi TJ, Ullah W, et al (2023) A new deep boosted cnn and ensemble learning based iot malware detection. Computers & Security 133:103385

[103] Khan ZA, Namin AS (2024) A survey of vulnerability detection techniques by smart contract tools. IEEE Access 12:70870–70910

[104] Khaqqi KN, Sikorski JJ, Hadinoto K, et al (2018) Incorporating seller/buyer reputation-based system in blockchain-enabled emission trading application. Applied energy 209:8–19

[105] Kiani R, Sheng VS (2024) Automated repair of smart contract vulnerabilities: A systematic literature review. Electronics 13(19):3942

[106] Kim D, Chung J, Choi J, et al (2022) Accurate auto-labeling of chest x-ray images based on quantitative similarity to an explainable ai model. Nature communications 13(1):1867

[107] Kim KB, Lee J (2020) Automated Generation of Test Cases for Smart Contract Security Analyzers. IEEE Access 8:209377–209392. https://doi.org/10.1109/access.2020.3039990, URL https://ieeexplore.ieee.org/document/9268135/, publisher: Institute of Electrical and Electronics Engineers (IEEE)

[108] Kim TH, Reeves D (2020) A survey of domain name system vulnerabilities and attacks. Journal of Surveillance, Security and Safety 1(1):34–60

[109] Kingma DP (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980

[110] Knirsch F, Unterweger A, Engel D (2018) Privacy-preserving blockchain-based electric vehicle charging with dynamic tariff decisions. Computer Science-Research and Development 33(1):71–79

[111] Kosba A, Miller A, Shi E, et al (2016) Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. In: 2016 IEEE Symposium on Security and Privacy (SP). IEEE, San Jose, CA, https://doi.org/10.1109/sp.2016.55, URL http://ieeexplore.ieee.org/document/7546538/

[112] Kostopoulos N, Stamatiou YC, Halkiopoulos C, et al (2025) Blockchain applications in the military domain: A systematic review. Technologies 13(1):23

[113] Krichen M (2023) Strengthening the security of smart contracts through the power of artificial intelligence. Computers 12(5):107

[114] Krichen M, Ammi M, Mihoub A, et al (2022) Blockchain for modern applications: A survey. Sensors 22(14):5274

[115] Krupp J, Rossow C (2018) {teEther}: Gnawing at ethereum to automatically exploit smart contracts. In: 27th USENIX security symposium (USENIX Security 18), pp 1317–1333

[116] Kshetri N (2017) Can blockchain strengthen the internet of things? IT Professional 19(4):68–72. https://doi.org/10.1109/MITP.2017.3051335

[117] Kumar A, Liu R, Shan Z (2020) Is blockchain a silver bullet for supply chain management? technical challenges and research opportunities. Decision Sciences 51(1):8–37

[118] Kushwaha SS, Joshi S, Singh D, et al (2022) Ethereum smart contract analysis tools: A systematic review. Ieee Access 10:57037–57062

[119] Kushwaha SS, Joshi S, Singh D, et al (2022) Systematic review of security vulnerabilities in ethereum blockchain smart contract. Ieee Access 10:6605–6621

[120] Labrador M, Hou W (2019) Implementing blockchain technology in the internet of vehicle (iov). In: 2019 international conference on intelligent computing and its emerging applications (ICEA), IEEE, pp 5–10

[121] Lai J (2019) Research on cross-border e-commerce logistics supply under block chain. In: 2019 International Conference on Computer Network, Electronic and Automation (ICCNEA), IEEE, pp 214–218

[122] Leng K, Bi Y, Jing L, et al (2018) Retracted: Research on agricultural supply chain system with double chain architecture based on blockchain technology

[123] Li F, Lam KY, Chen HH, et al (2019) Spectral efficiency enhancement in satellite mobile communications: A game-theoretical approach. IEEE Wireless Communications 27(1):200–205

[124] Li L, Liu Y, Sun G, et al (2023) Smart contract vulnerability detection based on automated feature extraction and feature interaction. IEEE Transactions on Knowledge and Data Engineering 36(9):4916–4929

[125] Li Z, Lu S, Zhang R, et al (2023) Vulhunter: Hunting vulnerable smart contracts at evm bytecode-level via multiple instance learning. IEEE Transactions on Software Engineering 49(11):4886–4916

