Vulnerabilities and Open Issues of Smart Contracts: A Systematic Mapping

Gabriel de Sousa Matsumura[1(✉)], Luciana Brasil Rebelo dos Santos[1(✉)], Arlindo Flavio da Conceição[2(✉)], and Nandamudi Lankalapalli Vijaykumar[2,3(✉)]

[1] Federal Institute of Education, Science and Technology of São Paulo, São Paulo, Brazil
lurebelo@ifsp.edu.br
[2] Federal University of São Paulo, São Paulo, Brazil
{arlindo.conceicao,vijaykumar}@unifesp.br
[3] National Institute for Space Research, São José dos Campos, Brazil

Abstract. Smart Contracts (SCs) are programs stored in a Blockchain to ensure agreements between two or more parties. The combination of factors like the newness of the field, with less than a decade of practical use, and a high and growing popularity in industry, leads to an increase in the possibility of severe security issues occurrence. Even though, systems based on Blockchain and SCs technologies inherit several benefits, like a tamper-proof decentralized ledger and anonymous transactions. This way, even with security issues, the trend is that its popularity will increase. Due the unchangeable essence of Blockchain, failures or errors in SCs become perpetual once published, being critical to mitigate them, since the involvement of huge economic assets are common. To handle this, the academic and industrial communities have been expanding their efforts in this field, with a growth in research publications similar to exponential. Aiming at reasoning about the current state-of-art in vulnerabilities and open issues over Blockchain SCs, we conducted a systematic literature mapping over 32 primary and 18 secondary selected articles. As contribution, this work discusses and relates the selected papers, identifying gaps that may lead to research topics for future work.

Keywords: Blockchain · Smart Contracts · Systematic mapping · Tertiary study · Vulnerabilities · Security

1 Introduction

Software reliability is highly relevant given the dependence that modern society has on software systems. Defects can result in minor annoyances, such as not accessing social networks for a while. They may become serious, leading to severe financial losses. Amid advances in software technology, the idea of digital financial transactions emerges, to facilitate money transactions between people anywhere in the world, without the need for mediators, such as banks or brokers.

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12952, pp. 3–18, 2021.
https://doi.org/10.1007/978-3-030-86973-1_1

Blockchain [14] is a technology to make these financial operations viable. It is a distributed ledger that records all transactions of a digital currency, so it is constantly updated. In 2015, the introduction of Smart Contracts (SCs) extended the functionality of Blockchain.

An SC is a program designed to guarantee the execution of a transaction between two or more parties. There are fundamental differences between SC and standard software, and these differences introduce new vulnerabilities and concerns. SCs are immutable computer programs stored in a Blockchain and verified and executed by some of its nodes in a distributed manner [@13][1].

Notice that the code of an SC cannot be modified to correct defects [19], i.e., once implanted, it cannot be updated. Thus, the correctness and security of SCs are necessary, as failures can cause millionaire losses, as already extensively reported in the literature [19].

There is an extensive number of cases where bugs and breaches in SCs led to severe economic losses. In 2016, an attack stole approximately $50 million worth of Ether by exploiting a flaw in the Distributed Autonomous Organization (DAO) source code [5][@13]. In 2017, the multi-signature wallet Parity had embezzlement of about $30 million as a consequence of an attack on Ethereum. In 2018, about $900 million was stolen due to the BEC (BeautyChain) token attack [@13]. Also in 2018, researchers performed a security analysis of nearly a million SCs using the MAIAN tool, resulting in 34,200 SCs flagged as vulnerable [13]. Then, with a random sample of 3,759 contracts obtained from the vulnerable contracts set, they found that 3,686 SCs had an 89% probability of vulnerability [13][@5].

