electronics

Review

Ethereum Smart Contract Vulnerability Detection and Machine Learning-Driven Solutions: A Systematic Literature Review

Rasoul Kiani and Victor S. Sheng *

Department of Computer Science, Texas Tech University, Lubbock, TX 79409, USA; rasoul.kiani87@yahoo.com
* Correspondence: victor.sheng@ttu.edu

Abstract: In recent years, emerging trends like smart contracts (SCs) and blockchain have promised to bolster data security. However, SCs deployed on Ethereum are vulnerable to malicious attacks. Adopting machine learning methods is proving to be a satisfactory alternative to conventional vulnerability detection techniques. Nevertheless, most current machine learning techniques depend on sufficient expert knowledge and solely focus on addressing well-known vulnerabilities. This paper puts forward a systematic literature review (SLR) of existing machine learning-based frameworks to address the problem of vulnerability detection. This SLR follows the PRISMA statement, involving a detailed review of 55 papers. In this context, we classify recently published algorithms under three different machine learning perspectives. We explore state-of-the-art machine learning-driven solutions that deal with the class imbalance issue and unknown vulnerabilities. We believe that algorithmic-level approaches have the potential to provide a clear edge over data-level methods in addressing the class imbalance issue. By emphasizing the importance of the positive class and correcting the bias towards the negative class, these approaches offer a unique advantage. This unique feature can improve the efficiency of machine learning-based solutions in identifying various vulnerabilities in SCs. We argue that the detection of unknown vulnerabilities suffers from the absence of a unique definition. Moreover, current frameworks for detecting unknown vulnerabilities are structured to tackle vulnerabilities that exist objectively.

Keywords: data security; smart contract; Ethereum; vulnerability detection; machine learning; class imbalance; unknown vulnerabilities

check for updates

Citation: Kiani, R.; Sheng, V.S. Ethereum Smart Contract Vulnerability Detection and Machine Learning-Driven Solutions: A Systematic Literature Review. *Electronics* **2024**, *13*, 2295. https://doi.org/10.3390/electronics13122295

Academic Editors: Boni García, Mario Munoz-Organero, Patricia Callejo and Miguel Ángel Hombrados

Received: 21 March 2024
Revised: 9 June 2024
Accepted: 10 June 2024
Published: 12 June 2024

1. Introduction

A growing and widespread interest in the research community has arisen due to the emergence of smart contracts (SCs). A trusted environment for SCs has been created by decentralized blockchain technology in recent years without the need for third-party intervention [1]. A smart contract is a digital agreement that operates on a blockchain network and is automatically executed once specific terms and conditions are met. In other words, SCs work similarly to If-Then statements in different computer programs, and in this way, real-world assets may be affected by changes in the smart contract [2,3]. Compared with traditional contracts, they have the following significant merits: (i) reducing transaction risks; (ii) reducing administrative and service costs; and (iii) improving the efficiency of business processes [4].

Despite the unique benefits provided by SC technology, it is in the early stages of development. Moreover, there are still many problems that need to be addressed [4]. The exploitation of vulnerabilities in SCs to target the contracts themselves has witnessed a moderate increase recently. In this context, the core question is: What is the reason for SCs being targeted by hackers? Liao et al. [1] answered the question as follows: (i) SCs are valuable but vulnerable; (ii) SCs are challenging to patch due to the immutability nature of the blockchain; and (iii) there is a lack of assessment standards for ensuring.

The classification of traditional vulnerability detection tools by researchers involves five core methods, namely, static analysis, symbolic execution, dynamic analysis, formal verification, and fuzzy testing [5–7]. It should be noted that symbolic execution and formal verification are typically considered part of static analysis, whereas fuzzy testing falls under dynamic analysis. However, conventional strategies heavily lean on vulnerability rules that are manually created, a method that is time-consuming and challenging to address in all possible cases [3,5,7–9]. Moreover, it is still a challenge to discover unknown vulnerabilities [10].

Machine learning models can provide the possibility of finding vulnerabilities in substantially less time while retaining reasonable accuracy. Moreover, adopting more analyzers in the machine learning model can significantly increase the probability of finding the intersected vulnerabilities. Furthermore, applying a machine learning model to predict the security vulnerabilities in a SC can lead to a substantial enhancement in efficiency when compared with conventional vulnerability detection tools [11,12].

To date, many systematic reviews have assessed traditional vulnerability detection tools, applications of SCs, and future research opportunities [1,2,13,14]. To the best of our knowledge, we identify a research gap: a lack of a systematic review that covers machine learning-based solutions to cope with vulnerability detection problems in SCs. Our primary focus was developing a systematic literature review (SLR) to bridge this gap. Moreover, we observed that the class imbalance issue has been neglected by a significant majority of machine learning-driven solutions. The detection of unknown vulnerabilities in SCs has also been ignored in many proposed machine-learning-based frameworks. Motivated by these limitations and to resolve them, we arranged an SLR about vulnerability detection tools and machine learning-driven solutions.

In summary, the novel and significant contributions of this paper are enumerated as follows:

- The focus is on state-of-the-art machine learning-driven solutions that have been applied for vulnerability detection in SCs. In this respect, 55 journal articles and conference papers published between 2019 and 18 May 2024, will be explored. Moreover, we classify these methods under three different machine learning perspectives (i.e., classical models, deep learning, and ensemble models).
- We underline cutting-edge machine learning-driven frameworks that address the class imbalance issue and unknown vulnerabilities.
- We propose some contemporary open challenges to cover significant gaps when machine learning methods are applied for vulnerability detection.

The remainder of this paper is organized as follows: Preliminaries are introduced in Section 2. Section 3 briefly describes related work. In Section 4, the research methodology covers our paper selection procedure. The classification of the selected papers is represented in Section 5. Section 6 provides an analytical comparison. Sections 7 and 8 ultimately lead to discussions and conclusions.

2. Preliminaries

This section introduces terminology, such as blockchain, Ethereum, SCs, vulnerability detection tools, machine learning techniques, and the class imbalance necessary for the paper.

2.1. Blockchain

Blockchain is a chain structure formed by the orderly concatenation of data blocks according to the generation time. The development of blockchain technology can be divided into three stages. It has passed the 1.0 stage and is in the process of moving to blockchain 2.0, that is, the smart contract stage [4,12,15,16].

2.2. Ethereum

In essence, Ethereum is a distributed (also decentralized) ledger that uses blockchain as its basic support technology, like Bitcoin. It supports the execution and invocation environments of SCs through a Turing-complete machine that is called the Ethereum Virtual Machine [5,7].

2.3. Smart Contracts

SCs were first proposed by Szabo (1997) [17], emphasizing that they facilitate the execution of contracts using protocols and user interfaces. When predetermined conditions are satisfied, it is triggered to execute and update the blockchain. Once a contract related to the transaction is executed, the transaction result cannot be changed or reversed. It includes a set of executable functions and state variables. There are deterministic and non-deterministic SCs. A deterministic SC does not need any information from an external party (outside the blockchain). A non-deterministic SC depends on oracles or data flows from an external party. SC technology is based on three things: the platform, the programming language, and the execution environment [4,18].

2.4. Vulnerability Detection Tools

SCs written in a high-level language such as Solidity are compiled into bytecodes and executed by the transaction drivers. Therefore, the security threats of SCs mainly come from potential vulnerabilities at the three levels of virtual machines, high-level languages, and blockchains. Table 1 summarizes the most discussed Ethereum SC vulnerabilities [6,14,19,20].

There are five distinct categories of vulnerability detection tools: (1) static analysis, (2) symbolic analysis, (3) dynamic analysis, (4) formal verification methods, and (5) fuzzy testing [7,8,18]. Table 2 presents a brief explanation of these analysis tools and their well-known samples.

2.5. Machine Learning Techniques

Machine learning is a subset of artificial intelligence and enables computers to learn from data and make decisions or predictions without being explicitly programmed to do so [11].

2.5.1. Classical Machine Learning Models

Classical machine learning techniques are classified into four types based on the nature of the learning regime and the data available, including (1) supervised, (2) semi-supervised, (3) unsupervised, and (4) reinforcement learning. Table 3 briefly describes common machine learning algorithms [11].

2.5.2. Deep Learning Models

Deep learning, a specialized subset of machine learning, is inspired by the information processing patterns found in the human brain and distinguished by its use of neural networks with three or more layers. Three types of deep neural networks include: (i) multi-layer perceptron (MLP), (ii) convolutional neural networks (CNN), and (iii) recurrent neural networks (RNN) [21,22].

2.5.3. Ensemble Learning

The concept driving this idea is that the combined intellectual capacity of two minds surpasses that of a single mind. Ensemble learning's basic principle is to train multiple base learners as ensemble members and integrate their predictions into a single output that should have superior performance on average than any other ensemble member with uncorrelated error on the target datasets. Ensemble methods can improve resilience to noise. The most popular advanced ensemble methods are boosting, bagging, and stacking [23].

Table 1. Twelve types of security vulnerabilities in SCs.

Vulnerability	Description
Timestamp Dependency	When a contract employs block variables as a call condition to execute important operations or as a seed for generating random numbers [24,25].
Reentrancy	It allows attackers to invoke a specific contract function multiple times while the contract execution remains incomplete, enabling them to manipulate the contract's behavior or pilfer funds [1,24–26].
Transaction Ordering Dependency	The attacker can manipulate the order in which transactions occur, thus potentially exploiting the logic of a contract [24,25].
tx.origin	A transaction origin attack is a phishing attack that can drain a contract of all funds [1,25,26].
Block-hashBlock Number	When the block has a block number, it is utilized to generate randomness by generating random numbers [24,25].
Gas Related Issues	The dependency of contract codes on data items that are either provided or manipulated by contract users is what causes these vulnerabilities [24,25].
Delegate Call	Using Ethereum virtual machine opcodes maliciously by the callee contract to bring up to date the state variables of the caller's contract [24,26].
Arithmetic UnderflowOverflow	When the result of a mathematical operation exceeds the maximum value that the program can store, an arithmetic underflow is the opposite of an arithmetic overflow [1,24,26].
Unchecked Call	When an exception is thrown by the called contract and the return value is not validated, there is a risk of the execution continuing, which could result in unexpected behavior within the contract [1].
Self Destruct	The attacker can use the "self destruct" method to kill a contract because of poor authentication in the contract [1,24,26].
Access Control	Inadequate authorization or authentication enables the attacker to maliciously access the critical functions [1].
Denial of Service	Acting in a malicious manner while receiving a transaction. Enhancing the gas artificially to calculate a function. Exploiting access controls to gain unauthorized entry into private components of SCs [1].

Table 2. Classification of vulnerability detection (VD) tools.

VD Tools	Description	Samples
Static	These tools use a compliance pattern to check the vulnerability of a contract [4,13].	Smartcheck, Securify, and Slither
Symbolic	These tools use symbolic execution to check for the transformation of the memory state [4,13].	Oyenty and Mythril
Dynamic	By employing a dynamic tool, the contract's behavior is continuously monitored during execution to identify and mitigate any potential vulnerabilities or security breaches proactively [4,13].	Maian and ECFChecker
Formal Verification	These methods produce proofs based on an abstract "mathematical model" of a system [4,13].	F* Framework, Isabelle/HOL, and FEther
Fuzzy Testing	These methods run the SC on the blockchain and fuzz some input to the contract [4,13].	ContractFuzzer, Echidna, and ILF

Table 3. Classical machine learning algorithms.