[126] Liao X (2024) Smart contract vulnerability detection based on dynamic and static combination. In: Proceedings of the International Conference on Digital Economy, Blockchain and Artificial Intelligence, pp 412–416

[127] Liao Z, Zheng Z, Chen X, et al (2022) Smartdagger: a bytecode-based static analysis approach for detecting cross-contract vulnerability. In: Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis, pp 752–764

[128] Lilly B, Lilly S (2021) Weaponising blockchain: military applications of blockchain technology in the us, china and russia. The RUSI Journal 166(3):46–56

[129] Lin IC, Liao TC (2017) A survey of blockchain security issues and challenges. Int J Netw Secur 19(5):653–659

[130] Linh PT, Minh Thanh T (2023) Proposing of imaging graph neural network with defined security pattern for improving smart contract vulnerability detection. ict research 70–79

[131] Liu J, Feng Y, Liu X, et al (2023) Mrm-dldet: a memory-resident malware detection framework based on memory forensics and deep neural network. Cybersecurity 6(1):21

[132] Liu K, Chen W, Zheng Z, et al (2019) A novel debt-credit mechanism for blockchain-based data-trading in internet of vehicles. IEEE Internet of Things Journal 6(5):9098–9111

[133] Liu S, Liu Z, Chen B, et al (2024) Construction and application of online learning resource incentive mechanism driven by smart contract. IEEE Access 12:37080–37092

[134] Liu X, Li B, Vernooij MW, et al (2025) Ai-based association analysis for medical imaging using latent-space geometric confounder correction. Medical Image Analysis 102:103529

[135] Liu Z, Qian P, Wang X, et al (2021) Smart contract vulnerability detection: from pure neural network to interpretable graph feature and expert pattern fusion. arXiv preprint arXiv:210609282

[136] Liu Z, Qian P, Wang X, et al (2021) Combining graph neural networks with expert knowledge for smart contract vulnerability detection. IEEE Transactions on Knowledge and Data Engineering 35(2):1296–1310

[137] Liu Z, Qian P, Yang J, et al (2023) Rethinking smart contract fuzzing: Fuzzing with invocation ordering and important branch revisiting. IEEE Transactions on Information Forensics and Security 18:1237–1251

[138] Loporchio M, Di Francesco Maesa D, Bernasconi A, et al (2024) Comparing ethereum fungible and non-fungible tokens: an analysis of transfer networks. Applied Network Science 9(1):72

[139] Lu Z, Liu W, Wang Q, et al (2018) A privacy-preserving trust model based on blockchain for vanets. Ieee Access 6:45655–45664

[140] Luu L, Chu DH, Olickel H, et al (2016) Making smart contracts smarter. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pp 254–269

[141] Ma C, Liu S, Xu G (2023) HGAT: smart contract vulnerability detection method based on hierarchical graph attention network. Journal of Cloud Computing 12(1). https://doi.org/10.1186/s13677-023-00459-x, URL https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00459-x, publisher: Springer Science and Business Media LLC

[142] Mahmud T, Barua K, Habiba SU, et al (2024) An Explainable AI Paradigm for Alzheimer's Diagnosis Using Deep Transfer Learning. Diagnostics 14(3):345. https://doi.org/10.3390/diagnostics14030345, URL https://www.mdpi.com/2075-4418/14/3/345, publisher: MDPI AG

[143] Marchesi L (2022) Software engineering practices applied to blockchain technology and decentralized applications

[144] Marino B, Juels A (2016) Setting standards for altering and undoing smart contracts. In: International Symposium on Rules and Rule Markup Languages for the Semantic Web, Springer, pp 151–166

[145] Martinovic I, Kello L, Sluganovic I (2017) Blockchains for governmental services: Design principles, applications, and case studies. Center for Technology and Global Affairs, University of Oxford

[146] McGhin T, Choo KKR, Liu CZ, et al (2019) Blockchain in healthcare applications: Research challenges and opportunities. Journal of network and computer applications 135:62–75

[147] Metcalfe W, et al (2020) Ethereum, smart contracts, dapps. Blockchain and Crypt Currency 77:77–93