Although recent studies have proposed a huge number of tools and techniques to detect bugs in SCs, the literature shows that the development of SCs needs to be improved in many ways, in order to minimize the problems detected so far. A possible reason for this situation is the lack of a comprehensive mapping of all the existing bugs [7] and how to deal with each of them. The present paper carried out a systematic literature mapping to characterize the techniques, methods, and tools that deal with SCs' vulnerabilities and showing recent advances in this area. Thus, the results can identify eventual gaps and use this information to mitigate problems in future research.

While conducting the study, in the stage of Data Extraction and Synthesis, we have noticed that a relevant number of secondary studies were returned. So, to make a more comprehensive contribution, we have decided to include them as part of the mapping, presenting also a tertiary study. In this way, the answers to the research questions consider both primary and secondary studies.

This paper is organized as follows. Section 2 shows the background to this paper. The secondary studies related to this mapping are commented in Sect. 3. Section 4 describes the research questions and research methods adopted.

[1] In this text, for a question of space and simplicity, we differentiate among *0)* regular references, *1)* primary studies (denoted with #) and *2)* secondary studies (written with @). Regular references are available at the end of the text. In order to have access to the aforementioned primary and secondary studies, the reader must check the list of papers in https://bit.ly/3tNkIG6 [11].

Section 5 presents the results obtained, making considerations and discussions about the outcomes. Section 6 discusses the potential threats to the research validity. Finally, Sect. 7 presents final remarks about the mapping conducted.

2 Background

This section address needed concepts for understanding this work.

2.1 Blockchain

A Blockchain platform is a decentralized ledger. It stores transactions permanently, in a secure and auditable way [3]. Blockchain technology was popularized by Bitcoin in 2008 [12], enabling reliable digital financial transactions without a trusted third party between anonymous entities. Bitcoin merged the qualities of cryptography techniques and peer-to-peer networks. In this network, miners verify transactions as they occur, checking signatures and balances. Then, a distributed consensus algorithm is used to create a block with the validated transactions. The block is broadcasted to the entire network so that all nodes can, after validating its correctness, add the new blocks to their copy of the ledger [1]. The consensus algorithm maintains a trusted network without the need to trust any node in particular [@5].

2.2 Smart Contracts

A Smart Contract (SC) is software that runs on the Blockchain and represents an agreement between non-trusting participants [1]. The most popular Blockchain platform that supports SCs is Ethereum [4,18]. However, the development of complex SCs is not trivial [1]. For example, the Ethereum platform uses a network of Turing complete virtual machines to validate transactions [1]. In Ethereum, the programmer uses a high-level programming language (e.g., Solidity) to write the SCs. The interaction between users and SCs occurs by sending a transaction to the contract address. An SC can call other SCs during its execution and can pass data as parameters, making it possible to execute untrusted code [2]. So there is a need to ensure the correctness of the executable codes, but that is a challenging task [@13].

Before the Blockchain technology arising, although the concept of SCs already existed [16], they were not well developed [@5]. The participation of central authorities or resource managers was a requirement at the time, limiting strongly the usefulness of smart contracts since these third parties are able to handle agreements themselves [@10, @13]. Since there is no need for a trusted third party in the Blockchain, smart contracts can be advantageously employed.

3 Related Literature

In this section, we provide a concise overview of the main secondary studies related to our systematic mapping. Regarding publication sources, Table 1 shows

that journals are composed of the following study types: systematic reviews, surveys (37.5% each), and systematic mappings (25%). Regarding the conference publications, there was a multivocal review, a systematic review (11.11% each) and surveys (77.78%). Also, we have an e-print survey publication. So, in journals, systematic approaches are more often, while surveys are more often in conferences. Even though the e-print survey is not yet published in a journal or conference, it is worth mentioning that it was included since it has rich information for our paper.