Learning Algorithm	Description
Supervised	It is a type of labeling learning technology that uses a labeled training dataset to construct a model representing the learned relationship between the input and output values.
Semi-supervised	It is a type of learning technology in which most of the training samples are unlabeled and a few are labeled.
Unsupervised	It is a machine learning algorithm that uses training datasets without labels.
Reinforcement	Training algorithms using a system of reward and punishment mechanisms.

2.6. Class Imbalance

Classification is one of the core tasks in the field of machine learning. The class imbalance issue can make learning difficult since learning tends to be biased towards the majority class. That is to say, learners will overclassify the majority group when there is a class imbalance in training data due to its increased prior probability. As a result, instances belonging to the minority group are misclassified more often than those belonging to the majority group. Current solutions for class imbalance can be classified into three categories: data-level methods, algorithm-level methods, and hybrid approaches [27].

2.6.1. Data-Level Methods

Data-level methods for addressing class imbalances include over-sampling and under-sampling. These techniques modify the training distributions to decrease the level of imbalance or reduce noise, e.g., mislabeled samples or anomalies. In their simplest forms, random under-sampling discards random samples from the majority group, while random over-sampling duplicates random samples from the minority group [27].

2.6.2. Algorithm-Level Methods

Unlike data sampling methods, algorithmic methods for handling class imbalances do not alter the training data distribution. Instead, the learning or decision-making process is adjusted in a way that increases the importance of the positive class. Most commonly, algorithms are modified to take a class penalty or weight into consideration, or the decision threshold is shifted in a way that reduces bias towards the negative class [27].

2.6.3. Hybrid Approaches

Data-level and algorithm-level methods have been combined and applied to class imbalance problems. One strategy includes performing data sampling to reduce class noise and imbalance and then applying cost-sensitive learning or thresholding to further reduce the bias towards the majority group [27].

3. Related Work

This section reviews 14 surveys and SLRs published from 2019 to 2023 in blockchain and SCs, then discusses the advantages and limitations of each study. Table 4 details review papers related to SCs. Table 5 describes the method of their article selection. Table 6 refers to research questions, followed by recently published survey papers.

Table 4. Overview of related reviews and their coverage based on SCs.

Reference	Environment	Review Type	Year of Pub	Selection Process	Taxonomy	Year Covered
[28]	Security, performance, and applications	SLR	2019	clear [1]	✓	2016–2018
[29]	Security	survey	2019	clear [1]	✓	Until Deccember 2018
[30]	Applications	SLR	2019	clear [1]	✓	2014–April 2018
[31]	Security	SLR	2020	semi clear [2]	✓	2015–July 2019
[32]	Cyber security	SLR	2020	clear [1]		April 2018
[33]	Development, test, and security	SLR	2021	clear [1]	✓	2016–2020
[34]	Technical aspect, future directions	survey	2021	not clear [3]		Until 2020
[4]	Difficulties and future trends	review	2022	not clear [3]		Until 2022
[13]	Analysis tools	SLR	2022	semi clear [4]	✓	2016–December 2021
[35]	Vulnerability detection and machine learning	survey	2022	not clear [3]		Until 2021
[36]	Vulnerability detection tools within Web 3.0	survey	2023	not clear [3]	✓	2011–2023
[14]	Security threats, collaborative defense	review	2023	not clear [3]		2018–2023
[37]	Vulnerability detection	survey	2023	not clear [3]		Until 2023
[38]	Platforms, applications, and challenges	review	2023	clear [1]	✓	2015–May 2020

[1] The identification, exclusion, eligibility, and inclusion methodology are clear. [2] The number of selected papers is not clear. [3] The identification, exclusion, eligibility, and inclusion methodology are not clear. [4] Keywords are not mentioned in the methodology.

A systematic survey [28] reviews the security, performance, and applications of SCs. The method of selecting articles is explicitly mentioned in this study. However, the search domain for the reviewed articles is not clear. The scholars introduced a taxonomy that categorizes recent studies and developments regarding SCs based on three main categories: (i) security methods and tools, (ii) performance improvement approaches, and (iii) decentralized applications.

In another study, a survey [29] reviews the security verification of blockchain SCs. The method of article selection has not been well demonstrated in this research. Their article search is limited to December 2018. They proposed a taxonomy for the topic of security verification of blockchain SCs. The security assurance aspects are classified into three categories, including environment security, vulnerability scanning, and performance impacts. In contrast, the correctness verification aspect is classified into two categories, including programming correctness and formal verification.

Table 5. The method of article selection in review papers.

Reference	Database								Keywords	# Papers
	Google Scholar	IEEE	ScienceDirect	Springer	Wiley	Taylor & Francis	ACM	Other		
[28]		✓	✓	✓	✓		✓	✓	Smart contract, contract, chaincode	90
[29]		✓	✓	✓			✓	✓	Smart contract, security, verification	53
[30]								✓ [1]	Blockchain, application	260
[31]	✓								Smart contract, Dapps, token, EVM, security	-
[32]	✓	✓	✓	✓			✓		Blockchain, block-chain, distributed ledger, security	30
[33]	✓	✓	✓				✓		Blockchain, smart contract, Ethereum	96
[13]		✓	✓	✓	✓				-	132
[38]		✓	✓	✓			✓		Smart contract, smart contract applications, smart contract tools, smart contract platforms, smart contract challenges, blockchain	183

[1] Scopus database.

Table 6. Research questions followed by survey papers.

Reference	Research Questions
[28]	1: What are the security problems with SCs, and how can we address them? 2: What methods are applied to improve the performance of SCs?
[31]	1: How is the existing security state-of-play for SCs? 2: How do you design countermeasures to the identified security challenges? 3: When and where are these studies about SC security published? 4: What are potential future research challenges?
[32]	1: What are the latest blockchain applications focused on security? 2: How is blockchain used to improve cyber security? 3: What methods are available for blockchain solutions to manage security without requiring a cryptocurrency token?
[13]	1: What are the static analysis tools available for Ethereum blockchain SCs? 2: Which dynamic analysis tools are available for the Ethereum blockchain SC? 3: For Ethereum blockchain SCs, what kind of analysis approaches are employed by static/dynamic analysis tools? 4: What are the five most common vulnerabilities detected by analysis tools?
[38]	1: How can available platforms support SC development? 2: What are the presented decentralized applications, which mainly discuss SC design and implementation? 3: What are the challenges that relate to the application of SCs? 4: What are the solutions to the identified challenges?

Likewise, a systematic literature review [30] reviews blockchain-based applications, their current status, classification, and open issues. The method of article selection in their study has not been clearly indicated. Moreover, the search scope of their article is not clear. They presented a comprehensive classification of blockchain-enabled applications across diverse sectors such as supply chain, business, healthcare, IoT, privacy, and data management.

Furthermore, a survey [31] explores Ethereum SC security and future research. The method of article selection in their study has not been clearly revealed. Their article search domain spans from 2015 to 2019. They focused on how SCs can be maliciously exploited and targeted. Specifically, they concentrated on abnormal contracts, program vulnerabilities, and unsafe external data issues in SCs.

Another systematic literature review [32] is dedicated to utilizing blockchain for cyber security. The researchers demonstrated their article selection method. The scope of their article search is limited to April 2018. Their study presented a systematic analysis of the most frequently adopted blockchain security applications. Moreover, they highlighted opportunities for future research via three potential research agendas. However, the scholars do not introduce a taxonomy regarding cyber security.

In another study, a survey [33] reviews SC development, techniques, tools, and open challenges. The researchers effectively indicated their article selection method. The scope of their article searches spans from 2016 to 2020. They reviewed papers related to six specific topics: SC testing, SC code analysis, SC metrics, SC security, decentralized application (DApp) performance, and blockchain applications. Moreover, this paper identifies open challenges for each of the six topics above.

The survey [34] reviews the technical aspects and future research directions of blockchain-based SCs. The method of article selection is not described in their research. Their article search domain is not clear. They reviewed security attacks, vulnerabilities, and possible solutions. Moreover, they elaborated on future research topics, including SCs and game theory, SCs and artificial intelligence, and SCs in data science.

The review [4] summarizes the recent progress of SCs in blockchain. The method of article selection is not described in their research. Their article search domain is not clear. They investigated the difficulties faced by SCs and the corresponding solutions. Moreover, this review examines and judges the future challenges and development trends of SCs.

The systematic review [13] reviews the analysis tools associated with the Ethereum blockchain SC. They described their method of article selection. The scope of their article search covers the years from 2016 to December 2021. They investigated a detailed review of 86 analysis tools, covering all the analysis tools present in the literature or on the web, irrespective of their type and analysis approach.

The survey [35] reveals vulnerability detection using machine learning. The method of selecting articles is not explicitly mentioned in this study. Moreover, the search domain for the reviewed articles is not clear. They provided insights on the limitations and advancements of machine learning-driven solutions.

In another study, a survey [36] reviews the approaches for vulnerability detection in SCs within Web 3.0 applications. The method of article selection has not been demonstrated in this research. Moreover, their article search is not clear. They proposed a taxonomy of vulnerability detection approaches into four categories: formal verification, symbolic execution, fuzzing, and taint analysis.

The literature review [14] investigates the research progress on blockchain security threats and collaborative defense. The method of article selection in their study has not been indicated. Moreover, the search scope of their article is not clear. They discussed the challenges of blockchain security and future research directions, such as parallel detection and federated learning.

Furthermore, the survey [37] explores vulnerability detection and security enhancement. The method of article selection in their study has not been clearly revealed. Their article search domain is not clear. They provided a taxonomy of SC vulnerability analysis relevant to security.

Finally, the review [38] focuses on SC-based platforms, applications, and challenges. The researchers indicated their article selection method. The scope of their article search is not clear. They provided a table that shows the distribution of academic papers according to category. Moreover, they categorized large SC implementations by domain, including insurance, supply chain management, the Internet of Things, healthcare, multimedia, cloud computing, identity management, and record management.

Table 5 presents the databases, keywords, and the number of articles associated with the references mentioned in Table 4. The selection process for these references is either "clear" or "semi-clear".

In summary, reviewing previous articles, the following weaknesses were found: (1) The survey articles did not include the newly proposed machine learning-drive solutions for vulnerability detection in SCs. (2) The problem of class imbalance and unknown vulnerabilities has been ignored by many research papers that rely on machine learning-driven solutions. Considering the cases mentioned above, this SLR is presented to solve the problems of previous research.

4. Methodology

Our systematic literature review on Ethereum smart contract vulnerability detection has been carried out following the guidelines of Moher et al. [39]. In this section, we will introduce how the examined papers have been selected, prepared for analysis, and an overview of the produced research (how the papers are distributed over time and publication types).