[148] Mi F, Wang Z, Zhao C, et al (2021) Vscl: automating vulnerability detection in smart contracts with deep learning. In: 2021 IEEE International Conference

on Blockchain and Cryptocurrency (ICBC), IEEE, pp 1–9

[149] Mi F, Zhao C, Wang Z, et al (2023) An automated vulnerability detection framework for smart contracts. Distributed Ledger Technologies: Research and Practice

[150] Mircea M, Stoica M, Ghilic-Micu B (2022) Analysis of the impact of blockchain and internet of things (biot) on public procurement. IEEE Access 10:63353–63374

[151] Mishra D, Phansalkar S (2025) Blockchain security in focus: A comprehensive investigation into threats, smart contract security, cross-chain bridges, vulnerabilities detection tools & techniques. IEEE Access

[152] Molnar A, Janssen M, Weerakkody V (2015) E-government theories and challenges: findings from a plenary expert panel. In: Proceedings of the 16th Annual International Conference on Digital Government Research, pp 160–166

[153] Morgan H, Elgendy A, Said A, et al (2024) Enhanced lithological mapping in arid crystalline regions using explainable ai and multi-spectral remote sensing data. Computers & Geosciences 193:105738

[154] Mossberg M, Manzano F, Hennenfent E, et al (2019) Manticore: A user-friendly symbolic execution framework for binaries and smart contracts. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, pp 1186–1189

[155] Nakamoto S (2008) Bitcoin: A peer-to-peer electronic cash system

[156] Nikolić I, Kolluri A, Sergey I, et al (2018) Finding the greedy, prodigal, and suicidal contracts at scale. In: Proceedings of the 34th annual computer

security applications conference, pp 653–663

[157] Oppong SA (2025) The effectiveness of green public procurement policies on promoting regional sustainable development

[158] Paik HY, Xu X, Bandara HD, et al (2019) Analysis of data management in blockchain-based systems: From architecture to governance. Ieee Access 7:186091–186107

[159] Palma LM, Vigil MA, Pereira FL, et al (2019) Blockchain and smart contracts for higher education registry in brazil. International Journal of Network Management 29(3):e2061

[160] Park D, Zhang Y, Saxena M, et al (2018) A formal verification tool for ethereum vm bytecode. In: Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering, pp 912–915

[161] Park LW, Lee S, Chang H (2018) A sustainable home energy prosumer-chain methodology with energy tags over the blockchain. Sustainability 10(3):658

[162] Parvaneh P, Strand L (2023) Post-trade: An examination of blockchain technology's capabilities for future development

[163] Peelam MS, Chaurasia BK, Sharma AK, et al (2024) Unlocking the potential of interconnected blockchains: A comprehensive study of cosmos blockchain interoperability. IEEE Access

[164] Portmann E (2018) Rezension blockchain: Blueprint for a new economy "

[165] Praitheeshan P, Pan L, Yu J, et al (2019) Security Analysis Methods on Ethereum Smart Contract Vulnerabilities: A Survey. https://doi.org/10.48550/

ARXIV.1908.08605, URL https://arxiv.org/abs/1908.08605, version Number: 3

[166] Priyanka R, Gajendran G, Boulaaras S, et al (2025) Pediapulmodx: Harnessing cutting edge preprocessing and explainable ai for pediatric chest x-ray classification with densenet121. Results in Engineering 25:104320

[167] Qian P, Liu Z, He Q, et al (2020) Towards Automated Reentrancy Detection for Smart Contracts Based on Sequential Models. IEEE Access 8:19685–19695. https://doi.org/10.1109/access.2020.2969429, URL https://ieeexplore.ieee.org/document/8970384/, publisher: Institute of Electrical and Electronics Engineers (IEEE)

[168] Qian P, Liu Z, Yin Y, et al (2023) Cross-modality mutual learning for enhancing smart contract vulnerability detection on bytecode. In: Proceedings of the ACM web conference 2023, pp 2220–2229

[169] Qian S, Ning H, He Y, et al (2022) Multi-label vulnerability detection of smart contracts based on bi-lstm and attention mechanism. Electronics 11(19):3260

[170] Qin SJ, Liu Z, Ren F, et al (2022) Smart contract vulnerability detection based on critical combination path and deep learning. In: Proceedings of the 2022 12th International Conference on Communication and Network Security, pp 219–226

