Contents lists available at GrowingScience

International Journal of Data and Network Science

homepage: www.GrowingScience.com/ijds

Machine learning approaches for enhancing smart contracts security: A systematic literature review

Areej AlShorman[a*], Fatima Shannaq[b] and Mohammad Shehab[b]

[a]*Department of Computer Science, Faculty of Prince Al-Hussein Bin Abdallah II for IT, Al al-Bayt University, Mafraq, Jordan*
[b]*College of Computer and Informatics, Amman Arab University, Amman, Jordan*

C H R O N I C L E

Article history:
Received: January 3, 2024
Received in revised format: February 29, 2024
Accepted: April 12, 2024
Available online: April 12, 2024

Keywords:
Ethereum
Smart Contracts
Machine Learning
Vulnerability
Attack
Detection

A B S T R A C T

Smart contracts offer automation for various decentralized applications but suffer from vulnerabilities that cause financial losses. Detecting vulnerabilities is critical to safeguarding decentralized applications before deployment. Automatic detection is more efficient than manual auditing of large codebases. Machine learning (ML) has emerged as a suitable technique for vulnerability detection. However, a systematic literature review (SLR) of ML models is lacking, making it difficult to identify research gaps. No published systematic review exists for ML approaches to smart contract vulnerability detection. This research focuses on ML-driven detection mechanisms from various databases. 46 studies were selected and reviewed based on keywords. The contributions address three research questions: vulnerability identification, machine learning model approaches, and data sources. In addition to highlighting gaps that require further investigation, the drawbacks of machine learning are discussed. This study lays the groundwork for improving ML solutions by mapping technical challenges and future directions.

1. Introduction

Smart contracts are being widely adopted across sectors like finance, supply chain, healthcare, etc. (Jiang et al., 2023; Shorman et al., 2020; Litke et al., 2019; de la Rosa et al., 2016; Garg et al., 2019). However, their immutability poses security concerns if vulnerabilities exist in code. As networks cannot be changed post-deployment, reviewing and testing contracts pre-deployment is critical. Attacks like the $150 million DAO hack highlight the need for improved security methods and best practices to prevent such incidents (Ivanov et al., 2023). Generally, the development of savvy contract applications requires satisfactory arrangements to identify bugs that were recently sent and distributed on blockchain systems. Formal confirmation distinguishes vulnerabilities in shrewd contracts, sometime recently arranged by characterizing determinations, making models, and utilizing instruments to analyze models. Be that as it may, formal confirmation is challenging, time-consuming, and requires noteworthy manual exertion. In differentiation, machine learning strategies can consequently identify irregularities and vulnerabilities, lessening manual exertion. Also, machine learning models are more adaptable as the number of keen contracts develops, whereas formal confirmation can end up computationally costly. Generally, machine learning complements formal confirmation by empowering cost-effective and adaptable defenselessness locations (Eshghie et al., 2021). Machine learning for recognizing shrewd contract vulnerabilities has risen as an imperative investigation region as of late. Whereas numerous consider utilizing ML for defenselessness locations, a comprehensive survey of solidifying approaches is missing. This paper conducts an efficient overview of 46 papers from 2018-2023 to think about, analyze, and classify ML strategies for defenselessness discovery. The overview serves to develop an understanding of state-of-the-art location apparatuses and illuminate future inquiries about bearings. This survey includes the following contributions:

* Corresponding author.
E-mail address areej2017shorman@aabu.edu.jo (A. AlShorman)

ISSN 2561-8156 (Online) - ISSN 2561-8148 (Print)

doi: 10.5267/j.ijdns.2024.4.007

- We analyzed a comprehensive review of 46 relevant studies that focused on applying machine-learning techniques to detect smart contract security vulnerabilities.
- We conducted a deep analysis to understand smart contract vulnerabilities detected by ML techniques.
- We presented a classification of the machine learning methods used for detecting vulnerabilities in smart contracts according to their prevalence across primary studies.
- We studied the main datasets used in smart contract vulnerability detection based on the sources and types of these datasets.
- We discuss the main challenges that are facing the use of machine learning techniques for vulnerability detection in the future.

This work offers valuable insights on machine learning for smart contract vulnerability detection to researchers and industry professionals. By synthesizing current literature, we highlight useful models, benchmarks, and challenges to guide academics and practitioners toward impactful progress on this critical emerging need. The structured analysis aims to advance development processes, validate contract correctness, and reduce cyber risks.

The paper is organized as follows: Section 2 summarizes existing surveys and systematic reviews focusing on smart contract vulnerability detection using machine learning. Section 3 discusses the research methodology and criteria for choosing papers. Section 4 discusses the three research questions in the systematic literature review and presents the main results. Section 5 examines the main limitations of this survey. Finally, Section 6 concludes the paper and highlights possible future directions in this area.

2. Background and Related work

This section provides a background on smart contract vulnerabilities and utilizes machine learning methods to detect them. Then we compare our survey with related surveys and highlight their differences.

2.1. Background

2.1.1. Smart Contracts Vulnerabilities

Though smart contracts promise automation, reliability, and trust less, they are prone to vulnerabilities that can lead to exploited contracts, locked funds, and other issues (Atzei et al., 2017; Chen et al., 2020). Given the significant value flowing through blockchain platforms like Ethereum via smart contracts, securing them is paramount. In the next subsection, we present the most common smart contract vulnerabilities as follows:

- **Reentrancy vulnerabilities:** this vulnerability allows attackers to manipulate smart contract logic and steal assets by repetitively invoking functions, disrupting execution flow. These prevalent vulnerabilities, named after “reentrancy attacks” pose risks like enabling recursive withdrawal function calls that repeatedly transfer funds without updating balances (Qian et al., 2020; Jiang et al., 2023).
- **Gasless Send:** The gasless send powerlessness in Ethereum keen contracts happens when Ethers are exchanged (utilizing send work) to a contract and there's insufficient gas to execute the beneficiary contract's fallback work, causing an out-of-gas special case and unforeseen behavior (Qian et al., 2020). This powerlessness can empower denial-of-service assaults, information misfortune, or state changes. For example, a lottery contract utilizing send to pay victors may be abused by an assailant making a fallback work that intentioned causes blunders, causing the exchange to fall flat midway and solidifying the contract (Narayana & Sathiyamurthy, 2023). Distinguishing and moderating gasless send vulnerabilities through testing and secure coding is basic for guaranteeing shrewd contract security and unwavering quality on blockchain stages. Designers must scrutinize contracts to reveal and address this predominant defenselessness.
- **Call integrity**: This defenselessness permits assailants to misuse inappropriate dealing with outside calls to control contract behavior. This will empower unauthorized get to or control of reserves and information. Strong approval of outside call return values is key to invigorating contract keenness against such assaults (Lutz et al., 2021; Zheng et al., 2023).
- **Type Casts:** This helplessness happens when shrewd contracts incorrectly change over information sorts without approval, empowering assailants to supply inputs that trigger fizzled changes or unforeseen yields. For illustration, the unvalidated string-to-integer transformations can disturb contract rationale streams. To relieve abuses, input

information requires arranged approval, sometimes recently sorted changes, and yields ought to be checked for blunders. Recognizing this helplessness prompts designers to actualize thorough controls when dealing with blended information sorts (Raja et al., 2020).

- **Stack Size Limit:** This helplessness misuses Ethereum Virtual Machine's stack-based design. Aggressors can mishandle call stack profundity by recursive self-calls, inevitably surpassing the restrain to disturb execution and cause financial misfortunes (Eshghie et al., 2021; Lutz et al., 2021). Mitigations incorporate minimizing inner calls, capping recursion, and optimizing preparation but eventually require EVM convention changes to handle stack flood special cases. By understanding this center imperative, engineers can plan contracts protectively against exchange atomicity abuse.

- **Locked Ether:** This powerlessness happens when savvy contract stores are blocked off due to imperfections in rationale or startling conditions (Rameder et al., 2022). For illustration, contracts missing withdrawal highlights can trap ether sent by assailants, whereas imperfect withdrawal rationale can empower assaults activating special cases to bolt client stores (Ivanov et al., 2023). To avoid ether locking, withdrawal highlights, and blunder-taking care are basic when exchanging ether possession.

- **Unprotected Self-Destruction:** This powerlessness emerges when a savvy contract utilizes self-destructive work without appropriate control. This empowers assailants to abuse it to divert reserves or disturb contract behavior, posturing critical dangers (Hwang et al., 2022). For illustration, a contract with an admin-only murder switch needs checks, permitting any client to end it. By doing so, assailants for all time deny benefit accessibility to authentic clients. To avoid this, designers ought to execute get-to controls limiting self-destruct conjuring to authorized substances as it were (Zheng et al., 2023). Shrewd contracts require intensive testing and reviewing to distinguish and relieve vulnerabilities related to unprotected utilization of self-destruct.

- **Timestamp Dependency:** This powerlessness emerges when diggers control savvy contract square timestamps to alter time-dependent rationale like due dates and clocks (Zheng et al., 2023). For illustration, aggressors may abuse pool contracts selecting arbitrary champs based on piece timestamps, guaranteeing they win over and over. Additionally, offered closure timestamps can be modified to negate authentic offers rashly (Hwang et al., 2022). To moderate assaults, designers ought to assess time conditions, join security measures like state checks, and input approval. Clients ought to moreover work out caution when locked in contracts defenseless to timestamp control.

- **Access control:** This defenselessness permits unauthorized contracts to get to, empowering noxious exercises like unauthorized support exchanges or information spills (Sun et al., 2023). In this case, the imperfect rationale may let anybody conjure limited capacities, controlling contract states and stores (Xu et al., 2021). Appropriate get-to-control instruments inside contract code are basic to avoid this, like role-based get-to-control and fitting work permeability settings. Executing solid get-to controls is pivotal for guaranteeing keen contract security and keenness.

- **Transaction-ordering dependencies**: This vulnerability poses security risks by enabling attackers to exploit the sequence of blockchain transactions. Vulnerabilities like front-running let attackers unfairly manipulate orders for profit (Narayana & Sathiyamurthy, 2023). For example, decentralized exchanges can be targeted by initiating token sales without balances, then quickly purchasing those tokens before sell transactions confirm (Lutz et al., 2021). To mitigate this, transaction sequence assumptions should be discarded, critical operations consolidated, and blockchain states cross-referenced across transactions to ensure execution order consistency.
- **Bad Random Number Generation:** Inadequate random number generation in smart contracts stems from insecure methods vulnerable to prediction or manipulation by attackers leveraging blockchain transparency (Zhang et al., 2022). For example, lotteries using previous block hashes for winner selection can be exploited by miners ensuring favorable hashes through specific transactions. Pseudo-randomness from block data has proven insecure. Solutions like timed commitment protocols better secure randomness by having participants commit secrets and deposits, later disclosing secrets or forfeiting deposits to compute final random numbers (Ibba et al., 2021). Secure blockchain randomness requires replacing vulnerable methods with robust cryptographic protocols.
- **Unpredictable state:** This vulnerability leads to unforeseen behaviors or outcomes due to race conditions, external influences, or interactions with other contracts (Liu et al., 2021b). These blemishes have caused major security breaches and budgetary misfortunes, just like the Equality Wallet occurrence, which cost $300 million worth of Ether (Qian et al., 2020). To moderate eccentric contract states, engineers must actualize shields and validations to guarantee secure behavior, nearby comprehensive testing and reviews some time recently sending. Distinguishing and tending to non-deterministic code viewpoints is basic to avoid troublesome occasions stemming from untrustworthy contract states.