4.1. Research Questions (RQ)

By studying review articles and SLRs, challenges in different machine learning-driven vulnerability detection tools in the SC area remain; thus, several questions have been raised regarding machine learning-based approaches to vulnerability detection. The existing studies are entirely valid and relevant to the field in question. Therefore, three questions were prepared, which will be answered analytically and graphically in the following sections.

- RQ1: What machine learning-driven techniques are used in vulnerability detection tools?
- RQ2: What machine learning-driven vulnerability detection tools have addressed the class imbalance issue?
- RQ3: What machine learning-driven vulnerability detection tools have addressed unknown vulnerabilities?

4.2. Selection of Primary Studies

On 23 February 2024 the search was conducted, and it was later updated on May 18, 2024. The platforms used for the search were IEEE, ScienceDirect, Springer, Wiley, Taylor & Francis, and ACM. The search covers title, abstract, and keywords. To search for primary studies, we have identified the following keywords: "smart contract" AND "vulnerability" AND ("machine learning" OR "deep learning" OR "ensemble learning") AND "vulnerability detection".

For each paper found, the title was analyzed, followed by an abstract, an introduction, and conclusions to determine whether it pertained to the established inclusion criteria. Then we analyze the rest of each document to find significant contributions and open issues.

4.3. Inclusion and Exclusion Criteria

This SLR targets journal articles, international conference proceedings, and symposiums published between 2019 and 18 May 2024. Papers on topics not belonging to the Ethereum smart contract vulnerability detection and machine learning-driven solutions have been excluded. Duplicated articles and all other kinds of work like books, surveys, review articles, technical reports, or master theses were omitted. Moreover, any articles not written in English were excluded from the study.

4.4. Selection Results

Considering the keywords, the following results have been obtained, divided by platform:

- IEEE: 57 results.
- ScienceDirect: 12 results.

- Springer: 29 results.
- ACM: 7 results.
- Wiley: 3 results.
- Taylor & Francis: 3 results.

Subsequently, the duplicated studies were removed. Then the papers concerning the exclusion criteria were removed. Finally, 55 papers remained on which analysis was conducted. All the phases of the methodology for the inclusion and exclusion of research articles are depicted in Figure 1.

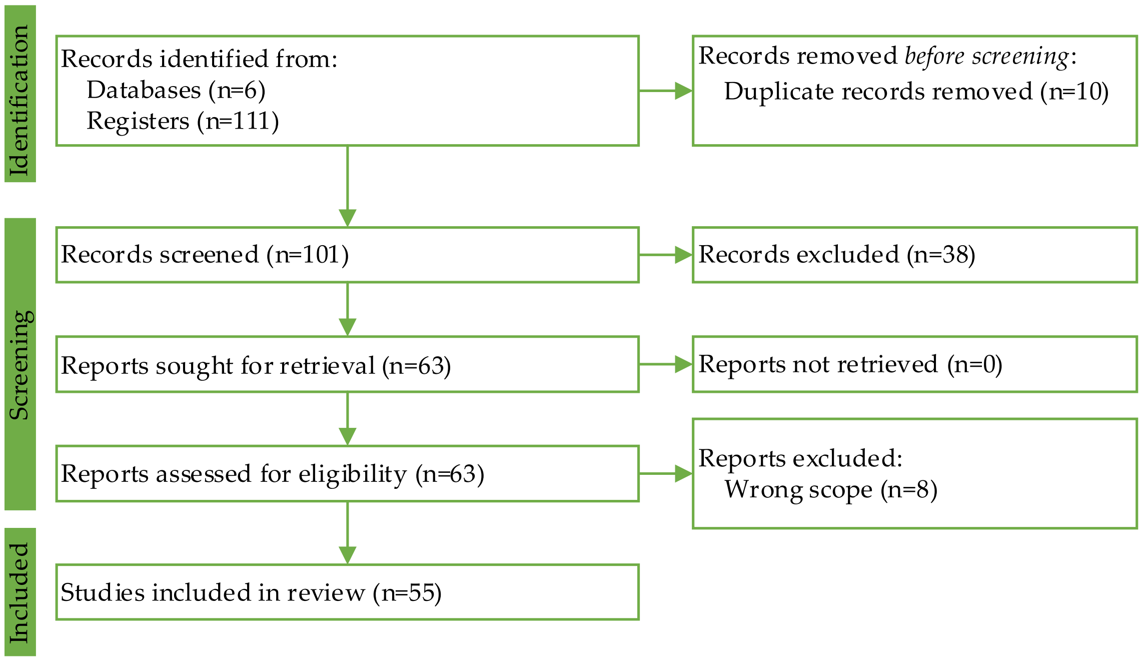
Identification
Records identified from:
Databases (n=6)
Registers (n=111)
Records removed *before screening*:
Duplicate records removed (n=10)
Screening
Records screened (n=101)
Records excluded (n=38)
Reports sought for retrieval (n=63)
Reports not retrieved (n=0)
Reports assessed for eligibility (n=63)
Reports excluded:
Wrong scope (n=8)
Included
Studies included in review (n=55)

Figure 1. Literature selection process based on PRISMA flowchart on three levels.

In addition, we plot the number of papers published per year to reveal temporal trends (see Figure 2).

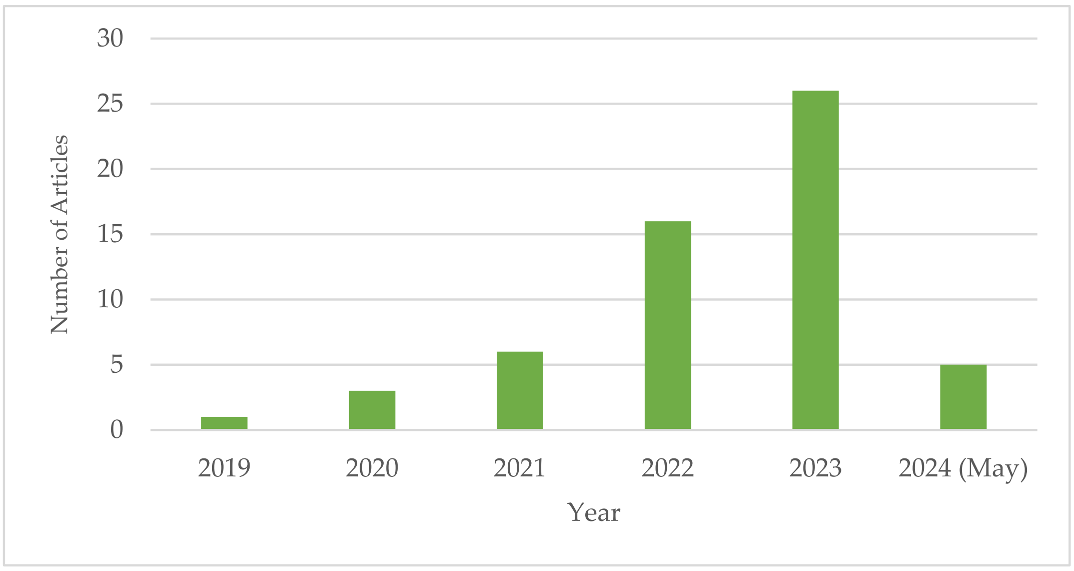

Figure 2. The number of covered papers published between 2019 and May 2024.

5. Taxonomy of Machine Learning-Driven Solutions for Vulnerability Detection

In this section, cutting-edge algorithms are classified under three different machine learning perspectives (i.e., classical, deep learning, and ensemble models). Furthermore, this SLR explores state-of-the-art machine learning-driven solutions that address the class imbalance issue and unknown vulnerabilities. In addition, we investigate the development of contemporary solutions in line with standardized quality metrics.

5.1. Machine Learning Models

5.1.1. Classical Machine Learning Models

Momeni et al. [26] and Li et al. [40] propose classical machine learning frameworks to detect security vulnerabilities of SCs on the Ethereum platform. The K-Nearest Neighbor and Decision Tree are common classifiers among their models. Momeni et al. [26] utilize Random Forest and Support Vector Machine, while Li et al. [40] employ Linear Regression and Stochastic Gradient Descent in their method. The model developed by Momeni et al. [26] not only identifies security vulnerabilities but can also be applied to other languages and platforms. However, Li et al.'s [40] model detects unknown vulnerabilities based on the similarity principle. In contrast to the model proposed by Momeni et al. [26] that relies on manual feature extraction, the Eth2Vec [41] scheme utilizes a neural network to automatically learn features of vulnerable Ethereum Virtual Machine (EVM) bytecodes.

Xu et al. [42] and Yang et al. [43] propose vulnerability detection techniques for SCs, which extract SC features based on the abstract syntax tree (AST) and train the models to detect smart contract vulnerabilities. Xu et al. [42] use K-Nearest Neighbor and Stochastic Gradient Descent classifiers, while Yang et al.'s [43] model is developed based on Random Forest classifier. Xing et al. [21] propose Random Forest based on opcode features (RFBOOF) model. This model was developed based on classical and deep learning algorithms. In comparison with classical machine learning-based approaches, the method proposed by Eshghie et al. [44] (Dynamit) eliminates the need for any domain knowledge, code instrumentation, or exceptional execution environment.

5.1.2. Deep Learning Models

The methods proposed by Tang et al. [45], Jain and Tripathi [46], Shen and Li [47], Demir et al. [48], Han et al. [49], Zhu et al. [50], and Chen et al. [51] utilize a combination of machine learning algorithms to detect SC vulnerabilities. The Lightning Cat method [45] trains three deep learning models: Optimized-CodeBERT, Optimized-LSTM, and Optimized-CNN. This framework extracts problem code segments functions, which not only considers the key features of SC vulnerability code but also solves the length limitation problem of deep learning for processing long texts. The method proposed by Jain and Tripathi [46] focuses on a feature extraction mechanism based on a two-step hierarchical model using deep learning techniques: Bidirectional Gated Recurrent Unit (Bi-GRU) and Text-CNN. The framework proposed by Shen and Li [47] follows a parallel feature extraction strategy based on CNN-LSTM and RNN-GRU. Demir et al. [48] focus on detecting Reentrancy vulnerabilities using a Bidirectional LSTM (Bi-LSTM) model by employing assembly data to detect flaws. Han et al. [49] propose a model based on GNN and CNN to obtain the semantic and structural information of the source code, unlike methods that mostly view SC source code as natural language for processing. The GraBit method [50] is developed based on GraphCodeBERT and Bidirectional LSTM, which embeds both the source code and concise key data flow graphs extracted from the code. The SCVSN paradigm [51] utilizes the Siamese network and LSTM to complete the task of vulnerability detection. The Siamese network consists of two subnetworks that share the same parameters in a low-dimensional and easily separable feature space. The SCVSN scheme can identify the presence of vulnerabilities within a SC but cannot detect what kinds of vulnerabilities there are.