[171] Rahman MA, Masum MI, Hasib KM, et al (2024) GliomaCNN: An Effective Lightweight CNN Model in Assessment of Classifying Brain Tumor from Magnetic Resonance Images Using Explainable AI. Computer Modeling in Engineering & Sciences 140(3):2425–2448. https://doi.org/10.32604/cmes.2024.050760, URL https://www.techscience.com/CMES/v140n3/57248,

publisher: Tech Science Press

[172] Raval H, Chaki J (2024) Ensemble transfer learning meets explainable AI: A deep learning approach for leaf disease detection. Ecological Informatics 84:102925. https://doi.org/10.1016/j.ecoinf.2024.102925, URL https://linkinghub.elsevier.com/retrieve/pii/S1574954124004679, publisher: Elsevier BV

[173] Rehman KU, Andleeb S, Ashfaq M, et al (2023) Blockchain-enabled smart agriculture: Enhancing data-driven decision making and ensuring food security. Journal of Cleaner Production 427:138900

[174] Rehman Z, Gregory MA, Gondal I, et al (2025) Eclipse attacks in blockchain networks: detection, prevention, and future directions. IEEE Access

[175] Rijanto A (2021) Business financing and blockchain technology adoption in agroindustry. Journal of Science and Technology Policy Management 12(2):215–235

[176] Rossini M, Ferretti S (2025) A comparative evaluation of deep learning techniques for smart contract vulnerability classification. Distributed Ledger Technologies: Research and Practice 4(2):1–20

[177] Saad M, Spaulding J, Njilla L, et al (2020) Exploring the attack surface of blockchain: A comprehensive survey. IEEE Communications Surveys & Tutorials 22(3):1977–2008

[178] Sarmah SS (2018) Understanding blockchain technology. Computer science and engineering 8(2):23–29

[179] Senanayake J, Kalutarage H, Al-Kadri MO, et al (2023) Android source code vulnerability detection: a systematic literature review. ACM Computing Surveys 55(9):1–37

[180] Settey T, Gnap J, Beňová D, et al (2021) The growth of e-commerce due to covid-19 and the need for urban logistics centers using electric vehicles: Bratislava case study. Sustainability 13(10):5357

[181] Shamrat FJM, Idris MYI, Zhou X, et al (2024) Pollennet: A novel architecture for high precision pollen grain classification through deep learning and explainable ai. Heliyon 10(19)

[182] Sharma A, Kaur S, Singh M (2021) A comprehensive review on blockchain and internet of things in healthcare. Transactions on Emerging Telecommunications Technologies 32(10):e4333

[183] Sharma A, Bahl S, Bagha AK, et al (2022) Blockchain technology and its applications to combat covid-19 pandemic. Research on Biomedical Engineering 38(1):173–180

[184] Sharma N, Sharma S, et al (2022) A survey of mythril, a smart contract security analysis tool for evm bytecode. Indian Journal of Natural Sciences 13(75):51003–51010

[185] Sharma S, Mohan S (2016) Cognitive radio adhoc vehicular network (cravenet): Architecture, applications, security requirements and challenges. In: 2016 IEEE international conference on advanced networks and telecommunications systems (ANTS), IEEE, pp 1–6

[186] Shen L, Sun Y, Yu Z, et al (2024) On efficient training of large-scale deep learning models. ACM Computing Surveys 57(3):1–36

[187] Shiri Harzevili N, Boaye Belle A, Wang J, et al (2025) A Systematic Literature Review on Automated Software Vulnerability Detection Using Machine Learning. ACM Computing Surveys 57(3):1–36. https://doi.org/10.1145/3699711, URL https://dl.acm.org/doi/10.1145/3699711, publisher: Association for Computing Machinery (ACM)

[188] Shrestha R, Nam SY (2019) Regional blockchain for vehicular networks to prevent 51% attacks. Ieee Access 7:95033–95045

[189] Shrestha R, Nam SY, Bajracharya R, et al (2020) Evolution of v2x communication and integration of blockchain for security enhancements. Electronics 9(9):1338

[190] Siegel D (2018) Understanding the dao attack (2016). URL https://www coindesk com/understanding-dao-hack-journalists 41:43