2.1.2. Machine Learning Approaches for Detecting Vulnerabilities in Smart Contracts

Machine learning strategies have revolutionized shrewd contract powerlessness location (Cai et al., 2023; Yang et al., 2022; Liu et al., 2021b; Melody et al., 2019; Xing et al., 2020), empowering mechanized location, more profound investigation, and the recognizable proof of obscure vulnerabilities. Machine learning is broadly utilized for defenselessness location by analyzing the source code of savvy contracts (Cai et al., 2023; Momeni et al., 2019; Kim et al., 2019; Zhou et al., 2022; Yang et al., 2022; Liu et al., 2021b, 2022; Yashavant et al., 2022; Tune et al., 2019; Xu et al., 2021; Jie et al., 2023), bytecode (Wang et al., 2020; Xing et al., 2020), or arrange environment vulnerabilities (Varun et al., 2022; Eshghie et al., 2021; Eduardo A. Sousa et al., 2021; Aziz et al., 2022; Mandloi and Bansal, 2022; Tooth et al., 2020; Pahuja and Kamal; Xiong et al., 2023). ML models can be prepared to utilize known powerlessness information to identify vulnerabilities in savvy contracts. The preparing handle includes extricating highlights from shrewd contract code, bytecode, or the environment arrangement. These highlights are utilized to prepare a classification demonstration. The show predicts vulnerabilities in keen contracts based on learned designs from labelled information. The result of the show is assessed utilizing measurements such as review, exactness, and exactness to survey its capacity to precisely classify powerless and non-vulnerable keen contracts.

Shrewd contract vulnerabilities can be recognized utilizing machine learning models, which offer noteworthy focal points over classical techniques. The other section will examine these focal points as follows:

Automation: ML models can computerize the location handle of vulnerabilities within the contracts, decreasing the requirement for manual assessment and examination (Lutz et al., 2021). This mechanization can essentially speed up the discovery preparation, particularly when managing an expansive number of keen contracts.
Adaptability: Machine-learning models can adjust and learn to identify unused sorts of vulnerabilities (Rameder et al., 2022). This capability empowers keen contracts to quickly recognize unused security vulnerabilities that are sometimes recently sent on blockchain systems.
Reduce Human Bias: A machine-learning demonstration can aid decrease human inclination to helpless discovery by giving objective and data-driven bits of knowledge. As a result, the forecast handle gets to be more exact and solid (Jiang et al., 2023).
Complex Pattern Recognition: ML models can recognize complex designs and connections inside savvy contract code which will not be effectively recognizable through conventional techniques (Sürücü et al., 2022). This capacity to recognize perplexing designs can lead to the location of unobtrusive vulnerabilities.
Precision: Machine learning models can be prepared to distinguish vulnerabilities with high accuracy, diminishing untrue positives and untrue negatives (Lutz et al., 2021). This exactness is fundamental for precisely recognizing security issues in savvy contracts.

2.2. Related work

This segment offers related overviews on the considered theme and highlights their highlights, as appeared in Table 1. The contributions in (Jiang et al., 2023) present a survey of machine learning-based smart contract vulnerability detection tools, outlining challenges and future trends. (Virani & Kyada, 2022) conducts a systematic literature review on smart contract security, identifying weaknesses and detection techniques. (Wang et al., 2021b) presented a comprehensive literature review categorizing security tools and techniques into six groups. (Chen et al., 2021) conducted an empirical study analyzing papers to identify twelve maintenance issues and classify current methods. The goal is to explore vulnerabilities, detection approaches, maintenance concerns, and mitigation measures to enhance smart contract security across these works. Ivanov et al. (2023) conducted a survey presenting a taxonomy to classify 133 smart contract threat mitigation solutions into eight core methods and created standardized workflows. It performs an evolutionary analysis identifying trends and future perspectives while highlighting weaknesses. Alharby and Van Moorsel (2017) performed a systematic mapping study summarizing the state-of-the-art in smart contract security and correctness verification. It introduces taxonomy, addresses open challenges, and proposes solutions. Sürücü et al. (2022) presented the first survey focused on ML models for detecting smart contract vulnerabilities, showing higher efficiency than traditional methods. Tikhomirov (2017) contributed by selecting 53 papers showcasing verification methods, introduced a categorization taxonomy, and emphasized formal methods' importance. Mololoth et al. (2023) explored integrating blockchain and ML for future smart grids. It highlighted security and transparency benefits, grid management and prediction optimization, and using smart contracts to automate energy trading. Research gaps and future directions were also discussed. In Huang et al. (2019), a literature review on smart contract security is conducted from a software lifecycle perspective, highlighting vulnerabilities and best practices for secure development. Kushwaha et al. (2022) analyzes Ethereum smart contract vulnerabilities, detection tools, real-world attacks, and precautions through a systematic approach. Imperius and Alahmar (2022) present a systematic overview of techniques for testing smart contract reliability and security on blockchains. Chithanuru and Ramaiah (2023) studied utilizing AI for inconsistency discovery in blockchain, emphasizing security angles and utilization cases. Hasanova et al. (2019) give an in-depth examination of blockchain vulnerabilities and propose relief methodologies like upgrades to cryptography, agreement calculations, and secure coding hones. The common topic is distinguishing security issues in savvy contracts and blockchain frameworks whereas investigating methods to anticipate, distinguish, and relieve vulnerabilities over the computer program lifecycle. The orderly survey by Kirli et al. (2022) gives an outline of keen contract applications and benefits in vitality frameworks, highlighting the potential

for change in nearby bits of knowledge for industry partners. Huang et al. (2019) audit savvy contract security over the program lifecycle, pointing to distinguish and relieve vulnerabilities utilizing procedures like audits and ML. Kushwaha et al. (2022) analyze Ethereum contract vulnerabilities, devices, assaults, and safety measures methodically. Imperius and Alahmar (2022) present an outline of unwavering quality and security testing strategies for contracts on blockchains. Chithanuru and Ramaiah (2023) overview AI strategies for peculiarity discovery in blockchain covering angles like contracts, agreements, and frameworks. The center ranges span analyzing contract vulnerabilities, looking into location approaches, analyzing unwavering quality testing strategies, utilizing AI for security, and tackling preferences of savvy contracts in vitality frameworks.

Table 1
Recent surveys have examined machine learning models for enhancing smart contract security

Ref.	Title	Publication venue	Year	Time line	Syst. review	Smart contracts	ML focused	Datasets description
(Atzei et al., 2017)	A survey of attacks on ethereum smart contracts (sok)	Springer	2017	No	Yes	No	No	No
(Alharby and Van Moorsel, 2017)	Blockchain-based smart contracts: A systematic mapping study	arXiv	2017	No	Yes	No	No	No
(Tikhomirov, 2017)	A survey on Security verification of blockchain smart contracts	IEEE	2019	No	Yes	No	No	No
(Huang et al., 2019)	Smart contract security: A software lifecycle perspective	IEEE	2019	No	Yes	No	No	No
(Hasanova et al., 2019)	A survey on blockchain cybersecurity vulnerabilities and possible countermeasures	Wiley	2019	No	Yes	No	No	No
(Chen et al., 2020)	A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defenses	ACM	2020	No	Yes	No	No	No
(Wang et al., 2021b)	Security enhancement technologies for smart contracts in the blockchain: A survey	Wiley	2021	Yes	Yes	No	No	No
(Chen et al., 2021)	Maintenance-related concerns for postdeployed Ethereum smart contract development: issues, techniques, and future challenges	Springer	2021	Yes	Yes	No	Yes	Yes
(Timucin and BIRO˙GUL, 2021)˘	A survey: Making “Smart Contracts” really smart	Wiley	2021	No	Yes	Yes	No	No
(Virani and Kyada, 2022)	A Systematic Literature Review on Smart Contracts Security	arXiv	2022	Yes	Yes	No	No	No
(Su¨ru¨cu¨ et al., 2022)	A survey on ethereum smart contract vulnerability detection using machine learning	SPIE	2022	No	Yes	Yes	Yes	Yes
(Kirli et al., 2022)	Smart contracts in energy systems: A systematic review of fundamental approaches and implementations	Elsevier	2022	Yes	Yes	Yes	No	No
(Kushwaha et al., 2022)	Systematic review of security vulnerabilities in ethereum blockchain smart contract	IEEE	2022	Yes	Yes	No	No	No
Imperius and Alahmar (2022)	Systematic Mapping of Testing Smart Contracts for Blockchain Applications	IEEE	2022	Yes	Yes	No	No	No
Rameder et al. (2022)	Review of automated vulnerability analysis of smart contracts on Ethereum	Frontiers	2022	Yes	Yes	No	No	No
(Ivanov et al., 2023)	Security Threat Mitigation For Smart Contracts: A Comprehensive Survey	ACM	2023	No	Yes	Yes	No	No
(Jiang et al., 2023)	Enhancing Smart-Contract Security through Machine Learning: A Survey of Approaches and Techniques	Electronics	2023	No	Yes	Yes	Yes	Yes
(Mololoth et al., 2023)	Blockchain and machine° learning for future smart grids: A review	MDPI	2023	No	Yes	Yes	No	No
(Chithanuru and Ramaiah, 2023)	An anomaly detection on blockchain infrastructure using artificial intelligence techniques: Challenges and future directions–A review	Wiley	2023	No	Yes	Yes	Yes	Yes
(Piantadosi et al., 2023)	Detecting functional and securityrelated issues in smart contracts: A systematic literature review	Wiley	2023	Yes	Yes	No	Yes	Yes
Our SLR	Machine Learning Approaches for Enhancing Smart Contracts Security: A Systematic Literature Review	-	-	Yes	Yes	Yes	Yes	Yes

Hasanova et al. (2019) presents an in-depth examination of blockchain vulnerabilities, proposing relief techniques including cryptography, agreement calculations, and secure coding best hones. Timucin and BIROË GUL (2021) studied approaches to improve savvy contract insights utilizing procedures like ML and formal confirmation. Piantadosi et al. (2023) surveys inquire about consequently distinguishing shrewd contract bugs and vulnerabilities. Atzei et al. (2017) presented the primary scientific classification of programming pitfalls causing Ethereum contract vulnerabilities, analyzing related dangers. Rameder et al. (2022) systematically reviews the state-of-the-art automated smart contract vulnerability analysis on Ethereum, examining classifications, detection methods, tools, and benchmarks. The key focus areas are analyzing vulnerabilities in blockchain systems and smart contracts, reviewing detection approaches, discussing enhancement techniques to improve contract robustness, and suggesting countermeasures to security issues.

Alharby and van Moorsel (2017) conducted a systematic mapping study of 24 papers on smart contracts, highlighting current research topics and open challenges. Tikhomirov (2017) summarized Ethereum's technical overview, problems, and suggested solutions. Atzei et al. (2017) developed a taxonomy of Ethereum contract vulnerabilities and reviewed related attacks. Wöhrer and Zdun (2018) presented a literature review defining Solidity-level smart contract patterns in terms of problems and solutions. Grishchenko et al. (2018) surveyed verification and security-focused design approaches for Ethereum contracts, discussing advancements like the EtherTrust static analyzer. Mense and Flatscher (2018) presented a new taxonomy of vulnerabilities based on ecosystem severity and compared detection tools to help developers avoid critical issues. The focus is on summarizing Ethereum smart contract security issues, vulnerabilities, verification methods, design patterns, and detection tools to enhance robustness.

Macrinici et al. (2018) conducted a systematic mapping study covering smart contract knowledge, vulnerabilities in security/privacy/scalability/programming, and prospective solutions. Miller et al. (2018) surveyed the smart contract ecosystem and formal analysis methods, challenges, and opportunities. Harz and Knottenbelt (2018) reviewed languages and tools based on security properties for handling vulnerabilities. Kirillov et al. (2019) reviewed Solidity contract vulnerabilities in Ethereum and proposed software analysis tools alongside manual auditing for security. Rouhani and Deters (2019) systematically surveyed smart contracts in terms of security methods/tools, performance, and applications. This SLR presents a structured analysis of ML techniques and models for smart contract vulnerability detection across dimensions like vulnerabilities covered, models adopted, and dataset sources/types while proposing future work to advance ML solutions.