The models proposed by Zhang et al. [52], Hwang et al. [53], Zhou et al. [54], Mittal et al. [55], Feng et al. [56], Liang and Zhai [57], Zhou et al. [58], and Zeng et al. [59]

are developed based on convolutional neural networks (CNNs). The Multi-Objective Detection Neural Network (MODNN) method [52] can validate twelve types of vulnerabilities, including ten recognized threats and more unknown types, without the need for predefined knowledge. CodeNet [53] includes two main steps: (1) the SC source code is compiled into bytecodes; bytecodes are transformed into an SC-based input image for CNN architectures while keeping the semantics and context of an SC; and (2) it analyzes the SC-based input images to detect vulnerable SCs. The Tree-based Machine Learning Vulnerability Detection (TMLVD) method [54] feeds the intermediate representations of SCs derived from AST into a tree-based CNN for building the prediction model. Like TMLVD, the technique introduced by Mittal et al. [55] and the SmartEmbet-TextCNN-TMP (SEET) framework [56] vectorize the Solidity code into AST, while SCGRU [57] transforms the standardized data into word vector representations of SCs with semantic information through the Word2Vec word embedding module. The CS-VDM scheme [58] automatically detects the vulnerabilities in the SC without expert knowledge and in very little time. Like the CS-VDM method, the SolGPT technique [59] focuses on runtime. Moreover, it boosts feature extraction capabilities while reducing reliance on labeled data. Nonetheless, this strategy may have constraints when it comes to the diversity of types and samples.

Graph Neural Networks (GNNs) are the central architecture of vulnerability detection models proposed by Cai et al. [60], Chen et al. [61], Liu et al. [62], Liu et al. [63], Liu et al. [29], Zhen et al. [64], Nguyen et al. [65], Xiong et al. [66], Cai et al. [60], Wang et al. [67], Wang et al. [68], Lin et al. [69], Wei et al. [70], and Wu et al. [71]. The method proposed by Cai et al. [60] constructs a graph representation for a SC function with syntactic and semantic features by combining AST, control flow graph (CFG), and program dependency graph (PDG). In contrast, Chen et al. [61] construct a novel semantic graph (SG) for each function and learn the SGs using graph convolutional networks (GCNs) with residual blocks and edge attention. The Multi-Relational Nested Contract Graph (MRNG) scheme [62] characterizes the rich syntactic and semantic information in the SC code, including the relationships between data and instructions while Combining the graph feature and the expert patterns (CGE) method [63] is a fully automated approach for SC vulnerability detection at the function level. The paradigm followed by Liu et al. [72] blocks risky transactions during the operation phase. In this method, the code violates the rules, the interrupt module is triggered, the code transaction is terminated, and an error report is issued. The Dual Attention Graph Neural Network (DA-GNN) scheme [64] combines attention mechanisms and GNN. This method starts by generating attributes of the SC from its bytecode, including the control flow graph and opcode sequence. It extracts attribute features from both parts and combines them as input to the dual attention module to obtain graph features for identifying vulnerabilities in SCs.

The MANDO-HGT framework [65] adapts heterogeneous graph transformers (HGTs) with customized meta relations for graph nodes and edges to learn their embeddings and train classifiers for detecting various vulnerabilities. In contrast, the FBB-VD scheme [66] uses a graph neural network to combine multiple code representations and detect vulnerabilities in SCs. The method proposed by Wang et al. [67] chooses the appropriate network depth and does not blindly increase the number of hidden layers. The GVD-net method [68] uses a distinct strategy in the preprocessing phase based on a weight matrix and generates the corresponding relationships of nodes based on a non-Euclidean graph. The Smart Contract Vulnerabilities by Fusing Semantic Features with Expert Features (SmartFuSE) model [69] conducts static analysis to extract vulnerability-specific expert patterns and joint graph structures at the function level. This model allows for the representation of the comprehensive program semantics of vulnerable code. Additionally, it utilizes a unique graph neural network with a hybrid attention pooling layer to emphasize critical vulnerability features. In contrast to other graph-based approaches, the technique suggested by Wei et al. [70] capitalizes on structural information and artificial rules. The Self-Attention Graph Pooling SAGDP framework [71] focuses more on essential features than unimportant features to capture the graph's structural and feature information jointly.

LSTM is another deep learning method used by Ren et al. [73], Colin et al. [25], Xu et al. [74], Zhang et al. [52], Qian et al. [75], Zhou et al. [76], and Hu et al. [77] to detect vulnerabilities in SCs. The Blass framework [73] constructs program slices with complete semantic structure information (CPSs). It uses AST and a depth-first traversal algorithm to convert CPSs into code chains during CPS vectorization. The paradigm proposed by Colin et al. [25] not only standardizes the pre-processing method for SC training data but also introduces bugs to create a balanced dataset of flawed files across Solidity versions using AST. The HAM-BiLSTM model [74] uses a hierarchical attention mechanism. This framework takes the code segment and account information of an SC as input and divides the input samples into three levels of documents: word level, sentence level, and document level, whereas Zhang et al.'s method [52] first vectorizes the SC code, then inputs the vectorized data into the LSTM network to generate a model, and finally uses the model to detect defects. Wang et al. [78] propose a different vulnerability detection method based on LSTM that uses a code representation fusion strategy. The bidirectional LSTM with attention mechanism (BLSTM-ATT) method was developed by Qian et al. [75] as a distinct technique to detect Reentrancy vulnerabilities. This work stands out due to the implementation of a representation method that contributes to capturing essential semantic information and control flow dependencies. Although most of the methods reviewed in this paper are focused on the detection of Reentrancy vulnerabilities, the HuntFlow strategy [76] detects arithmetic vulnerabilities such as Integer Overflow vulnerabilities. The framework proposed by [77] stands out from other LSTM-based strategies due to its incorporation of two parsing methods: a combination of opcode sequences and AST.

The methods proposed by Vu et al. [79] and Zeng et al. [80] are focused on Bidirectional Encoder Representations from Transformers (BERT). SecBERT [79] is a pre-trained model to extract the implicit features and analyze them using the Multi-Layer Perceptron algorithm, whereas SCVulBERT [80] utilizes SCVulTokenizer for code tokenization and rich prior knowledge for training.

Furthermore, researchers also paid attention to other deep learning models that were found less frequently in the analyzed works. Narayana and Sathiyamurthy [81] develop a model based on Autoencoders. The Link-DC model [82] uses expert knowledge as the original data input to capture richer feature information. The Clear method [83] employs a contrastive learning (CL) model to capture the fine-grained correlation information among contracts. The S-BiGRU scheme [84] combines a Bidirectional Gated Recurrent Unit and SMOTE for smart contract vulnerability detection. This paradigm processes and parses Solidity source code into a series of tokens embedded into feature vectors using Word2Vec. These word vectors are fed into the model for training to obtain a model capable of detecting smart contract vulnerabilities for subsequent detection work. The method proposed by Qin et al. [85] focuses on the problem of poor detection caused by excessive noise. Other researchers have primarily overlooked this issue. The bytecode for smart contract vulnerability detection is given special attention in the cross-modality framework proposed by Qian et al. [86]. This scheme is developed based on a large-scale labeled benchmark dataset that consists of four different types of vulnerabilities.

5.1.3. Ensemble Learning Models

Wang et al. [24], JJ et al. [87], Ma et al. [88], and Huang et al. [89] develop ensemble learning models to detect vulnerabilities in SCs. ContractWard [24] is a model for effectively and efficiently detecting six types of vulnerabilities based on extracted static characteristics. It can be applied to detect vulnerabilities in SCs written in all high-level languages, such as Solidity, Serpent, and LLL. JJ et al.'s [87] model includes Bagging, AdaBoost, and Gradient Boost classifiers, while the HGAT scheme [88] uses functions based on AST and CFG. The CDRF [89] is a time-saving vulnerability detection method that processes the opcode fragments by word2vec and PCA to obtain one-dimensional binary features.

5.2. Class Imbalance Issue

This subsection reviews machine learning-driven vulnerability detection tools that have considered the class imbalance issue in their framework.

The Synthetic Minority Oversampling Technique (SMOTE) is applied by [24,46,84,87]. SMOTE is an oversampling technique, interpolating between the minority classes to generate extra ones [24]. ContractWard, proposed by Wang et al. [24], follows two sampling algorithms, namely, SMOTE and SMOTETomek, to balance the training datasets. That is to say, these methods are adopted to extend the number of minority classes to be similar to that of the majority classes. Hence, it can eliminate unnecessary samples while conducting the sampling process. Jain and Thripathi [46] are convinced that unless the level of imbalance is substantial, it will not have a significant impact on the performance of the model. In this context, they calculate the class imbalance ratio. Then their approach aligns with the policy outlined by Wang et al. [24], except for the substitution of SMOTETomek with TomeLinks. Finally, Jain and Thripathi [46] verify that the balanced dataset's ratio of vulnerable to non-vulnerable contracts is approximately 1:1. In contrast to the approaches suggested in [24,46], Song et al. [84] and JJ et al. [87] exclusively employ the SMOTE model to avoid any biasing effect and hence possess a uniform sample distribution. It concentrates on the feature space to produce new examples by interpolating between positive samples that are close together.

Zhang et al. [52] and Cai et al. [60] adopt entirely distinct approaches from previous methodologies to address the class imbalance issue. Zhang et al. [52] utilize the focal loss algorithm to deal with the class imbalance problem. Their scheme (MODNN) uses k-fold cross-validation (KCV) to validate each randomly generated subset. Each sample of data for KCV can be used for testing and training to avoid overfitting and underfitting and to obtain more credible results. When the loss of MODNN is within the acceptable range, the stop condition control parameter "early_stop" value is reset to 1. Cai et al. [60] believe that a sample imbalance may threaten the validity of their approach. With the size of the dataset increasing, more factors need to be controlled in the experiment. In this context, they randomly shuffle the dataset in the same proportion and utilize a validation set to test the model's generalization performance to alleviate this potential threat.

5.3. Unknown Vulnerabilities

The present machine learning-based vulnerability detection techniques mainly focus on known vulnerabilities, while unknown vulnerabilities in SCs have been rarely investigated. This subsection delves into examining recently published articles [40,52,65,71] that have tackled this particular issue.

Li et al. [40] propose a classical machine learning-based method for detecting unknown vulnerabilities in SCs using opcode sequences. It combines an n-gram model and a vector weight penalty mechanism to enhance the detection of unknown vulnerabilities. They consider unknown vulnerabilities to be those that already exist but cannot be discovered using existing vulnerability detection methods. Moreover, researchers emphasize that such vulnerabilities are objectively present, even if they have not been found. Their scheme analyzes the similarity of the behavior of unknown and known vulnerabilities from two aspects: (1) vulnerability and (2) attack.

The methods proposed by Zhang et al. [52], Nguyen et al. [65], and Zou et al. [71] utilize deep learning models to detect unknown vulnerabilities in SCs. MODNN [52] uses a CNN-based model to identify more unknown vulnerabilities without needing a specialist or predefined knowledge. This method learns vulnerabilities in unknown patterns from explicit features. By contrast, Nguyen et al. [65] and Zou et al. [71] use a GNN-driven strategy to deal with unknown vulnerabilities. The framework proposed by Nguyen et al. [65] can be re-trained for new types of vulnerabilities and detect vulnerabilities in large sets of contracts efficiently and accurately. In contrast, the SAGP approach [71] is equipped with the self-attention mechanism to detect vulnerabilities that may not be identifiable by expert knowledge, thereby minimizing the possibility of errors that can arise from manually established detection patterns. This method is implemented using a hierarchical pooling structure.