[191] Singh SK, Rathore S, Park JH (2020) Blockiotintelligence: A blockchain-enabled intelligent iot architecture with artificial intelligence. Future Generation Computer Systems 110:721–743

[192] Singhal R (2022) Recognizing Reentrancy attacks in Ethereum Smart Contracts https://doi.org/10.13140/RG.2.2.29494.78409, URL https://rgdoi.net/10.13140/RG.2.2.29494.78409, publisher: Unpublished

[193] Srinivasan P, et al (2023) Tp-detect: trigram-pixel based vulnerability detection for ethereum smart contracts. Multimedia Tools and Applications 82(23):36379–36393

[194] Sui J, Chu L, Bao H (2023) An Opcode-Based Vulnerability Detection of Smart Contracts. Applied Sciences 13(13):7721. https://doi.org/10.3390/app13137721, URL https://www.mdpi.com/2076-3417/13/13/7721, publisher:

MDPI AG

[195] Sun SF, Au MH, Liu JK, et al (2017) Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In: European Symposium on Research in Computer Security, Springer, pp 456–474

[196] Sun Y, Gu L (2021) Attention-based Machine Learning Model for Smart Contract Vulnerability Detection. Journal of Physics: Conference Series 1820(1):012004. https://doi.org/10.1088/1742-6596/1820/1/012004, URL https://iopscience.iop.org/article/10.1088/1742-6596/1820/1/012004, publisher: IOP Publishing

[197] Sunyaev A (2020) Distributed ledger technology. In: internet Computing. Springer, p 265–299

[198] Tahir U, Siyal F, Ianni M, et al (2023) Exploiting Bytecode Analysis for Reentrancy Vulnerability Detection in Ethereum Smart Contracts. In: 2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech). IEEE, Abu Dhabi, United Arab Emirates, pp 0779–0783, https://doi.org/10.1109/dasc/picom/cbdcom/cy59711.2023.10361441, URL https://ieeexplore.ieee.org/document/10361441/

[199] Tang X, Du Y, Lai A, et al (2023) Lightning Cat: A Deep Learning-based Solution for Smart Contracts Vulnerability Detection. https://doi.org/10.21203/rs.3.rs-3104649/v1, URL https://www.researchsquare.com/article/rs-3104649/v1

[200] Tanriverdí M (2024) Publiceduchain: A framework for sharing student-owned educational data on public blockchain network. IEEE Access 12:51772–51785

[201] Tapscott D, Kaplan A (2019) Blockchain revolution in education and lifelong learning. Blockchain research institute-IBM institute for business value

[202] Tikhomirov S, Voskresenskaya E, Ivanitskiy I, et al (2018) Smartcheck: Static analysis of ethereum smart contracts. In: Proceedings of the 1st international workshop on emerging trends in software engineering for blockchain, pp 9–16

[203] Torres CF, Schütte J, State R (2018) Osiris: Hunting for integer bugs in ethereum smart contracts. In: Proceedings of the 34th annual computer security applications conference, pp 664–676

[204] Torres CF, Baden M, Norvill R, et al (2020) ÆGIS: Shielding Vulnerable Smart Contracts Against Attacks. https://doi.org/10.48550/ARXIV.2003.05987, URL https://arxiv.org/abs/2003.05987, version Number: 1

[205] Tredinnick L (2019) Cryptocurrencies and the blockchain. Business Information Review 36(1):39–44

[206] Tsankov P, Dan A, Drachsler-Cohen D, et al (2018) Securify: Practical security analysis of smart contracts. In: Proceedings of the 2018 ACM SIGSAC conference on computer and communications security, pp 67–82

[207] Tsigos K, Apostolidis E, Baxevanakis S, et al (2024) Towards Quantitative Evaluation of Explainable AI Methods for Deepfake Detection. https://doi.org/10.48550/ARXIV.2404.18649, URL https://arxiv.org/abs/2404.18649, version Number: 1

[208] Wang A, Wang H, Jiang B, et al (2020) Artemis: An Improved Smart Contract Verification Tool for Vulnerability Detection. In: 2020 7th International Conference on Dependable Systems and Their Applications (DSA). IEEE, Xi'an, China, pp 173–181, https://doi.org/10.1109/dsa51864.2020.00031, URL https://ieeexplore.ieee.org/document/9331236/