3. Research methodology

SLRs study, evaluate, and interpret existing research on vulnerability detection in smart contracts using ML techniques through a reasonable and impartial methodology. As no comprehensive systematic studies exist on ML for detecting contract vulnerabilities, this work follows Kitchenham's methodology (Keele et al., 2007; Kitchenham, 2004) to fill the gap. Additionally, the baseline method from Al-Shamayleh et al. (2018) is followed to identify the state-of-the-art research for answering the research questions. Conducting an SLR enables trustworthy, rigorous, and unbiased assessment to advance understanding of ML techniques applied to smart contract security. The current SLR involves several steps, including planning, conducting, and reporting as shown in Fig. 1.

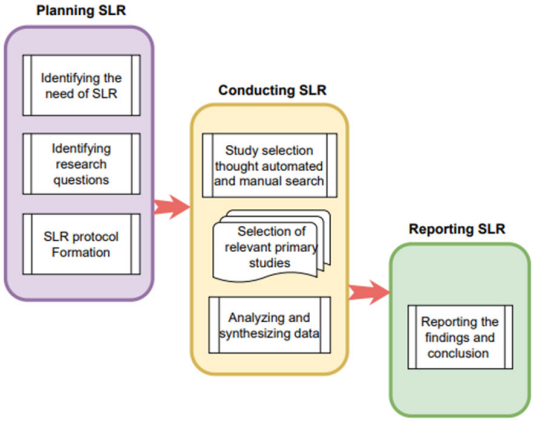

Fig. 1. Flowchart of SLR process (Al-Shamayleh et al., 2018)

3.1. Research design

In this step, the current research requirements are explained by identifying the research questions and their corresponding keywords.

3.1.1. Research Questions

According to our literature review, the state-of-the-art lacks systematic reviews utilizing ML to detect smart contract vulnerabilities in terms of technique characteristics and dataset sources. Conducting such a survey would benefit practitioners and researchers by advancing understanding of detection tools and informing future work. To address this gap, our study conducts a comprehensive survey, reviewing, analyzing, describing, and classifying 46 articles from 2018-2023 to delve into vulnerability detection across perspectives. We examined papers through research questions focused on critical dimensions of ML techniques and datasets for security enhancement.

- **RQ1:** What are the most frequent smart contract vulnerabilities faced by machine learning approaches?
- **RQ2:** What are the machine learning models used for smart contracts vulnerability detection?
- **RQ3:** What are the sources and types of datasets used in smart contract security using machine learning?

3.1.2. Research process

By utilizing this SLR process, primary study papers can be located by searching five electronic databases, as shown in Table 2, which contain full-text articles and papers about using machine learning techniques to detect vulnerabilities in smart contracts. A secondary scanning and analysis step, in addition to the first search using these databases, ensures the accuracy and comprehensiveness of the studies as they relate to the current research questions. Search results are primarily related to detecting vulnerabilities in Ethereum smart contracts using machine learning techniques.

Table 2
Online databases

Identifier	Database	URL
ED1	IEEE Xplore	https://ieeexplore.ieee.org/
ED2	ACM	https://dl.acm.org/
ED3	Springer Link	https://link.springer.com/
ED4	Science Direct	https://www.sciencedirect.com/
ED5	Wiley	https://onlinelibrary.wiley.com/

3.1.3. Keywords

The use of suitable keywords can help save valuable time by filtering studies relevant to the three main research questions. These keywords were derived from the titles, abstracts, or keywords of relevant papers. We employed interchangeable keywords to locate more research papers related to Ethereum smart contracts. The following keywords are used.

1. **Ethereum**
2. **Blockchain**
3. "**Smart Contracts**"
4. "**Machine Learning**"
5. "**Artificial Intelligence**"
6. **Venerability** OR **Detection** OR **Attacks** OR **Security**

These keywords were combined using the conjunction AND and disjunction OR logic operators. The resulting keyword for the automated search is given as follows:

((((**Ethereum** *OR* **Blockchain**) *AND* (" **Smart contracts**")) *AND* (("**Machine Learning**") *OR* (" **Artificial Intelligence**"))) *AND* (**Venerability** *OR* **Detection** *OR* **Attacks**
OR **Security**))

3.2. Review conduction

In this section, we illustrate how the SLR process is conducted. The SLR search process involves several phases based on the rules and frameworks used to create this review.

3.2.1. Selection of relevant studies

After obtaining the initial research studies, their relevance was assessed. To verify the relevance of the initially selected studies, a second round of research was conducted by reviewing them. To ensure that our inclusion and exclusion criteria

were consistent, we randomly re-evaluated the selected studies after the initial screening. The SLR study selection procedure is shown in Fig. 2.

The selection of relevant research studies was done in seven steps, as follows:

1. Find relevant research works by using appropriate keywords in databases.
2. Discard studies not relevant to the search criteria.
3. Remove studies based on their titles or abstracts that are irrelevant.
4. Read the context of the selected studies in full to assess them.
5. Assess other researchers' bibliographies.
6. Analyze the results for the second check in an organized manner.
7. Obtain initial studies.

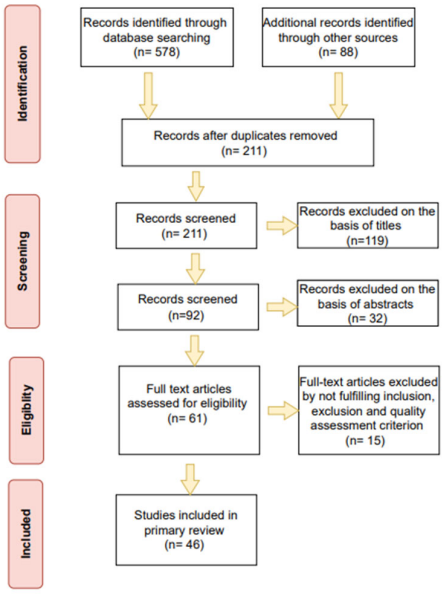

Fig. 2. Study selection procedure for the current SLR

3.2.2. Inclusion and exclusion criteria

Using the inclusion and exclusion criteria, we identified all relevant studies. The scope of this SLR encompasses research studies on machine learning and smart contract security. Furthermore, the study considered journal articles and full papers in English from peer-reviewed original studies published from January 2018 to July 2023 to ensure the inclusion of recent and up-to-date studies. In contrast, studies that do not directly answer the research questions or do not cover the same area are excluded from this review. The inclusion and exclusion criteria are summarized in Table 3. Moreover, the percentage of initial and selected studies for each online database source is provided in Fig. 3.

Table 3
Inclusion and exclusion criteria involved in the search process

Inclusion and exclusion criteria	
Inclusion Criteria	
IC1	Studies that are peer reviewed original studies.
IC2	Peer-reviewed studies that address the research questions
IC3	Studies that focus on machine learning and smart contracts vulnerability detection
IC4	Studies are published after 2018.
Exclusion criteria	
EC1	Studies are not in English.
EC2	Studies that are unrelated to the research questions.
EC3	Duplicated studies (by title or content).
EC4	Studies providing unclear results or findings.
EC5	Studies published in the conferences.
EC6	Studies addressing non-machine learning methods such as static analysis, dynamic analysis, etc.
EC7	Scurvies papers.

3.2.3. Quality assessment criteria

SLR quality was evaluated during the eligibility phase. A combination of inclusion/exclusion criteria and quality assessment criteria shown in Table 4 were applied to the studies screened in the screening phase. In total, 61 studies were submitted to six researchers. Three researchers reviewed each study to exclude irrelevant studies. Criteria for inclusion and exclusion were rated from 1 to 5. Based on rigorous review and screening, 46 studies were selected after applying inclusion/exclusion and quality assessment criteria to the full text of 61 studies.

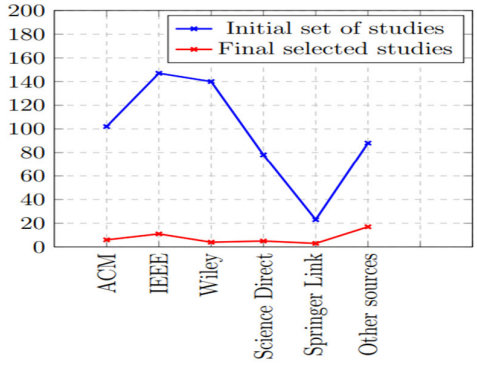

Fig. 3. Proportion of selected studies

Table 4
Quality assessment criteria for the SLR

Quality Assessment Criteria	
QC1	The proposed vulnerability detection approaches
QC2	Context and environment are clearly defined.
QC3	The research design supports the study goals and
QC4	Objectives, goals, and results are accurately deline-
QC5	Contributions and limitations are clearly defined.
QC6	The evaluation metrics are used to

3.2.4. Data extraction

In the data extraction and synthesis process, the main objective is to gather the information required from the studies that were selected after the second phase by a specialized team. A structured Microsoft Excel data extraction form as shown in Table 5 was used for the relevant information. Additionally, four researchers analyzed the full text of each study.

Table 5
Data extraction form

No.	Extracted data	Description
1.	Identity of study	Unique identity for the study.
2.	Bibliographic references	Authors, year of publication, title, citations, source of Publication, and publication venue.
3.	Empirical research method	Experiment, case study, and survey.
4.	Contribution	Method, model, framework, and tools.
5.	Focus of study	Main topic areas, keywords, motivation, and purpose of the study.
6.	Vulnerabilities of smart contracts	Studying the vulnerabilities that affect the security of smart contracts.
7.	The existing machine learning approaches	Description the machine learning methods, models, and tools that are applied to smart contract security.
8.	The datasets used	Description of the datasets used in training the ML models, including their sources.

3.3. Demographic data and overview

Analyzing the demographic features of the studies included in the evaluation is crucial to gauging the quality of research and the total result in this research area. Additionally, this analysis will provide future research by highlighting the most impactful studies in this domain.

4. Results

Based on the research questions outlined in the preceding section, we present the findings of the systematic literature review in this section.

4.1. RQ1: What are the most frequent smart contract vulnerabilities faced by machine learning approaches?

Various classifications exist in the literature for categorizing vulnerabilities in smart contracts. These classifications encompass factors such as the level at which they occur, potential attack vectors, and the severity of security risks they pose. For

example, Momeni et al. (2019) address 16 different types of vulnerabilities. Table 6 offers a comprehensive classification that considers both the introduction level and the associated security severity of vulnerabilities. Please note that some cells in Table 6 are empty and are denoted as "-".