6. Comparison

An analytical discussion and comparison between the final articles are provided in Section 5. Section 4.1 deals with research questions. The present section answers the questions.

6.1. RQ1: What Machine Learning-Driven Techniques Are Used in Vulnerability Detection Tools?

Table 7 illustrates a taxonomy of vulnerability detection tools based on machine learning models to develop methods for detecting vulnerabilities in SCs. Researchers in [21,25] use a hybrid method to build their framework. Moreover, supervised learning is the most found learning method among classical models, while unsupervised learning is neglected. Table 8 depicts the most found classifiers for detecting vulnerabilities in SCs using machine learning-driven solutions.

Table 7. A taxonomy of vulnerability detection tools based on machine learning models.

Model Type	References
Supervised Models	[26,40–44]
Deep Learning Models	[45–86,89,90]
Ensemble Models	[24,87,88]
Supervised and Deep Learning	[21]
All Models	[25]

Table 8. The most found classifiers for detecting vulnerabilities in SCs.

Models	References
Classical Models	
RF	[21,24–26,41,43,44,87,89]
SVM	[24–26,43,44,89]
KNN	[24,26,40,42,44]
DT	[26,40,44,87]
LR	[40,44]
SGD	[52]
NB	[44]
Deep Learning Models	
ANN	[21,25,26,44,61,79,81]
CNN	[21,45–47,49,52–59]
GNN	[29,49,51,60–68,70,71,88]
Autoencoder	[81]
LSTM	[25,45,47,48,50,69,73–76,83,89,90]
Bi-GRU	[46]
BERT	[45,50,79]
Other	[51,82–86] [1–2–3–4–5–6]
Ensemble Models	
Bagging	[87]
GBoost	[87]
XGBoost	[24,25,87,89] [7]
AdaBoost	[24,89]

[1] [82] DCN: deep forward propagation of a fully connected neural network in parallel. [2] [83] Contrastive Learning (CL). [3] [51] Siamese network. [4] [84] Bidirectional Gated Recurrent Unit (BiGRU). [5] [85] ResNet50. [6] [86] Cross-modal mutual learning. [7] [89] AdaBoost and Light Gradient Boosting Machine (LightGBM).

Turning to Figure 3, the pie chart details the percentage of machine-learning-driven solutions for vulnerability detection in SCs. A significant majority of this chart is accounted for deep learning models, and the remaining 21% is used for classical models (14%) and ensemble models (7%). With respect to Figure 4, RF, GNN, and XGBoost are the most commonly used classifiers among classical, deep learning, and ensemble models, respectively.

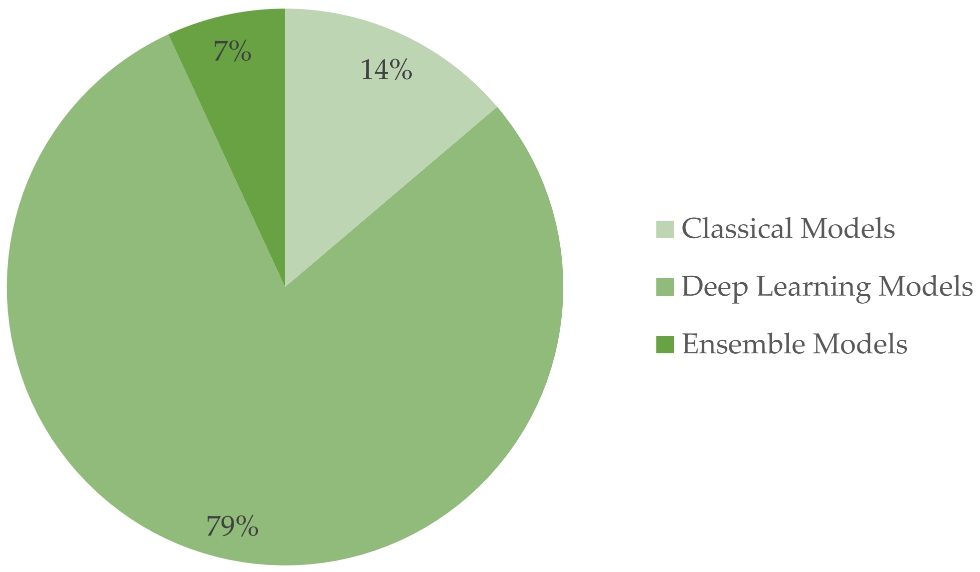

Figure 3. Percentage of machine learning-driven solutions for vulnerability detection in SCs.

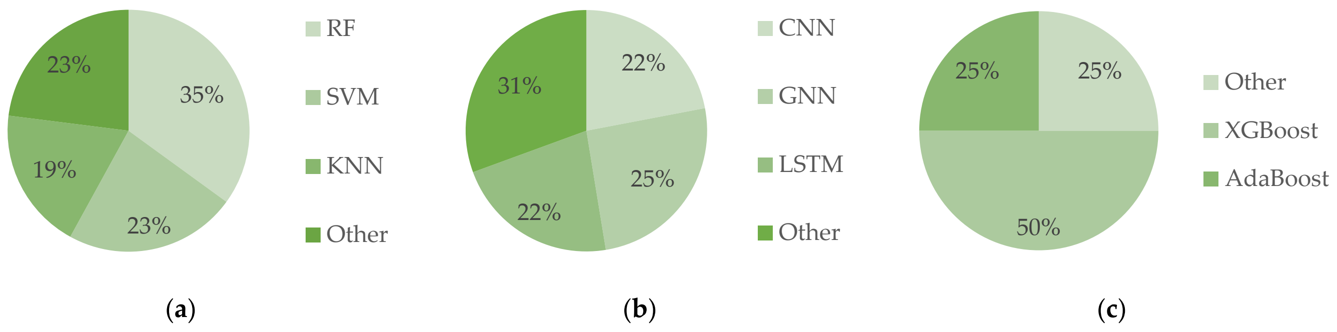

Figure 4. The diversity of classifiers among machine learning-based perspectives: (**a**) classical classifiers; (**b**) deep learning classifiers; and (**c**) ensemble classifiers. The models belonging to multiple categories are treated individually within each category.

6.2. What Machine Learning-Driven Vulnerability Detection Tools Have Addressed the Class Imbalance Issue or Unknown Vulnerabilitis?

RQ2 and RQ3 are addressed together in this subsection. Table 9 provides more details regarding vulnerability detection machine learning-driven approaches, including their method's name, number of detected vulnerabilities, which method supports unknown vulnerabilities, their benchmark datasets, and the class imbalance issue. According to our research findings, a substantial majority of frameworks identify Reentrancy as a widely recognized vulnerability in SCs. Moreover, many techniques concentrate solely on identifying one particular vulnerability.

Table 9. More details on machine learning-based vulnerability detection (VD) tools and datasets.

Reference	Method Name	# VD	Unknown VD	Dataset	Balanced Dataset
[26]	-	46		Etherscan [1]	
[21]	RFBOOF	3		--	
[24]	CotractWard	6		Etherscan [1]	✓
[42]	-	8		SolidiFI [2], SmartBugs [3], SmartBugs-wilds [4]	
[52]	MODNN	12	✓	Etherscan [1], SolidiFI [2], SmartBugs-wilds [4], EDAUB [5], Xblock [6]	✓
[53]	CodeNet	4		SolidiFI [2], SmartBugs [3]	
[54]	TMLVD	5		Etherscan [1]	
[60]	GNN	9		SolidiFI [2], Smartbugs-wilds [4], smartbugd-cuarted [7], HuanGai [8]	✓
[61]	SG and EA-RGCN	4		SmartBugs [3]	
[62]	MRNG	3		SmartBugs [3]	
[81]	-	3		Known_Attacks [9]	
[73]	Blass	4		Data are confidential	
[63]	CGE	3		Etherscan [1], VSC [10]	
[72]	-	4		Etherscan [1]	
[46]	-	13		Smartbugs-wilds [4]	✓
[87]		6		Etherscan [1], SmartBugs [3], Blockchain [11], Positive [12]	✓
[88]	HGAT	6		SmartBugs-wilds [4]	
[45]	Lightning Cat	4		SolidiFI [2]	
[64]	DA-GNN	3		*, **	
[25]	-	7		SolidiFI [2], ***	✓
[40]	-	7	✓	Etherscan [1]	
[65]	MANDO-HGT	7	✓	SolidiFI [2], SmartBugs [3], SmartBugs-wilds [4]	
[43]	-	3		Etherscan [1]	
[55]	-	1		SmartBugs-wilds [4]	
[66]	FBB-VD	-		Etherscan [1], SolidiFI[2], SmartBugs [3]	
[47]	-	1		Etherscan [1], SmartBugs [3]	
[74]	HAM-BiLSTM	1		Etherscan [1]	
[90]	-	4		Not mentioned	
[79]	-	4		Etherscan [1]	
[48]	-	1		Etherscan [1]	
[82]	Link-DC	2		Etherscan [1]	
[56]	SETT	2		[91]	
[80]	SCVulBERT	3		Smart-Contract-Dataset	
[57]	SCGRU	6		SmartBugs [3]	
[83]	Clear	3		[86]	
[78]	AFS	1		Github, Gitter chat room [13], Karl [14]	
[49]	-	4		Eth2Vec [41]	
[50]	GraBit	1		Smartbugs-wilds [4]	
[67]	-	1		Etherscan [1]	
[75]	BLSTM_ATT	1		RSC, RCS	
[68]	GVD-net	10		Smartbugs-wilds [4]	
[76]	HuntFlow	1		[92]	
[77]	-	4		Etherscan[1], Eth2Vec [41]	
[69]	SmartFuSE	3		ESC and VSC [63,93]	
[51]	SCVSN	1		SolidiFI [2], Smartbugs-wilds [4]	
[89]	CDRF	4		Etherscan [1]	
[70]	-	4		ESC and VSC [63,93]	
[71]	SAGP	2	✓	SmartBugs [3], [63]	
[84]	BiGRU	5		Etherscan [1], +, ++	✓
[58]	SC-VDM	4		Etherscan [1]	
[59]	SolGPT	4		+++	
[41]	Eth2Vec	7		Etherscan[1]	
[44]	Dynamit	1		Etherscan[1]	
[85]	-	4		#	
[86]	-	4		##	

[1] https://goto.etherscan.com; [2] https://github.com/DependableSystemsLab/SolidiFI-benchmark; [3] https://smartbugs.github.io; [4] https://github.com/smartbugs/smartbugs-wild; [5] https://library.dedaub.com; [6] https://xblock.pro/#/datasets; [7] https://github.com/smartbugs/smartbugs-curated; [8] https://github.com/xf97/HuangGai/tree/master/manualCheckDataset; [9] https://consensys.github.io/smart-contract-bestpractices/known_attacks/; [10] https://github.com/vntchain/go-vnt; [11] https://blockchain.unica.it/; [12] https://blog.positive.com/; [13] https://gitter.im/orgs/ethereum/rooms/; [14] https://github.com/cleanunicorn/karl; * https://github.com/thec00n/etherscan_verified_contracts; ** https://github.com/ZZXLX/contract_hex/tree/master; *** https://github.com/tintinweb/smart-contract-sanctuary; + https://github.com/enzymefinance/oyente; ++ https://github.com/ethereum/remix-project; +++ https://github.com/Messi-Q/Smart-Contract-Dataset; # https://github.com/sec-bit/awesome-buggy-erc20-tokens; ## https://github.com/Messi-Q/Cross-Modality-Bug-Detection.