[209] Wang Q, Liu Y (2023) Blockchain for public safety: A survey of techniques and applications. Journal of Safety Science and Resilience 4(4):389–395

[210] Wang S, Wang J, Song Y, et al (2022) Malware variants detection model based on mff–hdba. Applied Sciences 12(19):9593

[211] Wang W, Song J, Xu G, et al (2021) ContractWard: Automated Vulnerability Detection Models for Ethereum Smart Contracts. IEEE Transactions on Network Science and Engineering 8(2):1133–1144. https://doi.org/10.1109/tnse.2020.2968505, URL https://ieeexplore.ieee.org/document/8967006/, publisher: Institute of Electrical and Electronics Engineers (IEEE)

[212] Wang X, Zha X, Ni W, et al (2019) Survey on blockchain for internet of things. Computer Communications 136:10–29

[213] Warkentin M, Orgeron C (2020) Using the security triad to assess blockchain technology in public sector applications. International Journal of Information Management 52:102090

[214] Wei Z, Sun J, Zhang Z, et al (2023) A comparative evaluation of automated analysis tools for solidity smart contracts. arXiv preprint arXiv:231020212

[215] Wohrer M, Zdun U (2018) Smart contracts: security patterns in the ethereum ecosystem and solidity. In: 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE), IEEE, pp 2–8

[216] Wu L, Meng K, Xu S, et al (2017) Democratic centralism: a hybrid blockchain architecture and its applications in energy internet. In: 2017 IEEE international conference on energy internet (ICEI), IEEE, pp 176–181

[217] Wu S, Li Z, Yan L, et al (2024) Are we there yet? unraveling the state-of-the-art smart contract fuzzers. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, pp 1–13

[218] Xie J, Tang H, Huang T, et al (2019) A survey of blockchain technology applied to smart cities: Research issues and challenges. IEEE communications surveys & tutorials 21(3):2794–2830

[219] Xiong H, Chen M, Wu C, et al (2022) Research on progress of blockchain consensus algorithm: A review on recent progress of blockchain consensus algorithms. Future Internet 14(2):47

[220] Xu G, Liu L, Zhou Z (2022) Reentrancy vulnerability detection of smart contract based on bidirectional sequential neural network with hierarchical attention mechanism. In: 2022 International Conference on Blockchain Technology and Information Security (ICBCTIS), IEEE, pp 56–59

[221] Xue Y, Ye J, Zhang W, et al (2022) xfuzz: Machine learning guided cross-contract fuzzing. IEEE Transactions on Dependable and Secure Computing 21(2):515–529

[222] Yadav SP, Agrawal KK, Bhati BS, et al (2022) Blockchain-based cryptocurrency regulation: An overview. Computational Economics 59(4):1659–1675

[223] Yaga D, Mell P, Roby N, et al (2019) Blockchain technology overview. arXiv preprint arXiv:190611078

[224] Yang Z, Zhu W, Yu M (2023) Improvement and Optimization of Vulnerability Detection Methods for Ethernet Smart Contracts. IEEE Access 11:78207–78223. https://doi.org/10.1109/access.2023.3298672, URL https://ieeexplore.ieee.org/document/10192903/, publisher: Institute of Electrical and Electronics Engineers (IEEE)

[225] Yashavant CS, Kumar S, Karkare A (2022) Scrawld: A dataset of real world ethereum smart contracts labelled with vulnerabilities. arXiv preprint arXiv:220211409

[226] Yli-Huumo J, Ko D, Choi S, et al (2016) Where is current research on blockchain technology?—a systematic review. PloS one 11(10):e0163477

[227] Yu R, Zhang Y, Wang Y, et al (2023) TxMirror: When the Dynamic EVM Stack Meets Transactions for Smart Contract Vulnerability Detection. Symmetry 15(7):1345. https://doi.org/10.3390/sym15071345, URL https://www.mdpi.com/2073-8994/15/7/1345, publisher: MDPI AG

[228] Yuan Y, Xie T (2022) Svchecker: a deep learning-based system for smart contract vulnerability detection. In: International Conference on Computer Application and Information Security (ICCAIS 2021), SPIE, pp 226–231