Table 6
Synthesis of Ethereum smart contract vulnerabilities covered in primary studies

Vulnerability level	Type of vulnerability	security level	#Studies	References
High level language	Re-entrancy	High	22	[Ashizawa et al. (2021), Sun and Gu (2021), Eshghie et al. (2021), Lutz et al. (2021), Mandloi and Bansal (2022), Zheng et al. (2023), Qian et al. (2022), Zhang and Liu (2022), Momeni et al. (2019), Qian et al. (2020), Zhou et al. (2022), Hwang et al. (2022), Yang et al. (2022), Liu et al. (2021b), Zhang et al. (2022), Yashavant et al. (2022), Sun et al. (2023), Song et al. (2019), Xing et al. (2020), Xu et al. (2021), Jie et al. (2023), Narayana and Sathiyamurthy (2023)]
	Gasless send	Medium High	12	[Ashizawa et al. (2021), Eduardo A. Sousa et al. (2021), Lutz et al. (2021), Gupta et al. (2022), Zheng et al. (2023), Liu et al. (2021a), Shah et al. (2023), Essaid et al. (2019), Fang et al. (2020), Yashavant et al. (2022), Xu et al. (2021), Narayana and Sathiyamurthy (2023)]
	Call integrity	Medium High	9	[Lutz et al. (2021), Zheng et al. (2023), Momeni et al. (2019), Zhou et al. (2022), Hwang et al. (2022), Yang et al. (2022), Zhang et al. (2022), Sun et al. (2023), Xu et al. (2021)]
	Type casts	Medium	2	[Ashizawa et al. (2021), Yashavant et al. (2022)]
	Leakage of confidentiality	High	1	[Momeni et al. (2019)]
-	-	SUM	45(28)	-
EVM bytecode	Stack size limit	Medium-High	14	[Ashizawa et al. (2021),Sun and Gu (2021), Lutz et al. (2021), Zheng et al. (2023), Qian et al. (2022), Momeni et al. (2019), Zhou et al. (2022), Zhang et al. (2022), Yashavant et al. (2022), Sun et al. (2023), Song et al. (2019), Xing et al. (2020), Xu et al. (2021), Jie et al. (2023)]
	Locked Ether	High	5	[Zheng et al. (2023), Tann et al. (2018), Momeni et al. (2019), Xing et al. (2020), Xu et al. (2021)]
	Unprotected self-destruction	High	4	[Zheng et al. (2023), Lutz et al. (2021), Gogineni et al. (2020), Momeni et al. (2019)]
-	-	SUM	23(17)	- -
Blockchain Environment	Timestamp dependency	Medium-High	12	[Ashizawa et al. (2021), Sun and Gu (2021), Zheng et al. (2023), Qian et al. (2022), Zhang and Liu (2022), Hwang et al. (2022), Liu et al. (2021b), Zhang et al. (2022), Yashavant et al. (2022), Sun et al. (2023), Song et al. (2019), Jie et al. (2023)]
	Access control	High	9	[Zheng et al. (2023), Gaur et al. (2022), Zhang and Liu (2022), Momeni et al. (2019), Hwang et al. (2022), Sun et al. (2023), Xu et al. (2021), Jie et al. (2023), Narayana and Sathiyamurthy (2023)]
	Transaction-ordering dependency (TOD)	High	6	[Varun et al. (2022), Lutz et al. (2021), Qian et al. (2022), Zhang et al. (2022), Sun et al. (2023), Song et al. (2019)]
	Bad random number generation	Medium - High	3	[Yang et al. (2022), Xu et al. (2021), Jie et al. (2023)]
	Unpredictable state	High	1	[Zhou et al. (2022)]
-	-	SUM	31(21)	- -

4.1.1. Source Code Vulnerabilities

The data presented in Table 6 demonstrates that among vulnerability classifications, source code vulnerabilities receive the most research attention. These refer to errors introduced in smart contracts authored by developers, wherein the programming code is written using programming languages. Hence, they are susceptible to the entire range of bugs and weaknesses known to afflict computer programs in general. When developers fail to identify and rectify vulnerabilities at the source code level, the subsequent execution of smart contracts may exhibit unanticipated behaviors. Attackers can then leverage these vulnerabilities to manipulate the execution logic of contracts for malicious objectives.

It is observable that Re-entrancy is the most frequently addressed vulnerability type by 23 primary studies (Ashizawa et al., 2021; Sun and Gu, 2021; Eshghie et al., 2021; Lutz et al., 2021; Mandloi and Bansal, 2022; Zheng et al., 2023; Qian et al., 2022; Zhang and Liu, 2022; Momeni et al., 2019; Kim et al., 2019; Zhou et al., 2022; Hwang et al., 2022; Yang et al., 2022; Liu et al., 2021b; Zhang et al., 2022; Yashavant et al., 2022; Sun et al., 2023; Song et al., 2019; Xing et al., 2020; Xu et al., 2021; Jie et al., 2023; Narayana and Sathiyamurthy, 2023). This powerlessness happens when the contract permits recursive calls back into itself sometime recently overhauling the state, empowering stores to be depleted more than once. Aggressors

can abuse this helplessness to create a recursive call back into the powerless work, rehashing operations sometimes recently the casualty contract's state has been overhauled. In reality, two components, to be specific the criticality score relegated to it and the recurrence of essential considerations tending to it, decide the significance of this helplessness. Coming in as the moment most commonly tended to powerlessness sort inside the source code blunders category, the issue of gasless send is investigated in profundity by 12 diverse essential ponders (Ashizawa et al., 2021; Eduardo A. Sousa et al., 2021; Lutz et al., 2021; Gupta et al., 2022; Zheng et al., 2023; Liu et al., 2021a; Shah et al., 2023; Essaid et al., 2019; Tooth et al., 2020; Yashavant et al., 2022; Xu et al., 2021; Narayana and Sathiyamurthy, 2023). This specific powerlessness, which is evaluated at a medium level of seriousness, can lead to fragmented overhauls of the contract's state, causing possibly hopeless resource or information debasement. Besides, assailants abuse this powerlessness to carry out denial-of-service assaults. Particularly, gasless send vulnerabilities happen when savvy contracts utilize capacities like send() or exchange() to exchange Ether without setting a gas constraint. Bypassing all available gas to the recipient, the contract retains no gas to handle exceptions, any errors, or unfinished computations after the transfer.

Call integrity has emerged as another prevalent vulnerability classification, with 9 primary studies (Zheng et al., 2023; Lutz et al., 2021; Zheng et al., 2023; Momeni et al., 2019; Zhou et al., 2022; Hwang et al., 2022; Yang et al., 2022; Zhang et al., 2022; Sun et al., 2023; Xu et al., 2021) dedicated to examining it. This vulnerability occurs when smart contracts call external function invocations towards untrusted contracts. Attackers can possibly abuse the inappropriate taking care of call return values to control execution rationale, bring about resource steady loss, repudiate legally binding rules, dissolve beliefs, and trigger high-priority keenness concerns. Call judgment vulnerabilities are regularly evaluated at medium to basic seriousness. As an outline, Decentralized Fund (DeFi) may join oracle-based contracts for securing resource estimating inputs to decide intrigued and collateralization extents on advances. Aggressors can develop false prophets to intentionally outfit distorted proportions by capitalizing on unconfirmed information ingestion. By misleadingly blowing up collateral resource costs, noxious substances seem to borrow entireties surpassing veritable security collateral, subsequently disturbing center rationale and controls. Subsequently, precise machine-learning models are essential to identify this sort of helplessness.

The slightest visit powerlessness sorts in this classification are sort casts (Ashizawa et al., 2021; Yashavant et al., 2022) and spillage of privacy (Momeni et al., 2019). The sort-cast powerlessness happens when there's an unseemly utilization of sort-casting capacities in a shrewd contract. Sort casting changes information from one sort to another, like changing over a string to a number. On the off chance that it is done improperly without approval, this will lead to unforeseen behaviors. Assailants misuse this by nourishing inputs that cause the sort of change to come up short or give startling results. For illustration, consider a keen contract that takes a string parameter, changes it over to a number utilizing the sort cast work, and after that employs it for a few calculations. Spillage of privacy happens when touchy information is disgracefully put away or taken care of inside a savvy contract's code. Since blockchains are transparent by design, data stored directly in contracts is visible to all. Attackers exploit this by analyzing contractual data flow to uncover secret values.

4.1.2. Bytecode Vulnerabilities

Numerous exploitation scenarios lead to unauthorized Ether drainage. One such scenario involves a malformed bytecode that can be crafted to make unauthorized Ether transfers or to skip authorization checks in contracts. A second scenario targets token transfer functionality, which is expected to check if the sender’s balance exceeds the transfer amount before execution. An attacker deploys a malformed bytecode that jumps straight to the token transfer logic without balance checks. As can be seen, the stack size limit is the most frequent type of vulnerability, accounting for 14 primary studies (Ashizawa et al., 2021; Sun and Gu, 2021; Lutz et al., 2021; Zheng et al., 2023; Qian et al., 2022; Momeni et al., 2019; Zhou et al., 2022; Zhang et al., 2022; Yashavant et al., 2022; Sun et al., 2023; Song et al., 2019; Xing et al., 2020; Xu et al., 2021; Jie et al., 2023). The stack size limit refers to a situation when smart contract functions make extensive internal function calls that occupy stack space beyond specified limits. In the Ethereum Virtual Machine (EVM), stack space is limited to 1024 calls. Attackers abuse this by constraining contracts into profound recursion. The stack estimates constraints helplessness in shrewd contracts and is ordinarily considered medium-critical in seriousness. As a result, this powerlessness makes dissent of benefit clear, can bolt client reserves, is dubious to fix, needs protocol-level changes, and ruins exchanges, making its security effect noteworthy. Bolted ether is the moment most noticeable defenselessness sought by 5 essential considerations (Zheng et al., 2023; Tann et al., 2018; Momeni et al., 2019; Xing et al., 2020; Xu et al., 2021). This classification of security defenselessness relates to getting ether caught and rendered unrecoverable. Aggressors abuse this by purposely activating conditions that exchange ether to keen contracts with no withdrawal capacities. Unprotected self-destruction has risen as another high-risk helplessness classification, with 4 essential things (Zheng et al., 2023; Lutz et al., 2021; Gogineni et al., 2020; Momeni et al., 2019) devoted to analyzing it. This defenselessness happens when shrewd contracts have capacities that can trigger the devastation or suicide of the contract without getting control. The self-destruct operation in Robustness ends code execution and wipes out the contract's information and usefulness all time. Assailants misuse this helplessness to mount changeless denial-of-service assaults. For illustration, consider a contract with an admin-only slaughter switch that permits the admin to end it beneath certain conditions.

4.1.3. Blockchain Vulnerabilities

Essential things about addressing the critical sorts of helplessness inside the blockchain environment. Blockchain environment vulnerabilities emerge when savvy contracts make imperfect presumptions around the fundamental blockchain biological system they exist inside. Aggressors misuse the decentralization and straightforwardness of open blockchains to break these presumptions and disturb contract rationale. For example, sequential and censorship resistance properties may erroneously lead designers to assume strict exchange requesting and idealize state confinement. Aggressors perform exchange reordering or replaying assaults to damage this center presumption. It is discernible that timestamp reliance (Ashizawa et al., 2021; Sun & Gu, 2021; Zheng et al., 2023; Qian et al., 2022; Zhang & Liu, 2022; Hwang et al., 2022; Liu et al., 2021b; Zhang et al., 2022; Yashavant et al., 2022; Sun et al., 2023; Tune et al., 2019; Jie et al., 2023) is the foremost visit powerlessness sort in this classification secured by the 12 essential ponders. This vulnerability occurs when smart contracts use block timestamps to trigger actions, make random numbers, or take blockchain-dependent decisions. As miners can manipulate block timestamps to some extent, attackers exploit this to influence contract execution.

Access control has surfaced as a prevalent, high-severity vulnerability, with coverage across 9 primary studies (Zheng et al., 2023; Gaur et al., 2022; Zhang & Liu, 2022; Momeni et al., 2019; Hwang et al., 2022; Sun et al., 2023; Xu et al., 2021; Jie et al., 2023; Narayana and Sathiyamurthy, 2023). This powerlessness happens when savvy contracts fall flat to appropriate limits and get to too touchy capacities and information. By abusing these imperfections, aggressors can execute favored operations and take over control of contracts. For this case, consider a contract that incorporates an admin-only "drainfunds" work to recover ether in crises. In the event that the work needs a get-to-control check for the admin part, any client seems to conjure it to take stores. Inside this classification, another high-severity defenselessness sort is Transaction-ordering reliance (TOD), which is examined in 6 essential things. This helplessness happens when the result of a keen contract's execution depends on the arrangement in which exchanges are included in a piece. Aggressors misuse this by controlling exchange orders to modify contract behavior. Terrible arbitrary number eras (Yang et al., 2022; Xu et al., 2021; Jie et al., 2023) and erratic state (Zhou et al., 2022) are the least visited powerlessness sorts in this classification. The awful arbitrary number era defenselessness happens when savvy contracts endeavor to produce arbitrariness in an uncertain way. The straightforward nature of block-chains implies such irregular values can be unsurprising or controlled by aggressors. An erratic state (energetic libraries) happens when savvy contracts stack outside library code or join third-party usefulness through interfacing like delegate calls or work calls to outside contracts. As this makes contract behavior subordinate to outside untrusted code, assailants misuse such associations. For illustration, consider a contract that depends on an outside Prophet library for sourcing basic information like resource costs. An assailant can send a noxious adaptation of this library with inaccurate information to seize the target contract's state.