With regards to Figure 5, looking from an overall perspective, a large proportion of published articles have not addressed unknown vulnerabilities (UV) in their proposed frameworks. Moreover, roughly a tiny minority of researchers use balanced datasets in their research. In addition, Etherscan and SmartBugs are the most favored datasets.

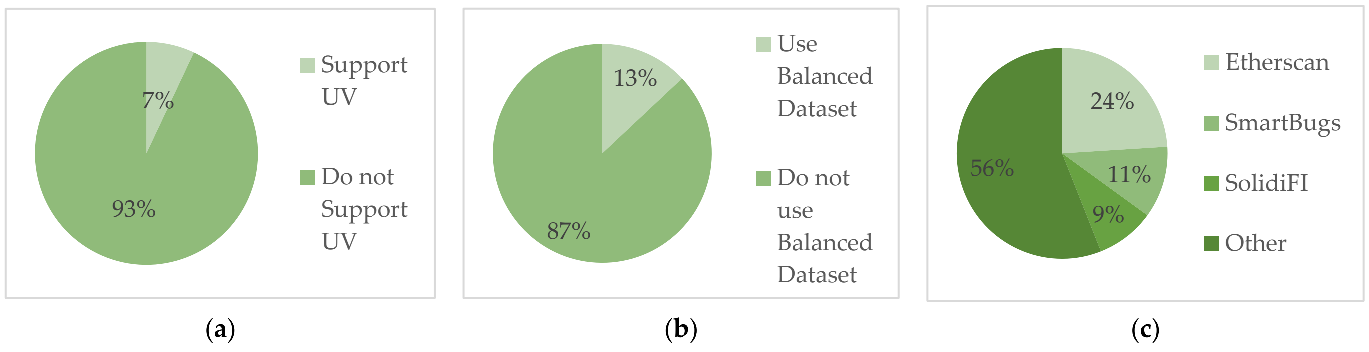

Figure 5. Overall perspective of published papers: (**a**) detect unknown vulnerabilities (UV); (**b**) address class imbalance issue; and (**c**) popular datasets.

7. Discussion

In this section, some challenges and key findings from our experiments are presented to provide insights into using machine learning-based methods for vulnerability detection.

Numerous investigations have been conducted to identify vulnerabilities, relying on traditional methods including static analysis, symbolic execution, dynamic analysis, formal verification, and fuzzy testing. These approaches frequently require tailored implementation of testing, analysis, and verification algorithms to cater to the unique SC language and vulnerability categories. Notably, their analysis algorithms may vary significantly when dealing with source code and bytecode, which restricts their adaptability to emerging languages or vulnerability types. Thus, developing vulnerability detection methods compatible with multiple platforms or programming languages can increase the generalizability of machine learning-driven solutions. Furthermore, employing single code representations like AST and control flow graphs in deep learning vulnerability detection methods can result in the occurrence of false vulnerability detection and the absence of semantic information. Moreover, the familiar deep learning method fails to consider the running logic and data flow of the program, as it treats the SC source code solely as a text sequence. As a result, it cannot extract the constructed features. In addition, these methods are limited to a particular range of vulnerability categories and do not consider the complex relationships among nodes.

Moreover, data-level methods are used by researchers to cope with the class imbalance issue, while these techniques suffer from reducing the total amount of information, increasing training time, and overfitting. Unlike data-level methods, algorithmic-level methods increase the importance of the positive class, and they can shift in a way that decreases bias towards the negative class. These characteristics can enhance the effectiveness of machine learning-based solutions in detecting multiple types of vulnerabilities in SCs.

Furthermore, the proposed unknown vulnerability techniques are focused on unknown vulnerabilities that are objectively present but cannot be discovered by existing vulnerability detection tools. In other words, unknown vulnerabilities have similarities with known vulnerabilities. Additionally, there might be particular unknown vulnerabilities that are not constrained by any limitations and may exhibit no similarities to recognized vulnerabilities. In this context, our proposed strategy is to use novelty and anomaly detection techniques since anomaly detection refers to non-typical events that only happen under exceptional circumstances. In contrast, novelty detection can identify unknown vulnerabilities that a machine learning system is not aware of during training. Moreover, we believe that the detection of unknown vulnerabilities suffers from the lack of a unique definition. Thus, the detection of unknown vulnerabilities assumes a crucial role in the development of a robust and efficient framework.

8. Conclusions and Future Work

In recent times, there has been a notable surge in interest in the application of machine learning-driven solutions for detecting vulnerabilities in smart contracts (SCs). Nevertheless, to the best of our knowledge, this is the first SLR that has focused on machine learning-based vulnerability detection approaches. This paper utilized a search query to find articles published from 2019 to 18 May 2024. Finally, we reviewed 55 articles published in peer-reviewed scientific research databases like IEEE, ScienceDirect, Springer, Wiley, Taylor & Francis, and ACM. The current study did not analyze all available studies. Non-English articles, editorials, book chapters, and books were removed from the review. However, given the growing number of studies in this area, it was impossible to cover all studies.

We proposed a taxonomy for machine learning-driven strategies under three different machine learning perspectives (i.e., classical, deep learning, and ensemble models). Moreover, we explored what strategies were implemented to deal with the class imbalance issue in this field. Furthermore, we highlighted machine learning-driven vulnerability detection methods that focused on detecting unknown vulnerabilities in SCs.

The significant observations are: (1) The graph neural network methods present a superior ability to detect vulnerabilities in SCs. (2) Ensemble learning-based methods can improve the reliability of vulnerability detection. (3) Methods for detecting unknown vulnerabilities need more attention since the existing machine learning-based vulnerability detection methods mainly focus on known vulnerabilities. The challenge lies in the fact that unknown and known vulnerabilities share certain similarities. (4) The development and reliability of vulnerability detection methods may be negatively affected by imbalanced datasets. Thus, analyzing vulnerabilities can pose a challenge when dealing with imbalanced data.

According to RQs, (1) deep learning techniques, specifically those based on graphs, have been observed to be more prevalent compared with other machine learning approaches; (2) attention to unknown vulnerabilities is essential to improving the Limited Coverage downside; and (3) the importance of the class imbalance issue has been almost ignored in machine learning-based vulnerability detection tools.

To conclude, this study can be followed by researchers who tend to develop ML-driven solutions for vulnerability detection in SCs. Alternatively, we assert that several challenges remain open and may be worth absorbing the attention of researchers.

We suggest that researchers shift frameworks onto Curriculum Learning (CL) [94] to reduce reliance on adequate expert knowledge and automatically extract key information about smart contract vulnerabilities. CL is a training strategy that trains a machine learning model from easier to more complex data. It benefits from hard and soft regularization terms in which they can not only detect noisy data and outlier samples but also reflect the importance of training samples and improve performance. In this context, an intuitive question is: Could the curriculum-like training strategy benefit machine learning-driven vulnerability detection tools?

Author Contributions: Conceptualization, R.K. and V.S.S.; methodology, R.K.; validation, V.S.S.; formal analysis, R.K.; writing—original draft preparation, R.K.; writing—review and editing, R.K. and V.S.S.; visualization, R.K.; supervision, V.S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors would like to thank the reviewers and editors for their valuable comments and guidance to make our work more comprehensive.