[229] Yue W, Li L (2020) Sentiment analysis using word2vec-cnn-bilstm classification. In: 2020 seventh international conference on social networks analysis, management and security (SNAMS), IEEE, pp 1–5

[230] Zarrin J, Wen Phang H, Babu Saheer L, et al (2021) Blockchain for decentralization of internet: prospects, trends, and challenges. Cluster Computing 24(4):2841–2866

[231] Zhang F, Cecchetti E, Croman K, et al (2016) Town Crier: An Authenticated Data Feed for Smart Contracts. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, Vienna Austria, pp 270–282, https://doi.org/10.1145/2976749.2978326, URL https://dl.acm.org/doi/10.1145/2976749.2978326

[232] Zhang L, Chen W, Wang W, et al (2022) Cbgru: A detection method of smart contract vulnerability based on a hybrid model. Sensors 22(9):3577

[233] Zhang L, Wang J, Wang W, et al (2022) Smart contract vulnerability detection combined with multi-objective detection. Computer Networks 217:109289

[234] Zhang R, Xue R, Liu L (2019) Security and privacy on blockchain. ACM Computing Surveys (CSUR) 52(3):1–34

[235] Zhang Y, Liu D (2022) Toward Vulnerability Detection for Ethereum Smart Contracts Using Graph-Matching Network. Future Internet 14(11):326. https://doi.org/10.3390/fi14110326, URL https://www.mdpi.com/1999-5903/14/11/326, publisher: MDPI AG

[236] Zhao H, Su P, Qiu M (2022) A novel machine learning-based model for reentrant vulnerabilities detection. In: International Conference on Smart Computing and Communication, Springer, pp 290–299

[237] Zhen Z, Zhao X, Zhang J, et al (2024) Da-gnn: A smart contract vulnerability detection method based on dual attention graph neural network. Computer Networks 242:110238

[238] Zheng D, Jing C, Guo R, et al (2019) A traceable blockchain-based access authentication system with privacy preservation in vanets. IEEE Access 7:117716–117726

[239] Zheng P, Zheng Z, Wu J, et al (2020) Xblock-eth: Extracting and exploring blockchain data from ethereum. IEEE Open Journal of the Computer Society 1:95–106

[240] Zheng Z, Xie S, Dai HN, et al (2018) Blockchain challenges and opportunities: A survey. International journal of web and grid services 14(4):352–375

[241] Zheng Z, Chen W, Zhong Z, et al (2023) Securing the ethereum from smart ponzi schemes: Identification using static features. ACM Transactions on Software Engineering and Methodology 32(5):1–28

[242] Zheng Z, Zhang N, Su J, et al (2023) Turn the Rudder: A Beacon of Reentrancy Detection for Smart Contracts on Ethereum. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). IEEE, Melbourne, Australia, pp 295–306, https://doi.org/10.1109/icse48619.2023.00036, URL https://ieeexplore.ieee.org/document/10172623/

[243] Zhou K, Cheng J, Li H, et al (2021) Sc-vdm: A lightweight smart contract vulnerability detection model. In: International Conference on Data Mining and Big Data, Springer, pp 138–149

[244] Zhou Z, Tan L, Xu G (2018) Blockchain and edge computing based vehicle-to-grid energy trading in energy internet. In: 2018 2nd IEEE conference on energy internet and energy system integration (EI2), IEEE, pp 1–5

[245] Zikratov I, Kuzmin A, Akimenko V, et al (2017) Ensuring data integrity using blockchain technology. In: 2017 20th Conference of Open Innovations Association (FRUCT), IEEE, pp 534–539

[246] Zou L, Goh HL, Liew CJY, et al (2023) Ensemble Image Explainable AI (XAI) Algorithm for Severe Community-Acquired Pneumonia and COVID-19

Respiratory Infections. IEEE Transactions on Artificial Intelligence 4(2):242–254. https://doi.org/10.1109/tai.2022.3153754, URL https://ieeexplore.ieee.org/document/9721585/, publisher: Institute of Electrical and Electronics Engineers (IEEE)

[247] Zyskind G, Nathan O, et al (2015) Decentralizing privacy: Using blockchain to protect personal data. In: 2015 IEEE security and privacy workshops, IEEE, pp 180–184