4.1.4. Others Vulnerabilities

Ethereum's programmable and decentralized nature allows developers to create complex smart contracts and decentralized applications (dapp). However, this also opens up attack vectors that malicious actors can exploit in several scenarios. Attackers exploit smart contract vulnerabilities in four key ways. First, they insert code bugs to steal funds or brick contracts (Aziz et al., 2022; Aljofey et al., 2022; Kim et al., 2019; Wang et al., 2020; Liu et al., 2022; Md et al., 2023; Pahuja and Kamal). Second, they manipulate cryptography to alter outcomes (Zhang et al., 2022). Third, they create fake dapp sites to phish users and misdirect wallet connections (Aladhadh et al., 2022; Xiong et al., 2023). Fourth, they build Ponzi schemes (Zheng et al., 2023; Yu et al., 2021; Wang et al., 2021a; Gupta et al., 2022; Zheng et al., 2023; Cai et al., 2023; Goswami et al., 2021) with smart contracts, paying early investors with funds from later ones until collapse when new investors stop. Anonymity and transparency help attackers build credibility. The complexity of smart contracts, coupled with blockchain properties like immutability enables these frauds. All these scenarios can be the root causes of fraud because of the immutability, irreversibility, and transparency of transactions on Ethereum. While these qualities have secured its blockchain, the complexity of smart contract programming opens up vulnerabilities that attackers can exploit through phishing sites to ultimately defraud users.

4.2. RQ2: What are the machine learning models used for smart contracts vulnerability detection?

This section provides a thorough overview of machine learning methods for detecting security vulnerabilities in smart contracts. It begins by analyzing the distribution of machine learning models over time. Moreover, it includes a comprehensive list of the most frequently used machine learning models in the primary studies analyzed. In Fig. 4, it is evident that 57% of studies use deep learning models for smart contract vulnerability detection (Ashizawa et al., 2021; Varun et al., 2022; Sun and Gu, 2021; Lutz et al., 2021; Yu et al., 2021; Wang et al., 2021a; Gupta et al., 2022; Gogineni et al., 2020; Tann et al., 2018; Liu et al., 2021a; Aladhadh et al., 2022; Shah et al., 2023; Qian et al., 2022; Zhang & Liu, 2022; Essaid et al., 2019; Kim et al., 2019; Qian et al., 2020; Zhou et al., 2022; Hwang et al., 2022; Liu et al., 2021b; Zhang et al., 2022; Liu et al., 2022; Yashavant et al., 2022; Sun et al., 2023; Xiong et al., 2023; Jie et al., 2023), while only 32% of studies utilize supervised machine learning models (Zheng et al., 2023; Eshghie et al., 2021; Eduardo A. Sousa et al., 2021; Aziz et al., 2022; Zheng et al., 2023; Aladhadh et al., 2022; Aljofey et al., 2022; Gaur et al., 2022; Momeni et al., 2019; Wang et al., 2020; Goswami et

al., 2021; Yang et al., 2022; Song et al., 2019; Xing et al., 2020; Xu et al., 2021; Md et al., 2023; Pahuja and Kamal; Jie et al., 2023). Additionally, 4 primary studies (Aladhadh et al., 2022; Momeni et al., 2019; Xing et al., 2020; Jie et al., 2023) adopt a hybrid approach, combining both supervised and deep learning models. These techniques leverage the strengths of supervised models to incorporate rules for known vulnerabilities, while the deep learning component reveals intricate patterns in smart contract code and transactions not easily detected by supervised classifiers. Furthermore, a limited number of studies use unsupervised learning models, denoted as clustering (Fang et al., 2020).

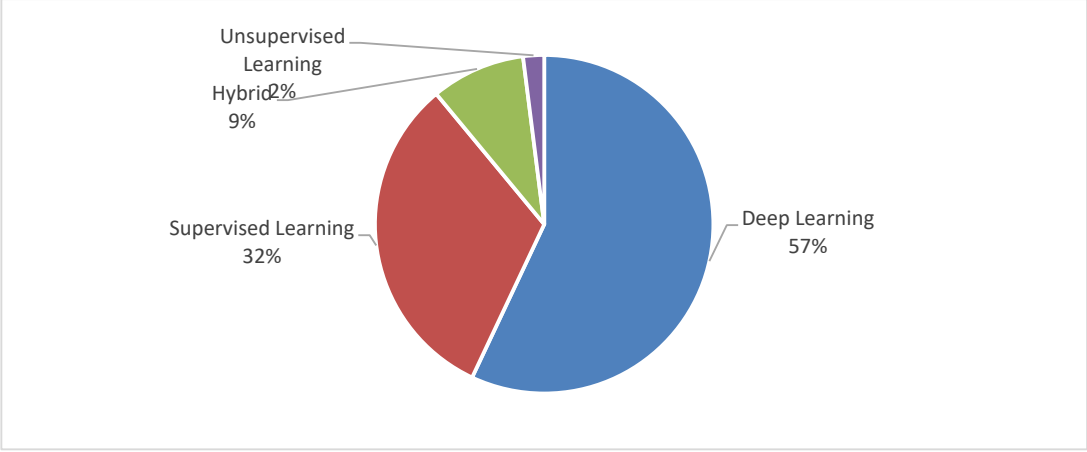

Fig. 4. Distribution of models used in primary studies

The trends depicted in Fig. 5. and Fig. 6. showcase the evolution of supervised and deep learning models in detecting smart contract vulnerabilities from 2018 to 2023, respectively. Supervised models were initially introduced in 2019, exhibiting a rising trend in usage for vulnerability detection until 2023. On the other hand, deep learning models were introduced in 2018, displaying a similar upward trajectory in usage for vulnerability detection until 2022. However, Fig. 6. indicates a declining trend in the utilization of deep learning for vulnerability detection in 2023.

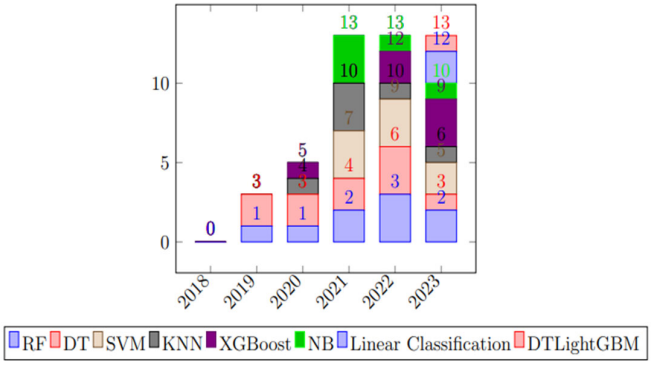

Fig. 5. Trend of supervised learning models over time

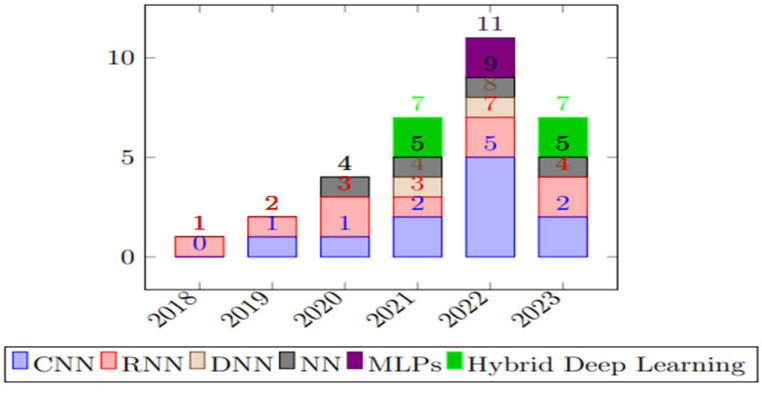

Fig. 6. Trend of deep learning models over time

Table 7 outlines the utilization of machine learning methods in primary studies. Supervised and deep learning are exclusively applied for smart contract vulnerability detection, while unsupervised, semi-supervised, and reinforcement learning methods are not utilized for this purpose. The table reveals the utilization of supervised learning models in primary studies. Random Forest (RF) emerges as the most used method, involved in 11 studies. Additionally, Decision Tree (DT) and Support Vector Machine (SVM) are popular choices, featured in 8 and 9 studies, respectively. Furthermore, K-Nearest Neighbor (KNN), Extreme Gradient Boosting (XGBoost), Naive Bayes (NB), and Logistic Regression are each present in 6, 6, 5, and 2 studies, respectively. Additionally, clustering models are featured in 1 study, where contracts are grouped by creator addresses to identify different versions by the same developers and match destructed contracts to later "upgraded" versions based on similarity.

In addition, Table 7 presents an overview of the prevalence of deep learning models in primary studies. Recurrent Neural Networks (RNN) emerge as the most utilized recurrent model, found in 9 studies, followed by Convolutional Neural Networks (CNN) with 7 occurrences. Neural networks (NN) emerge as the predominant graph-based model, appearing in 5 studies. Additionally, hybrid deep learning, Multilayer Perceptrons (MLPs), and A Deep Neural Network (DNN) are frequently employed, accounting for 4, 3, and 2 studies, respectively.

Table 7
Distribution of Machine Learning models in primary studies

Machine Learning Method	Method Name	No. Studies	References
Supervised Learning	RF	11	[Eshghie et al. (2021), Eduardo A. Sousa et al. (2021), Mandloi and Bansal (2022), Momeni et al. (2019), Wang et al. (2020), Yang et al. (2022), Song et al. (2019), Xing et al. (2020), Md et al. (2023), Pahuja and Kamal, Jie et al. (2023)]
	DT	8	[Eduardo A. Sousa et al. (2021),Aziz et al. (2022), Mandloi and Bansal (2022), Momeni et al. (2019), Wang et al. (2020), Goswami et al. (2021), Md et al. (2023), Pahuja and Kamal]
	SVM	9	[Eshghie et al. (2021), Eduardo A. Sousa et al. (2021), Zheng et al. (2023), Aladhadh et al. (2022), Gaur et al. (2022), Momeni et al. (2019), Goswami et al. (2021), Md et al. (2023), Pahuja and Kamal]
	KNN	6	[Eshghie et al. (2021), Eduardo A. Sousa et al. (2021), Wang et al. (2020), Xu et al. (2021), Md et al. (2023), Pahuja and Kamal]
	XGBoost	6	[Zheng et al. (2023), Zheng et al. (2023), Aljofey et al. (2022), Wang et al. (2020), Md et al. (2023), Pahuja and Kamal]
	NB	5	[Eshghie et al. (2021), Eduardo A. Sousa et al. (2021), Goswami et al. (2021), Md et al. (2023), Pahuja and Kamal]
	Linear Classification	2	[Zheng et al. (2023), Pahuja and Kamal]
	DTLightGBM	1	[Pahuja and Kamal]
-	SUM	48(19)	-
Deep Learning	RNNLSTM	9	[Varun et al. (2022), Wang et al. (2021a), Gupta et al. (2022), Gogineni et al. (2020), Tann et al. (2018), Shah et al. (2023), Qian et al. (2022), Essaid et al. (2019), Qian et al. (2020)]
	CNN	7	[Sun and Gu (2021), Qian et al. (2022), Kim et al. (2019), Zhou et al. (2022), Hwang et al. (2022), Liu et al. (2022), Xing et al. (2020)]
	NN	5	[Ashizawa et al. (2021), Cai et al. (2023), Momeni et al. (2019), Yashavant et al. (2022), Xing et al. (2020)]
	CNNGCN	4	[Yu et al. (2021), Zhang and Liu (2022), Xiong et al. (2023), Jie et al. (2023)]
	RNN	3	[Qian et al. (2022), Essaid et al. (2019), Jie et al. (2023)]
	MLPs	2	[Varun et al. (2022), Aladhadh et al. (2022)]
	DNN	2	[Lutz et al. (2021), Zhang et al. (2022)]
	RNNGRU	2	[Gupta et al. (2022), Shah et al. (2023)]
	Hybrid Deep Learning	4	[Liu et al. (2021a), Liu et al. (2021b), Sun et al. (2023),Narayana and Sathiyamurthy (2023)]
-	SUM	37(30)	-