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

References

1. Liao, J.-W.; Tsai, T.-T.; He, C.-K.; Tien, C.-W. Soliaudit: Smart contract vulnerability assessment based on machine learning and fuzz testing. In Proceedings of the 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), Granada, Spain, 22–25 October 2019; IEEE: New York, NY, USA, 2019; pp. 458–465.
2. Feng, X.; Liu, H.; Wang, L.; Zhu, H.; Sheng, V.S. An Interpretable Model for Large-Scale Smart Contract Vulnerability Detection. SSRN 4572174. Available online: https://ssrn.com/abstract=4572174 (accessed on 23 February 2024).
3. Wu, H.; Dong, H.; He, Y.; Duan, Q. Smart contract vulnerability detection based on hybrid attention mechanism model. *Appl. Sci.* **2023**, *13*, 770. [CrossRef]
4. Wu, C.; Xiong, J.; Xiong, H.; Zhao, Y.; Yi, W. A review on recent progress of smart contract in blockchain. *IEEE Access* **2022**, *10*, 50839–50863. [CrossRef]
5. Qian, S.; Ning, H.; He, Y.; Chen, M. Multi-label vulnerability detection of smart contracts based on Bi-LSTM and attention mechanism. *Electronics* **2022**, *11*, 3260. [CrossRef]
6. Sayeed, S.; Marco-Gisbert, H.; Caira, T. Smart contract: Attacks and protections. *IEEE Access* **2020**, *8*, 24416–24427. [CrossRef]
7. Sujeetha, R.; Akila, K. Improving Coverage and Vulnerability Detection in Smart Contract Testing Using Self-Adaptive Learning GA. *IETE J. Res.* **2023**, 1–14. [CrossRef]
8. Ndiaye, M.; Konate, K. Security strengths and weaknesses of blockchain smart contract system: A survey. *Int. J. Inf. Commun. Eng.* **2022**, *16*, 134–143.
9. Wu, H.; Zhang, Z.; Wang, S.; Lei, Y.; Lin, B.; Qin, Y.; Zhang, H.; Mao, X. Peculiar: Smart contract vulnerability detection based on crucial data flow graph and pre-training techniques. In Proceedings of the 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE), Wuhan, China, 25–28 October 2021; IEEE: New York, NY, USA, 2021; pp. 378–389.
10. Singh, U.K.; Joshi, C.; Kanellopoulos, D. A framework for zero-day vulnerabilities detection and prioritization. *J. Inf. Secur. Appl.* **2019**, *46*, 164–172. [CrossRef]
11. Liu, Y.; Yu, F.R.; Li, X.; Ji, H.; Leung, V.C. Blockchain and machine learning for communications and networking systems. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1392–1431. [CrossRef]
12. Wang, M.; Xie, Z.; Wen, X.; Li, J.; Zhou, K. Ethereum smart contract vulnerability detection model based on triplet loss and BiLSTM. *Electronics* **2023**, *12*, 2327. [CrossRef]
13. Kushwaha, S.S.; Joshi, S.; Singh, D.; Kaur, M.; Lee, H.-N. Ethereum smart contract analysis tools: A systematic review. *IEEE Access* **2022**, *10*, 57037–57062. [CrossRef]
14. Li, X.; Cheng, J.; Shi, Z.; Liu, J.; Zhang, B.; Xu, X.; Tang, X.; Sheng, V.S. *Blockchain Security Threats and Collaborative Defense: A Literature Review*; Tech Science Press: Nanjing, China, 2023.
15. Fei, J.; Chen, X.; Zhao, X. MSmart: Smart Contract Vulnerability Analysis and Improved Strategies Based on Smartcheck. *Appl. Sci.* **2023**, *13*, 1733. [CrossRef]
16. Gao, C.; Yang, W.; Ye, J.; Xue, Y.; Sun, J. *sGuard+: Machine Learning Guided Rule-Based Automated Vulnerability Repair on Smart Contracts*; ACM Transactions on Software Engineering and Methodology: New York, NY, USA, 2024.
17. Szabo, N. *Formalizing and Securing Relationships on Public Networks*; First Monday: Canton, TX, USA, 1997.
18. Luu, L.; Chu, D.-H.; Olickel, H.; Saxena, P.; Hobor, A. Making smart contracts smarter. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 254–269.
19. Hu, B.; Zhang, Z.; Liu, J.; Liu, Y.; Yin, J.; Lu, R.; Lin, X. A comprehensive survey on smart contract construction and execution: Paradigms, tools, and systems. *Patterns* **2021**, *2*, 100179. [CrossRef] [PubMed]
20. Lashkari, B.; Musilek, P. Evaluation of Smart Contract Vulnerability Analysis Tools: A Domain-Specific Perspective. *Information* **2023**, *14*, 533. [CrossRef]
21. Xing, C.; Chen, Z.; Chen, L.; Guo, X.; Zheng, Z.; Li, J. A new scheme of vulnerability analysis in smart contract with machine learning. *Wirel. Netw.* **2020**, 1–10. [CrossRef]
22. Zhang, Z.; Lei, Y.; Yan, M.; Yu, Y.; Chen, J.; Wang, S.; Mao, X. Reentrancy vulnerability detection and localization: A deep learning based two-phase approach. In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering, Rochester, MI, USA, 10–14 October 2022; pp. 1–13.
23. Lakshminarayana, K.; Sathiyamurthy, K. Towards auto contract generation and ensemble-based smart contract vulnerability detection. *Int. J. Electr. Comput. Eng. Syst.* **2022**, *13*, 747–757. [CrossRef]
24. Wang, W.; Song, J.; Xu, G.; Li, Y.; Wang, H.; Su, C. Contractward: Automated vulnerability detection models for ethereum smart contracts. *IEEE Trans. Netw. Sci. Eng.* **2020**, *8*, 1133–1144. [CrossRef]
25. Colin, L.S.H.; Mohan, P.M.; Pan, J.; Keong, P.L.K. An Integrated Smart Contract Vulnerability Detection Tool Using Multi-layer Perceptron on Real-time Solidity Smart Contracts. *IEEE Access* **2024**, *12*, 23549–23567. [CrossRef]
26. Momeni, P.; Wang, Y.; Samavi, R. Machine learning model for smart contracts security analysis. In Proceedings of the 2019 17th International Conference on Privacy, Security and Trust (PST), Fredericton, NB, Canada, 26–28 August 2019; IEEE: New York, NY, USA, 2019; pp. 1–6.
27. Kiani, R.; Jin, W.; Sheng, V.S. Survey on extreme learning machines for outlier detection. *Mach. Learn.* **2024**, 1–37. [CrossRef]
28. Rouhani, S.; Deters, R. Security, performance, and applications of smart contracts: A systematic survey. *IEEE Access* **2019**, *7*, 50759–50779. [CrossRef]
29. Liu, J.; Liu, Z. A survey on security verification of blockchain smart contracts. *IEEE Access* **2019**, *7*, 77894–77904. [CrossRef]

30. Casino, F.; Dasaklis, T.K.; Patsakis, C. A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telemat. Inform.* **2019**, *36*, 55–81. [CrossRef]
31. Wang, Z.; Jin, H.; Dai, W.; Choo, K.-K.R.; Zou, D. Ethereum smart contract security research: Survey and future research opportunities. *Front. Comput. Sci.* **2021**, *15*, 1–18. [CrossRef]
32. Taylor, P.J.; Dargahi, T.; Dehghantanha, A.; Parizi, R.M.; Choo, K.-K.R. A systematic literature review of blockchain cyber security. *Digit. Commun. Netw.* **2020**, *6*, 147–156. [CrossRef]
33. Vacca, A.; Di Sorbo, A.; Visaggio, C.A.; Canfora, G. A systematic literature review of blockchain and smart contract development: Techniques, tools, and open challenges. *J. Syst. Softw.* **2021**, *174*, 110891. [CrossRef]
34. Hewa, T.M.; Hu, Y.; Liyanage, M.; Kanhare, S.S.; Ylianttila, M. Survey on blockchain-based smart contracts: Technical aspects and future research. *IEEE Access* **2021**, *9*, 87643–87662. [CrossRef]
35. Sürücü, O.; Yeprem, U.; Wilkinson, C.; Hilal, W.; Gadsden, S.A.; Yawney, J.; Alsadi, N.; Giuliano, A. A survey on ethereum smart contract vulnerability detection using machine learning. *Disrupt. Technol. Inf. Sci. VI* **2022**, *12117*, 110–121.
36. Li, H.; Dang, R.; Yao, Y.; Wang, H. A Review of Approaches for Detecting Vulnerabilities in Smart Contracts within Web 3.0 Applications. *Blockchains* **2023**, *1*, 3–18. [CrossRef]
37. Porkodi, S.; Kesavaraja, D. Smart contract: A survey towards extortionate vulnerability detection and security enhancement. *Wirel. Netw.* **2023**, 1–20. [CrossRef]
38. Sharma, P.; Jindal, R.; Borah, M.D. A review of smart contract-based platforms, applications, and challenges. *Clust. Comput.* **2023**, *26*, 395–421. [CrossRef]
39. Moher, D.; Liberati, A.; Tetxlaff, J.; Altman, D.G.; The PRISMA Group. Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. *Ann. Intern. Med.* **2009**, *151*, 264–269. [CrossRef] [PubMed]
40. Li, P.; Wang, G.; Xing, X.; Li, X.; Zhu, J. Detecting unknown vulnerabilities in smart contracts using opcode sequences. *Connect. Sci.* **2024**, *36*, 2313853. [CrossRef]
41. Ashizawa, N.; Yanai, N.; Cruz, J.P.; Okamura, S. 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, Virtual Event Hong Kong, 7 June 2021; pp. 47–59.
42. Xu, Y.; Hu, G.; You, L.; Cao, C. A novel machine learning-based analysis model for smart contract vulnerability. *Secur. Commun. Netw.* **2021**, *2021*, 5798033. [CrossRef]
43. Yang, H.; Zhang, J.; Gu, X.; Cui, Z. Smart contract vulnerability detection based on abstract syntax tree. In Proceedings of the 2022 8th International Symposium on System Security, Safety, and Reliability (ISSSR), Chongqing, China, 27–28 October 2022; IEEE: New York, NY, USA, 2022; pp. 169–170.
44. Eshghie, M.; Artho, C.; Gurov, D. Dynamic vulnerability detection on smart contracts using machine learning. In Proceedings of the 25th International Conference on Evaluation and Assessment in Software Engineering, Trondheim, Norway, 21–23 June 2021; pp. 305–312.
45. Tang, X.; Du, Y.; Lai, A.; Zhang, Z.; Shi, L. Deep learning-based solution for smart contract vulnerabilities detection. *Sci. Rep.* **2023**, *13*, 20106. [CrossRef] [PubMed]
46. Jain, V.K.; Tripathi, M. An integrated deep learning model for Ethereum smart contract vulnerability detection. *Int. J. Inf. Secur.* **2024**, *23*, 557–575. [CrossRef]
47. Shen, X.; Li, M. Smart Contract Reentrancy Vulnerability Detection Method Based on Deep Learning Hybrid Model. In Proceedings of the 2023 5th International Conference on Artificial Intelligence and Computer Applications (ICAICA), Dalian, China, 28–30 November 2023; IEEE: New York, NY, USA, 2023; pp. 33–36.
48. Demir, H.O.; Parlat, S.Z.; Gumus, A. Ethereum Blockchain Smart Contract Vulnerability Detection Using Deep Learning. In Proceedings of the 2023 7th International Symposium on Innovative Approaches in Smart Technologies (ISAS), Istanbul, Turkey, 23–25 November 2023; IEEE: New York, NY, USA, 2023; pp. 1–5.
49. Han, D.; Li, Q.; Zhang, L.; Xu, T. A smart contract vulnerability detection model based on graph neural networks. In Proceedings of the 2022 4th International Conference on Frontiers Technology of Information and Computer (ICFTIC), Qingdao, China, 2–4 December 2022; IEEE: New York, NY, USA, 2022; pp. 834–837.
50. Zhu, H.; Yang, K.; Wang, L.; Xu, Z.; Sheng, V.S. GraBit: A Sequential Model-Based Framework for Smart Contract Vulnerability Detection. In Proceedings of the 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE), Florence, Italy, 9–12 October 2023; IEEE: New York, NY, USA, 2023; pp. 568–577.
51. Chen, W.; Guo, R.; Wang, G.; Zhang, L.; Qiu, J.; Su, S.; Liu, Y.; Xu, G.; Chen, H. Smart contract vulnerability detection model based on siamese network. In Proceedings of the International Conference on Smart Computing and Communication, New York, NY, USA, 18–20 November 2022; Springer: Cham, Switzerland, 2022; pp. 639–648.
52. Zhang, L.; Wang, J.; Wang, W.; Jin, Z.; Su, Y.; Chen, H. Smart contract vulnerability detection combined with multi-objective detection. *Comput. Netw.* **2022**, *217*, 109289. [CrossRef]
53. Hwang, S.-J.; Choi, S.-H.; Shin, J.; Choi, Y.-H. CodeNet: Code-targeted convolutional neural network architecture for smart contract vulnerability detection. *IEEE Access* **2022**, *10*, 32595–32607. [CrossRef]
54. Zhou, Q.; Zheng, K.; Zhang, K.; Hou, L.; Wang, X. Vulnerability analysis of smart contract for blockchain-based IoT applications: A machine learning approach. *IEEE Int. Things J.* **2022**, *9*, 24695–24707. [CrossRef]