4.3. RQ3: What are the sources and types of datasets used in smart contract security using machine learning?

ML-based smart contract vulnerability detection models require data to be built and evaluated (Ashizawa et al., 2021; Zhang and Liu, 2022; Kim et al., 2019; Wang et al., 2020; Yang et al., 2022; Liu et al., 2021b; Song et al., 2019). Several elements contribute to evaluating the quality of a dataset, including the source, size, scale, type, and preprocessing steps. Inappropriate data preprocessing can lead to suboptimal ML model performance. For instance, mishandling data preprocessing can result in inadequate ML model performance (Kim et al., 2019). In this section, we examine the steps involved in collecting, sorting, and representing data used in vulnerability detection studies. The need for accessible information for preparation has been

recognized as a noteworthy deterrent in utilizing machine learning for identifying shrewd contract vulnerabilities (Durieux et al., 2020). Subsequently, there's a need to inquire about tending to strategies for procuring satisfactory datasets to prepare machine learning models for powerlessness discovery forms. To address this issue, we analyzed dataset sources from 46 essential considerations. Based on our investigation, we found that datasets for this reason can be categorized into three primary bunches: benchmarks, collected information, and cross-breed information. These benchmarks comprise standardized datasets reasonable for evaluating the adequacy of helplessness location strategies and procedures (Ashizawa et al., 2021; Sun and Gu, 2021; Wang et al., 2021a; Gupta et al., 2022; Tann et al., 2018; Liu et al., 2021a; Aljofey et al., 2022; Shah et al., 2023; Zhang and Liu, 2022; Cai et al., 2023; Momeni et al., 2019; Kim et al., 2019; Tooth et al., 2020; Zhou et al., 2022; Hwang et al., 2022; Yang et al., 2022; Liu et al., 2022; Yashavant et al., 2022; Sun et al., 2023; Xu et al., 2021; Md et al., 2023; Jie et al., 2023).

Open sources and genuine Ethereum systems are as often as possible utilized as benchmarks for recognizing savvy contract vulnerabilities (Zhang and Liu, 2022; Cai et al., 2023; Hwang et al., 2022; Sun et al., 2023; Xu et al., 2021). As a portion of this exertion, we point to gather an assorted set of contracts that speak to a wide assortment of decentralized applications, savvy contract dialects, and sorts of vulnerabilities. From the collected information, particular contracts are selected to be included within the benchmark dataset. It is critical to consider components such as contract complexity and defenselessness diversity when selecting a contract. The objective is to form a dataset that envelops different sorts of vulnerabilities and precisely reflects real-world circumstances. In certain cases, benchmark datasets may incorporate engineered keen contracts inferred from controlled helplessness infusion methods. These contracts are ordinarily made utilizing code-generation strategies and follow particular designs or layouts. Through the engineered era, large-scale datasets can be made, covering a more extensive run of vulnerabilities. Also, benchmark datasets envelop honest-to-goodness savvy contract applications or handwritten code parts. By physically making these cases, we guarantee that the dataset joins the vulnerabilities agent of actual coding hones. To form shrewd contracts physically, one must recognize vulnerabilities within the source code, present fitting shortcomings, and keep up an adjustment between rightness and authenticity.

There are either freely accessible datasets from stores such as GitHub or Kaggle, or the creators have made their datasets (Zheng et al., 2023; Aziz et al., 2022; Aladhadh et al., 2022; Qian et al., 2022; Wang et al., 2020; Goswami et al., 2021; Liu et al., 2021b; Xiong et al., 2023). Moreover, a few consider combining different sources for powerlessness discovery to improve the precision of comes about (Wang et al., 2021a; Liu et al., 2021a; Aljofey et al., 2022; Zhang & Liu, 2022; Cai et al., 2023; Kim et al., 2019; Hwang et al., 2022; Liu et al., 2021b; Sun et al., 2023; Xu et al., 2021; Md et al., 2023), which is alluded to as crossover sources in this consider.

The distribution of dataset sources in the primary studies is shown in Fig. 7. Based on our analysis, 54 % of primary studies used collected datasets to detect smart contract vulnerabilities as shown in Table 8. It is believed that this trend is because smart contract analysis is a relatively new field of research and there are not widely used, standardized benchmarks for tasks such as vulnerability detection. Researchers thus must collect data from multiple resources to capture a wide range of vulnerabilities, code structures, and coding practices present in the actual smart contracts used in various applications. Researchers rely on benchmark datasets to assess and compare the efficacy of various ML models. Our perceptions uncover that 33% of the things about developing defenselessness location models utilize these benchmark datasets. This drift can be ascribed to a few variables. Firstly, the field of savvy contract examination is still in its early stages and experiencing fast advancement, driving a need for comprehensive and standardized benchmark datasets. Furthermore, the nonattendance of a bound-together approach for gathering and organizing vulnerabilities inside the Ethereum environment poses challenges in making broadly acknowledged benchmark datasets. Besides, the approval of these datasets, counting the foundation of ground truth for vulnerabilities, may be a multifaceted and time-intensive handle, encouraging restricting their accessibility.

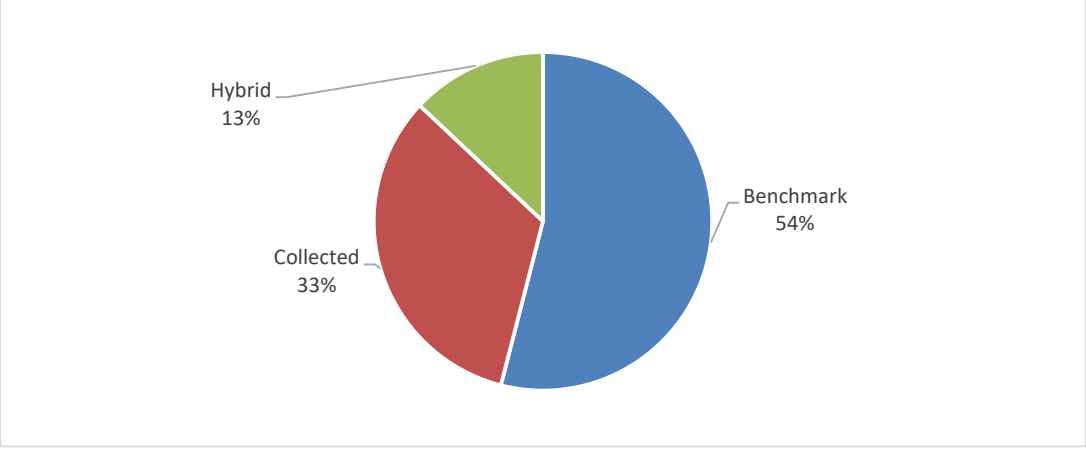

Fig. 7. The source of the datasets used in primary study papers

Table 8
Detailed distribution of collected sources

Source of dataset	No. Studies	References
Ethereum	6	[Lutz et al. (2021), Wang et al. (2021a), Gogineni et al. (2020), Goswami et al. (2021), Zhang et al. (2022), Song et al. (2019)]
Website	5	[Zheng et al. (2023), Qian et al. (2022), Wang et al. (2020), Goswami et al. (2021), Xiong et al. (2023)]
Kaggle	3	[Aziz et al. (2022), Md et al. (2023), Pahuja and Kamal]
Ethereum Classic (ETC)	2	[Aziz et al. (2022), Aladhadh et al. (2022)]
Github	2	[Aladhadh et al. (2022), Aljofey et al. (2022), Narayana and Sathiyamurthy (2023)]
SmartBugs Wild	2	[Cai et al. (2023), Xu et al. (2021)]
PonziContractDataset	1	[Zheng et al. (2023)]
RSC and RCS	1	[Qian et al. (2020)]
ESC (Ethereum Smart Contracts)	1	[Liu et al. (2021b)]
VSC (VNT chain Smart Contracts)	1	[Liu et al. (2021b)]
ScrawlD	1	[Yashavant et al. (2022)]
Others	11	[Varun et al. (2022), Eshghie et al. (2021), Eduardo A. Sousa et al. (2021), Yu et al. (2021), Mandloi and Bansal (2022), Zheng et al. (2023), Gaur et al. (2022), Cai et al. (2023), Essaid et al. (2019), Fang et al. (2020), Xing et al. (2020)]
SUM	37(31)	-

The third noteworthy information source, bookkeeping for 13% of essential ponders, is half-breed, combining different sources. Analysts habitually utilize half-breed sources within the powerlessness location to prepare to ease the limitations of personal information sources and procure more changed and comprehensive datasets. For example, analysts can generalize their machine learning models to distinguish a wide run of shortcomings by amalgamating information from benchmark datasets with information from stages such as GitHub, open-source ventures, or websites to develop a half-breed dataset that better mirrors real-world scenarios. Table 9 presents a comprehensive breakdown of the benchmark information utilized within the essential things about. Notably, Etherscan developed as the foremost as often as possible utilized source of benchmark datasets due to its expansive open dataset containing a confirmed collection of keen contracts, which is fundamental for preparing and testing machine learning models (Zheng et al., 2023). The accessibility of these materials advances repeatability and collaboration among analysts included in shrewd contract defenselessness location. In addition, 22 particular essential things about utilizing benchmark datasets sourced from different sources.

Table 9
Detailed distribution of benchmark sources

Source of dataset	No. Studies	References
Etherscan	12	[Ashizawa et al. (2021), Sun and Gu (2021), Wang et al. (2021a), Aljofey et al. (2022), Momeni et al. (2019), Kim et al. (2019), Zhou et al. (2022), Yang et al. (2022), Liu et al. (2022), Yashavant et al. (2022), Md et al. (2023)]
SmartBug	5	[Eshghie et al. (2021), Zhang and Liu (2022), Cai et al. (2023), Hwang et al. (2022), Sun et al. (2023), Xu et al. (2021)]
SolidFI	5	[Zhang and Liu (2022), Cai et al. (2023), Hwang et al. (2022), Sun et al. (2023), Xu et al. (2021)]
Google BigQuery	3	[Gupta et al. (2022), Tann et al. (2018), Kim et al. (2019)]
SoliAudit	1	[Sun et al. (2023)]
HuanGai ManualCheckDatase	1	[Cai et al. (2023)]
CICDDoS2019	1	[Liu et al. (2021a)]
BoT-IoT	1	[Liu et al. (2021a)]
CICIDS2017	1	[Liu et al. (2021a)]
SmartEmbed	1	[Jie et al. (2023)]
X-IIoTID	1	[Shah et al. (2023)]
Etherscamdb	1	[Md et al. (2023)]

On the other hand, datasets utilized for helplessness discovery can envelop changing information sorts. For occasion, machine learning models for defenselessness location can distinguish vulnerabilities in different information designs, counting source code, bytecode, exchanges, addresses, squares, or combinations thereof. It's vital to scrutinize these information sorts fastidiously, as they require particular preprocessing methods and must be spoken to unexpectedly when utilizing machine learning models. In addition, diverse information sorts call for different structural approaches in machine learning models. Table 10 provides a detailed breakdown of the specific data types employed in primary studies. It reveals that 12 primary studies exclusively utilized source code (Cai et al., 2023; Momeni et al., 2019; Kim et al., 2019; Zhou et al., 2022; Yang et al., 2022; Liu et al., 2021b, 2022; Yashavant et al., 2022; Song et al., 2019; Xu et al., 2021; Jie et al., 2023). According to our analysis,

the majority of primary studies employed a combination of various data types for detecting security vulnerabilities in contracts, totaling 18 primary studies. Notably, source code and bytecode emerge as the most prevalent data type combinations (Ashizawa et al., 2021; Sun and Gu, 2021; Zheng et al., 2023; Qian et al., 2022; Zhang and Liu, 2022; Hwang et al., 2022; Zhang et al., 2022).

Table 10
Data types of datasets involved in primary studies

Data Type	No. Studies	References
Source code	12	[Cai et al. (2023), Momeni et al. (2019), Qian et al. (2020), Zhou et al. (2022), Yang et al. (2022), Liu et al. (2021b), Liu et al. (2022), Yashavant et al. (2022), Song et al. (2019), Xu et al. (2021), Jie et al. (2023), Narayana and Sathiyamurthy (2023)]
Transaction	7	[Varun et al. (2022), Eshghie et al. (2021), Eduardo A. Sousa et al. (2021), Aziz et al. (2022), Mandloi and Bansal (2022), Fang et al. (2020), Pahuja and Kamal]
Source code + Bytecode	7	[Ashizawa et al. (2021), Sun and Gu (2021), Zheng et al. (2023), Qian et al. (2022), Zhang and Liu (2022), Hwang et al. (2022), Zhang et al. (2022)]
Source code + Bytecode + Transactions	4	[Zheng et al. (2023), Aljofey et al. (2022), Kim et al. (2019), Sun et al. (2023)]
Bytecode	2	[Wang et al. (2020), Xing et al. (2020)]
Bytecode + Addresses	2	[Gogineni et al. (2020), Tann et al. (2018)]
Transactions + Addresses	2	[Yu et al. (2021), Md et al. (2023)]
Bytecode + Transactions + Blocks + Addresses	1	[Lutz et al. (2021)]
Bytecode + Transactions + Source code + Addresses	1	[Goswami et al. (2021)]
Source code + Bytecode + Transactions + Blocks	1	[Wang et al. (2021a)]
Addresses	1	[Xiong et al. (2023)]
Others	6	[Gupta et al. (2022), Liu et al. (2021a), Aladhadh et al. (2022), Gaur et al. (2022), Shah et al. (2023), Essaid et al. (2019)]

5. Threats to Validity

Several validity threats should be considered when conducting a systematic literature review on ML for detecting smart contract vulnerabilities:

- Publication bias risks omitting relevant papers, distorting results (Vegendla et al., 2018). Comprehensive database searches, manual collection, and appropriate search terms help mitigate this.
- Search strategy limitations may cause relevant studies to be missed, reducing comprehensiveness.
- Findings may not generalize broadly due to dataset, model, and other limiting factors.
- The Evolution of ML and smart contract technologies may quickly render findings outdated.
- Interpretation bias may influence synthesis of results.
- Timeline restrictions to July 2023 omit more recent relevant papers.

With rapidly emerging research, it is important to monitor new technologies to ensure currency. Enhancing the review by expanding data sources, dataset characteristics, vulnerabilities, and ML models can increase robustness.

6. Conclusions

Ethereum smart contracts play a pivotal role in digital agreements but have substantial security concerns in managing significant sums of virtual currency. This SLR systematically surveyed 46 primary studies on using machine learning models for detecting vulnerabilities in smart contracts, structured around three research questions. Key findings show the most prevalent vulnerability addressed is Re-entrancy, featured in 23 studies, and growing usage of ML, particularly deep learning, for detection. 57% of studies employed DL models categorized across architectures, with LSTM RNNs and CNNs being the most prominent. Further examination revealed that 54% of studies used collected datasets. The review underscores the significance of ML techniques in addressing smart contract security challenges. Key future directions involve enhancing model robustness to new vulnerabilities through continuous learning, multi-language support, advanced features, and addressing data challenges around labeling, imbalance, and availability. Advancing analysis necessitates a focus on generalization, representation, completeness, and accessibility. Finally, key future directions involve enhancing model robustness to new vulnerabilities through continuous learning, multi-language support, and advanced features while also addressing underlying data challenges like labeling reliability, imbalance, and availability dependence.

References

Aladhadh, S., Alwabli, H., Moulahi, T., & Al Asqah, M. (2022). Bchainguard: a new framework for cyberthreats detection in blockchain using machine learning. *Applied Sciences*, *12*(23), 12026.

Alharby, M., & Van Moorsel, A. (2017). A systematic mapping study on current research topics in smart contracts. *Available at SSRN 3876872*.

Alharby, M., & Van Moorsel, A. (2017). Blockchain-based smart contracts: A systematic mapping study. *arXiv preprint arXiv:1710.06372*.

Aljofey, A., Rasool, A., Jiang, Q., & Qu, Q. (2022). A feature-based robust method for abnormal contracts detection in ethereum blockchain. *Electronics*, *11*(18), 2937.

Al-Shamayleh, A. S., Ahmad, R., Abushariah, M. A., Alam, K. A., & Jomhari, N. (2018). A systematic literature review on vision based gesture recognition techniques. *Multimedia Tools and Applications, 77,* 28121-28184.

Ashizawa, N., Yanai, N., Cruz, J. P., & Okamura, S. (2021, May). 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* (pp. 47-59).

Atzei, N., Bartoletti, M., & Cimoli, T. (2017). A survey of attacks on ethereum smart contracts (sok). In *Principles of Security and Trust: 6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings 6* (pp. 164-186). Springer Berlin Heidelberg.

Aziz, R. M., Baluch, M. F., Patel, S., & Kumar, P. (2022). A machine learning based approach to detect the Ethereum fraud transactions with limited attributes. *Karbala International Journal of Modern Science*, *8*(2), 139-151.

Cai, J., Li, B., Zhang, J., Sun, X., & Chen, B. (2023). Combine sliced joint graph with graph neural networks for smart contract vulnerability detection. *Journal of Systems and Software*, *195*, 111550.

Chen, H., Pendleton, M., Njilla, L., & Xu, S. (2020). A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. *ACM Computing Surveys (CSUR)*, *53*(3), 1-43.

Chen, J., Xia, X., Lo, D., Grundy, J., & Yang, X. (2021). Maintenance-related concerns for post-deployed Ethereum smart contract development: issues, techniques, and future challenges. *Empirical Software Engineering*, *26*(6), 117.

Chithanuru, V., & Ramaiah, M. (2023). An anomaly detection on blockchain infrastructure using artificial intelligence techniques: Challenges and future directions–A review. *Concurrency and Computation: Practice and Experience*, *35*(22), e7724.

de la Rosa, J. L., Gibovic, D., Torres, V., Maicher, L., Miralles, F., El-Fakdi, A., & Bikfalvi, A. (2016, December). On intellectual property in online open innovation for SME by means of blockchain and smart contracts. In *3rd Annual World Open Innovation Conf. WOIC*.

Durieux, T., Ferreira, J. F., Abreu, R., & Cruz, P. (2020, June). Empirical review of automated analysis tools on 47,587 ethereum smart contracts. In *Proceedings of the ACM/IEEE 42nd International conference on software engineering* (pp. 530-541).

Eduardo A. Sousa, J., Oliveira, V. C., Almeida Valadares, J., Borges Vieira, A., Bernardino, H. S., Moraes Villela, S., & Dias Goncalves, G. (2021). Fighting under-price DoS attack in ethereum with machine learning techniques. *ACM SIGMETRICS Performance Evaluation Review*, *48*(4), 24-27.

Eshghie, M., Artho, C., & Gurov, D. (2021, June). 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).

Essaid, M., Kim, D., Maeng, S. H., Park, S., & Ju, H. T. (2019, September). A collaborative DDoS mitigation solution based on ethereum smart contract and RNN-LSTM. In *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)* (pp. 1-6). IEEE.

Fang, L., Zhao, B., Li, Y., Liu, Z., Ge, C., & Meng, W. (2020). Countermeasure based on smart contracts and AI against DoS/DDoS attack in 5G circumstances. *IEEE Network*, *34*(6), 54-61.

Garg, K., Saraswat, P., Bisht, S., Aggarwal, S. K., Kothuri, S. K., & Gupta, S. (2019, April). A comparitive analysis on e-voting system using blockchain. In *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)* (pp. 1-4). IEEE.

Gaur, R., Prakash, S., Kumar, S., Abhishek, K., Msahli, M., & Wahid, A. (2022). A machine-learning–blockchain-based authentication using smart contracts for an ioht system. *Sensors*, *22*(23), 9074.

Gogineni, A. K., Swayamjyoti, S., Sahoo, D., Sahu, K. K., & Kishore, R. (2020). Multi-Class classification of vulnerabilities in Smart Contracts using AWD-LSTM, with pre-trained encoder inspired from natural language processing. *IOP SciNotes*, *1*(3), 035002.

Goswami, S., Singh, R., Saikia, N., Bora, K. K., & Sharma, U. (2021, August). TokenCheck: towards deep learning based security vulnerability detection in ERC-20 tokens. In *2021 IEEE Region 10 Symposium (TENSYMP)* (pp. 1-8). IEEE.

Grishchenko, I., Maffei, M., & Schneidewind, C. (2018). Foundations and tools for the static analysis of ethereum smart contracts. In *Computer Aided Verification: 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I 30* (pp. 51-78). Springer International Publishing.

Gupta, R., Patel, M. M., Shukla, A., & Tanwar, S. (2022). Deep learning-based malicious smart contract detection scheme for internet of things environment. *Computers & Electrical Engineering*, *97*, 107583.

Harz, D., & Knottenbelt, W. (2018). Towards safer smart contracts: A survey of languages and verification methods. *arXiv preprint arXiv:1809.09805*.

Hasanova, H., Baek, U. J., Shin, M. G., Cho, K., & Kim, M. S. (2019). A survey on blockchain cybersecurity vulnerabilities and possible countermeasures. *International Journal of Network Management*, *29*(2), e2060.

Huang, Y., Bian, Y., Li, R., Zhao, J. L., & Shi, P. (2019). Smart contract security: A software lifecycle perspective. *IEEE Access*, *7*, 150184-150202.

Hwang, S. J., Choi, S. H., Shin, J., & Choi, Y. H. (2022). CodeNet: Code-targeted convolutional neural network architecture for smart contract vulnerability detection. *IEEE Access*, *10*, 32595-32607.

Ibba, G., Pierro, G. A., & Di Francesco, M. (2021, May). Evaluating machine-learning techniques for detecting smart ponzi schemes. In *2021 IEEE/ACM 4th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)* (pp. 34-40). IEEE.

Imperius, N. P., & Alahmar, A. D. (2022). Systematic Mapping of Testing Smart Contracts for Blockchain Applications. *IEEE Access*, *10*, 112845-112857.

Ivanov, N., Li, C., Yan, Q., Sun, Z., Cao, Z., & Luo, X. (2023). Security threat mitigation for smart contracts: A comprehensive survey. *ACM Computing Surveys*, *55*(14s), 1-37.

Jiang, F., Chao, K., Xiao, J., Liu, Q., Gu, K., Wu, J., & Cao, Y. (2023). Enhancing smart-contract security through machine learning: A survey of approaches and techniques. *Electronics*, *12*(9), 2046.

Jie, W., Chen, Q., Wang, J., Koe, A. S. V., Li, J., Huang, P., ... & Wang, Y. (2023). A novel extended multimodal AI framework towards vulnerability detection in smart contracts. *Information Sciences*, *636*, 118907.