55. Mittal, A.; Widjaja, G.; Pecho, R.D.C.; Kiruba, R.; Roque, J.M.F.; Chandra, A. Blockchain Based Abstract Syntax Tree to Detect Vulnerability in IOT-Enabled Smart Contract. In Proceedings of the 2023 Second International Conference on Smart Technologies For Smart Nation (SmartTechCon), Singapore, 18–19 August 2023; IEEE: New York, NY, USA, 2023; pp. 270–275.
56. Feng, M.; Mi, W.; Zhang, X.; Chen, B.; Huang, M. A Smart Contract Vulnerability Detection Model Based on Multi-Type Features and Pre-Training Techniques. In Proceedings of the 2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security Companion (QRS-C), Chiang Mai, Thailand, 22–26 October 2023; IEEE: New York, NY, USA, 2023; pp. 280–289.
57. Liang, J.; Zhai, Y. SCGRU: A Model for Ethereum Smart Contract Vulnerability Detection Combining CNN and BiGRU-Attention. In Proceedings of the 2023 8th International Conference on Signal and Image Processing (ICSIP), Wuxi, China, 8–10 July 2023; IEEE: New York, NY, USA, 2023; pp. 831–837.
58. Zhou, K.; Cheng, J.; Li, H.; Yuan, Y.; Liu, L.; Li, X. SC-VDM: A lightweight smart contract vulnerability detection model. In Proceedings of the Data Mining and Big Data: 6th International Conference, DMBD 2021, Guangzhou, China, 20–22 October 2021; Proceedings, Part I 6. Springer: Singapore, 2021; pp. 138–149.
59. Zeng, S.; Zhang, H.; Wang, J.; Shi, K. SolGPT: A GPT-Based Static Vulnerability Detection Model for Enhancing Smart Contract Security. In Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing, Tianjin, China, 20–22 October 2023; Springer: Singapore, 2023; pp. 42–62.
60. Cai, J.; Li, B.; Zhang, J.; Sun, X.; Chen, B. Combine sliced joint graph with graph neural networks for smart contract vulnerability detection. *J. Syst. Softw.* **2023**, *195*, 111550. [CrossRef]
61. Chen, D.; Feng, L.; Fan, Y.; Shang, S.; Wei, Z. Smart contract vulnerability detection based on semantic graph and residual graph convolutional networks with edge attention. *J. Syst. Softw.* **2023**, *202*, 111705. [CrossRef]
62. Liu, H.; Fan, Y.; Feng, L.; Wei, Z. Vulnerable Smart Contract Function Locating Based on Multi-Relational Nested Graph Convolutional Network. *J. Syst. Softw.* **2023**, *204*, 111775. [CrossRef]
63. Liu, Z.; Qian, P.; Wang, X.; Zhuang, Y.; Qiu, L.; Wang, X. Combining graph neural networks with expert knowledge for smart contract vulnerability detection. *IEEE Trans. Knowl. Data Eng.* **2021**, *35*, 1296–1310. [CrossRef]
64. Zhen, Z.; Zhao, X.; Zhang, J.; Wang, Y.; Chen, H. DA-GNN: A smart contract vulnerability detection method based on Dual Attention Graph Neural Network. *Comput. Netw.* **2024**, *242*, 110238. [CrossRef]
65. Nguyen, H.H.; Nguyen, N.-M.; Xie, C.; Ahmadi, Z.; Kudendo, D.; Doan, T.-N.; Jiang, L. MANDO-HGT: Heterogeneous Graph Transformers for Smart Contract Vulnerability Detection. In Proceedings of the 2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR), Melbourne, Australia, 15–16 May 2023; IEEE: New York, NY, USA, 2023; pp. 334–346.
66. Xiong, H.; Zhong, Y.; Wu, C.; Yi, W.; Zhao, Y. A Multi-code Representation Fusion Smart Contract Vulnerability Line Detection Method Based on Graph Neural Network. In Proceedings of the 2023 11th International Conference on Information Systems and Computing Technology (ISCTech), Qingdao, China, 30 July–1 August 2023; IEEE: New York, NY, USA, 2023; pp. 28–33.
67. Wang, Z.; Wu, W.; Zeng, C.; Yao, J.; Yang, Y.; Xu, H. Smart contract vulnerability detection for educational blockchain based on graph neural networks. In Proceedings of the 2022 International Conference on Intelligent Education and Intelligent Research (IEIR), Wuhan, China, 18–20 December 2022; IEEE: New York, NY, USA, 2022; pp. 8–14.
68. Wang, Z.; Zheng, Q.; Sun, Y. Gvd-net: Graph embedding-based machine learning model for smart contract vulnerability detection. In Proceedings of the 2022 International Conference on Algorithms, Data Mining, and Information Technology (ADMIT), Xi'an, China, 23–25 September 2022; IEEE: New York, NY, USA, 2022; pp. 99–103.
69. Lin, X.; Zhou, M.; Cao, S.; Wang, J.; Sun, X. The Best of Both Worlds: Integrating Semantic Features with Expert Features for Smart Contract Vulnerability Detection. In Proceedings of the International Conference on Blockchain and Trustworthy Systems, Haikou, China, 8–10 August 2023; Springer: Singapore, 2023; pp. 17–31.
70. Wei, Z.; Zheng, W.; Su, X.; Tao, W.; Wang, T. A Graph Neural Network-Based Smart Contract Vulnerability Detection Method with Artificial Rule. In Proceedings of the International Conference on Artificial Neural Networks, Crete, Greece, 26–29 September 2023; Springer: Cham, Switzerland, 2023; pp. 241–252.
71. Zou, L.; Gong, C.; Wu, Z.; Tan, J.; Tang, J.; Jiang, Z.; Li, D. A General Smart Contract Vulnerability Detection Framework with Self-attention Graph Pooling. In Proceedings of the International Conference on Blockchain and Trustworthy Systems, Haikou, China, 8–10 August 2023; Springer: Singapore, 2023; pp. 3–16.
72. Liu, Z.; Jiang, M.; Zhang, S.; Zhang, J.; Liu, Y. A smart contract vulnerability detection mechanism based on deep learning and expert rules. *IEEE Access* **2023**, *11*, 77990–77999. [CrossRef]
73. Ren, X.; Wu, Y.; Li, J.; Hao, D.; Alam, M. Smart contract vulnerability detection based on a semantic code structure and a self-designed neural network. *Comput. Electr. Eng.* **2023**, *109*, 108766. [CrossRef]
74. Xu, G.; Liu, L.; Zhou, Z. Reentrancy vulnerability detection of smart contract based on bidirectional sequential neural network with hierarchical attention mechanism. In Proceedings of the 2022 International Conference on Blockchain Technology and Information Security (ICBCTIS), Huaihua City, China, 15–17 July 2022; IEEE: New York, NY, USA, 2022; pp. 56–59.
75. Qian, P.; Liu, Z.; He, Q.; Zimmermann, R.; Wang, X. Towards automated reentrancy detection for smart contracts based on sequential models. *IEEE Access* **2020**, *8*, 19685–19695. [CrossRef]
76. Zhou, K.; Cheng, J.; Liu, L.; Sheng, V.S. HuntFlow: Search the Arithmetic Vulnerability in Ethereum Smart Contract. In Proceedings of the International Conference on Artificial Intelligence and Security, Qinghai, China, 15–20 July 2022; Springer: Cham, Switzerland, 2022; pp. 158–168.

77. Hu, Z.; Tsai, W.-T.; Zhang, L. Smart-contract vulnerability detection method based on deep learning. In Proceedings of the International Conference on Smart Computing and Communication, New York, NY, USA, 18–20 November 2022; Springer: Cham, Switzerland, 2022; pp. 450–460.
78. Wang, B.; Chu, H.; Zhang, P.; Dong, H. Smart contract vulnerability detection using code representation fusion. In Proceedings of the 2021 28th Asia-Pacific Software Engineering Conference (APSEC), Taipei, Taiwan, 6–9 December 2021; IEEE: New York, NY, USA, 2021; pp. 564–565.
79. Vu, D.; Nguyen, T.; Tong, V.; Souihil, S. Enhancing Multi-Label Vulnerability Detection of Smart Contract Using Language Model. In Proceedings of the 2023 5th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), Paris, France, 11–13 October 2023; IEEE: New York, NY, USA, 2023; pp. 1–4.
80. Zeng, S.; Chen, R.; Zhang, H.; Wang, J. A High-Performance Smart Contract Vulnerability Detection Scheme Based on BERT. In Proceedings of the 2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS), Ocean Flower Island, China, 17–21 December 2023; IEEE: New York, NY, USA, 2023; pp. 653–658.
81. Narayana, K.L.; Sathiyamurthy, K. Automation and smart materials in detecting smart contracts vulnerabilities in Blockchain using deep learning. *Mater. Today Proc.* **2023**, *81*, 653–659. [CrossRef]
82. Li, N.; Liu, Y.; Li, L.; Wang, Y. Smart contract vulnerability detection based on deep and cross network. In Proceedings of the 2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA), Changchun, China, 20–22 May 2022; IEEE: New York, NY, USA, 2022; pp. 533–536.
83. Chen, Y.; Sun, Z.; Gong, Z.; Hao, D. Improving Smart Contract Security with Contrastive Learning-based Vulnerability Detection. In Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, Lisbon, Portugal, 14–20 April 2024; pp. 1–11.
84. Song, S.; Yu, X.; Ma, Y.; Li, J.; Yu, J. Multi-model Smart Contract Vulnerability Detection Based on BiGRU. In Proceedings of the International Conference on Neural Information Processing, Changsha, China, 20–23 November 2023; Springer: Singapore, 2023; pp. 3–14.
85. Qin, S.-J.; Liu, Z.; Ren, F.; Tan, C. Smart contract vulnerability detection based on critical combination path and deep learning. In Proceedings of the 2022 12th International Conference on Communication and Network Security, Beijing, China, 1–3 December 2022; pp. 219–226.
86. Qian, P.; Liu, Z.; Yin, Y.; He, Q. Cross-modality mutual learning for enhancing smart contract vulnerability detection on bytecode. In Proceedings of the ACM Web Conference 2023, Austin, TX, USA, 30 April–4 May 2023; pp. 2220–2229.
87. JJ, L.; Singh, K.; Chakravarthi, B. Digital forensic framework for smart contract vulnerabilities using ensemble models. *Multimed. Tools Appl.* **2024**, *83*, 51469–51512. [CrossRef]
88. Ma, C.; Liu, S.; Xu, G. HGAT: Smart contract vulnerability detection method based on hierarchical graph attention network. *J. Cloud Comput.* **2023**, *12*, 93. [CrossRef]
89. Huang, M.; Yang, J.; Liu, C. CDRF: A Detection Method of Smart Contract Vulnerability Based on Random Forest. In Proceedings of the International Conference on Provable Security, Wuhan, China, 20–22 October 2023; Springer: Cham, Switzerland, 2023; pp. 407–428.
90. Zhang, X.; Li, J.; Wang, X. Smart contract vulnerability detection method based on bi-lstm neural network. In Proceedings of the 2022 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), Dalian, China, 20–21 August 2022; IEEE: New York, NY, USA, 2022; pp. 38–41.
91. Liu, Z.; Qian, P.; Yang, J.; Liu, L.; Xu, X.; He, Q.; Zhang, X. Rethinking smart contract fuzzing: Fuzzing with invocation ordering and important branch revisiting. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 1237–1251. [CrossRef]
92. Durieux, T.; Ferreira, J.F.; Abreu, R.; Cruz, P. Empirical review of automated analysis tools on 47,587 ethereum smart contracts. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, Seoul, Republic of Korea, 5–11 October 2020; pp. 530–541.
93. Zhuang, Y.; Liu, Z.; Qian, P.; Liu, Q.; Wang, X.; He, Q. Smart contract vulnerability detection using graph neural networks. In Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, Yokohama, Japan, 7–15 January 2021; pp. 3283–3290.
94. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum learning. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 41–48.