Keele, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

Kim, Y., Pak, D., & Lee, J. (2019). ScanAT: identification of bytecode-only smart contracts with multiple attribute tags. *IEEE Access*, *7*, 98669-98683.

Kirillov, D., Iakushkin, O., Korkhov, V., & Petrunin, V. (2019). Evaluation of tools for analyzing smart contracts in distributed ledger technologies. In *Computational Science and Its Applications–ICCSA 2019: 19th International Conference, Saint Petersburg, Russia, July 1–4, 2019, Proceedings, Part II 19* (pp. 522-536). Springer International Publishing.

Kirli, D., Couraud, B., Robu, V., Salgado-Bravo, M., Norbu, S., Andoni, M., ... & Kiprakis, A. (2022). Smart contracts in energy systems: A systematic review of fundamental approaches and implementations. *Renewable and Sustainable Energy Reviews*, *158*, 112013.

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, *33*(2004), 1-26.

Kushwaha, S. S., Joshi, S., Singh, D., Kaur, M., & Lee, H. N. (2022). Systematic review of security vulnerabilities in ethereum blockchain smart contract. *IEEE Access*, *10*, 6605-6621.

Litke, A., Anagnostopoulos, D., & Varvarigou, T. (2019). Blockchains for supply chain management: Architectural elements and challenges towards a global scale deployment. *Logistics*, *3*(1), 5.

Liu, L., Tsai, W. T., Bhuiyan, M. Z. A., Peng, H., & Liu, M. (2022). Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum. *Future Generation Computer Systems*, *128*, 158-166.

Liu, T., Sabrina, F., Jang-Jaccard, J., Xu, W., & Wei, Y. (2021). Artificial intelligence-enabled DDoS detection for blockchain-based smart transport systems. *Sensors*, *22*(1), 32.

Liu, Z., Qian, P., Wang, X., Zhuang, Y., Qiu, L., & Wang, X. (2021). Combining graph neural networks with expert knowledge for smart contract vulnerability detection. *IEEE Transactions on Knowledge and Data Engineering*, *35*(2), 1296-1310.

Lutz, O., Chen, H., Fereidooni, H., Sendner, C., Dmitrienko, A., Sadeghi, A. R., & Koushanfar, F. (2021). Escort: ethereum smart contracts vulnerability detection using deep neural network and transfer learning. *arXiv preprint arXiv:2103.12607*.

Macrinici, D., Cartofeanu, C., & Gao, S. (2018). Smart contract applications within blockchain technology: A systematic mapping study. *Telematics and Informatics*, *35*(8), 2337-2354.

Mandloi, J., & Bansal, P. (2022). A machine learning-based dynamic method for detecting vulnerabilities in smart contracts. *International Journal of Applied Engineering & Technology*, *4*, 110-118.

Md, A. Q., Narayanan, S. S. S., Sabireen, H., Sivaraman, A. K., & Tee, K. F. (2023). A novel approach to detect fraud in Ethereum transactions using stacking. *Expert Systems*, *40*(7), e13255.

Mense, A., & Flatscher, M. (2018, November). Security vulnerabilities in ethereum smart contracts. In *Proceedings of the 20th international conference on information integration and web-based applications & services* (pp. 375-380).

Miller, A., Cai, Z., & Jha, S. (2018). Smart contracts and opportunities for formal methods. In *Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice: 8th International Symposium, ISoLA 2018, Limassol, Cyprus, November 5-9, 2018, Proceedings, Part IV 8* (pp. 280-299). Springer International Publishing.

Mololoth, V. K., Saguna, S., & Åhlund, C. (2023). Blockchain and machine learning for future smart grids: A review. *Energies*, *16*(1), 528.

Momeni, P., Wang, Y., & Samavi, R. (2019, August). Machine learning model for smart contracts security analysis. In *2019 17th International Conference on Privacy, Security and Trust (PST)* (pp. 1-6). IEEE.

Narayana, K. L., & Sathiyamurthy, K. (2023). Automation and smart materials in detecting smart contracts vulnerabilities in Blockchain using deep learning. *Materials Today: Proceedings*, *81*, 653-659.

Pahuja, L., & Kamal, A. (2023). EnLEFD-DM: Ensemble Learning Based Ethereum Fraud Detection Using CRISP-DM Framework. *Expert Systems*, *40*(9), e13379.

Piantadosi, V., Rosa, G., Placella, D., Scalabrino, S., & Oliveto, R. (2023). Detecting functional and security-related issues in smart contracts: A systematic literature review. *Software: Practice and Experience*, *53*(2), 465-495.

Qian, P., Liu, Z., He, Q., Zimmermann, R., & Wang, X. (2020). Towards automated reentrancy detection for smart contracts based on sequential models. *IEEE Access*, *8*, 19685-19695.

Qian, S., Ning, H., He, Y., & Chen, M. (2022). Multi-label vulnerability detection of smart contracts based on Bi-LSTM and attention mechanism. *Electronics*, *11*(19), 3260.

Raja, G., Manaswini, Y., Vivekanandan, G. D., Sampath, H., Dev, K., & Bashir, A. K. (2020, July). AI-powered blockchain-a decentralized secure multiparty computation protocol for IoV. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (pp. 865-870). IEEE.

Rameder, H., Di Angelo, M., & Salzer, G. (2022). Review of automated vulnerability analysis of smart contracts on Ethereum. *Frontiers in Blockchain*, *5*, 814977.

Rouhani, S., & Deters, R. (2019). Security, performance, and applications of smart contracts: A systematic survey. *IEEE Access*, *7*, 50759-50779.

Shah, H., Shah, D., Jadav, N. K., Gupta, R., Tanwar, S., Alfarraj, O., ... & Marina, V. (2023). Deep learning-based malicious smart contract and intrusion detection system for IoT environment. *Mathematics*, *11*(2), 418.

Shorman, A., Sabri, K. E., Abushariah, M., & Qaimari, M. (2020). Blockchain for banking systems: Opportunities and challenges. *Journal of Theoretical and Applied Information Technology*, *98*(23), 3703-3717.

Song, J., He, H., Lv, Z., Su, C., Xu, G., & Wang, W. (2019). An efficient vulnerability detection model for ethereum smart contracts. In *Network and System Security: 13th International Conference, NSS 2019, Sapporo, Japan, December 15–18, 2019, Proceedings 13* (pp. 433-442). Springer International Publishing.

Sun, X., Tu, L., Zhang, J., Cai, J., Li, B., & Wang, Y. (2023). ASSBert: Active and semi-supervised bert for smart contract vulnerability detection. *Journal of Information Security and Applications*, *73*, 103423.

Sun, Y., & Gu, L. (2021, March). Attention-based machine learning model for smart contract vulnerability detection. In *Journal of physics: conference series* (Vol. 1820, No. 1, p. 012004). IOP Publishing.

Sürücü, O., Yeprem, U., Wilkinson, C., Hilal, W., Gadsden, S. A., Yawney, J., ... & Giuliano, A. (2022). A survey on ethereum smart contract vulnerability detection using machine learning. *Disruptive Technologies in Information Sciences VI*, *12117*, 110-121.

Tann, W. J. W., Han, X. J., Gupta, S. S., & Ong, Y. S. (2018). Towards safer smart contracts: A sequence learning approach to detecting security threats. *arXiv preprint arXiv:1811.06632*.

Tikhomirov, S. (2018). Ethereum: state of knowledge and research perspectives. In *Foundations and Practice of Security: 10th International Symposium, FPS 2017, Nancy, France, October 23-25, 2017, Revised Selected Papers 10* (pp. 206-221). Springer International Publishing.

Timucin, T., & BİROĞUL, S. (2021). A survey: Making "Smart Contracts" really smart. *Transactions on Emerging Telecommunications Technologies*, *32*(11), e4338.

Varun, M., Palanisamy, B., & Sural, S. (2022, May). Mitigating frontrunning attacks in ethereum. In *Proceedings of the Fourth ACM International Symposium on Blockchain and Secure Critical Infrastructure* (pp. 115-124).

Varun, M., Palanisamy, B., & Sural, S. (2022, May). Mitigating frontrunning attacks in ethereum. In *Proceedings of the Fourth ACM International Symposium on Blockchain and Secure Critical Infrastructure* (pp. 115-124).

Vegendla, A., Duc, A. N., Gao, S., & Sindre, G. (2018). A systematic mapping study on requirements engineering in software ecosystems. *Journal of Information Technology Research (JITR)*, *11*(1), 49-69.

Virani, H., & Kyada, M. (2022). A Systematic Literature Review on Smart Contracts Security. *arXiv preprint arXiv:2212.05099*.

Wang, L., Cheng, H., Zheng, Z., Yang, A., & Zhu, X. (2021). Ponzi scheme detection via oversampling-based long short-term memory for smart contracts. *Knowledge-Based Systems*, *228*, 107312.

Wang, W., Song, J., Xu, G., Li, Y., Wang, H., & Su, C. (2020). Contractward: Automated vulnerability detection models for ethereum smart contracts. *IEEE Transactions on Network Science and Engineering*, *8*(2), 1133-1144.

Wang, Y., He, J., Zhu, N., Yi, Y., Zhang, Q., Song, H., & Xue, R. (2021). Security enhancement technologies for smart contracts in the blockchain: A survey. *Transactions on Emerging Telecommunications Technologies*, *32*(12), e4341.

Wöhrer, M., & Zdun, U. (2018, July). Design patterns for smart contracts in the ethereum ecosystem. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* (pp. 1513-1520). IEEE.

Xing, C., Chen, Z., Chen, L., Guo, X., Zheng, Z., & Li, J. (2020). A new scheme of vulnerability analysis in smart contract with machine learning. *Wireless Networks*, 1-10.

Xiong, A., Tong, Y., Jiang, C., Guo, S., Shao, S., Huang, J., ... & Qi, B. (2023). Ethereum phishing detection based on graph neural networks. *IET Blockchain*.

Xu, Y., Hu, G., You, L., & Cao, C. (2021). A novel machine learning-based analysis model for smart contract vulnerability. *Security and Communication Networks*, *2021*, 1-12.

Yang, H., Zhang, J., Gu, X., & Cui, Z. (2022, October). Smart contract vulnerability detection based on abstract syntax tree. In *2022 8th International Symposium on System Security, Safety, and Reliability (ISSSR)* (pp. 169-170). IEEE.

Yashavant, C. S., Kumar, S., & Karkare, A. (2022). Scrawld: A dataset of real world ethereum smart contracts labelled with vulnerabilities. *arXiv preprint arXiv:2202.11409*.

Yu, S., Jin, J., Xie, Y., Shen, J., & Xuan, Q. (2021). Ponzi scheme detection in ethereum transaction network. In *Blockchain and Trustworthy Systems: Third International Conference, BlockSys 2021, Guangzhou, China, August 5–6, 2021, Revised Selected Papers 3* (pp. 175-186). Springer Singapore.

Zhang, L., Wang, J., Wang, W., Jin, Z., Su, Y., & Chen, H. (2022). Smart contract vulnerability detection combined with multi-objective detection. *Computer Networks*, *217*, 109289.

Zhang, Y., & Liu, D. (2022). Toward vulnerability detection for ethereum smart contracts using graph-matching network. *Future Internet*, *14*(11), 326.

Zheng, Z., Chen, W., Zhong, Z., Chen, Z., & Lu, Y. (2023). Securing the ethereum from smart ponzi schemes: Identification using static features. *ACM Transactions on Software Engineering and Methodology*, *32*(5), 1-28.

Zhou, Q., Zheng, K., Zhang, K., Hou, L., & Wang, X. (2022). Vulnerability analysis of smart contract for blockchain-based IoT applications: a machine learning approach. *IEEE Internet of Things Journal*, *9*(24), 24695-24707.