For the classification of the secondary papers, we not only consider the types defined by the authors, but also the definition of each type. Systematic approaches have a precise methodology, like the one we present in the next section to minimize subjectivity and maximize reproducibility. Systematic mappings and reviews utilize a similar methodology, the main difference is the scope. Mappings are more likely to be high-level, having a broader view about a topic and a review is more likely to be low-level, aiming at a specific perspective of a subject, often comparing two specific topics. The multivocal literature review can also have a systematic methodology, but it includes in its scope gray literature, for instance, technical reports, thesis, dissertations, etc. Finally, a survey does not utilize a precise methodology to select their primary papers. They have the tendency to have more subjectivity, increasing the odds that relevant articles may have been left behind.

Table 1. Related secondary papers types and publication sources

Sources and secondary type	Journal	Conference	E-Print	Total
Multivocal literature review	0	1	0	1
Survey	3	7	1	11
Systematic literature mapping	2	0	0	2
Systematic literature review	3	1	0	4
Total	8	9	1	18

Table 2. Focus areas identified in the related secondary papers

Areas		@Secondary articles	Total (%)
Issues	Vulnerability	1, 3, 4, 6, 7, 11, 14, 15, 17	38.89
	Attacks	1, 7, 12, 13, 15, 17	33.33
	External data	8, 9, 17	16.67
Solutions	Design	4, 6, 9, 17, 18	27.78
	Implementation	2, 4, 5, 6, 7, 8, 12, 17, 18	50.00
	Software testing	3, 5, 13, 14, 16, 17, 18	38.89
	Formal methods	4, 5, 6, 9, 10, 13, 14, 16, 17, 18	55.56
	Machine learning	11, 12, 17, 18	22.22
	Tools	3, 4, 5, 6, 9, 10, 13, 14, 16	50.00
	Monitoring	4, 9, 12, 17, 18	27.78
Blockchain platform		11, 12, 14, 16	22.22

In Table 2, we provide a simple categorization of the related secondary articles concerning the kind of their contributions, being common the availability of some kind of classifications and, sometimes, a taxonomy. In these issues, among papers in vulnerability scope, all of them, except [@17], mention vulnerabilities related to programming languages. Also all, but [@15] and [@17], mention vulnerabilities related to Blockchain platform and virtual machine. Considering the papers on vulnerability, we can highlight that [@15] provides a great systematization of Ethereum vulnerabilities based on the Common Weakness Enumeration that may probably be extended to other platforms. About the external data scope, all of them mention oracles, and [@17] mentions verifiable third-party and cryptography technology as possible solutions for off-chain interactions.

For solutions concerning design, we selected papers that handle modeling or specifications. For the case of implementation, solutions like design patterns [@2, @4, @8, @17], templates [@4, @17], standards [@5], more secure domain-specific languages [@5, @9, @17] and code generation [@12] are mentioned. In relation to software testing, all of the selected papers mention fuzzing test, being that [@16] also mentions test case generation. Considering the papers that mention machine learning we highlight [@12], that, unlike others, focuses on Blockchain and machine learning and, therefore, handles them more deeply, with less emphasis on SCs. With respect to monitoring SCs after deployment, papers mention runtime monitoring [@4], bug bounty [@4, @9, @17], and monitoring strategies [@18], being also worthy to mention that monitoring is often related to the update of SCs [@12, @17].

4 Research Method

The research method adopted in this Systematic Literature Mapping (SLM) is defined based on the guidelines [9], involving three main phases: (i) Planning: referring to the pre-review activities, aiming to define the research questions, inclusion and exclusion criteria, study sources, ~~re~~search string, and mapping procedures to establish a protocol review; (ii) Realization: search and selection of studies, aiming at extracting and synthesizing their data; (iii) Report: final phase whose objective is to write the results and disseminate them to potentially interested parties, using the results to answer the research questions.

The research questions are listed in next section. The study selection is explained in Sect. 4.2, and data extraction is presented in Sect. 4.3.

4.1 Research Questions

This mapping presents an overview of the current state of research on the problems/issues directly involved with SCs in the Blockchain context. Table 3 shows the Research Questions (RQ) and the rationales considering them in this SLM.

Table 3. Research questions and their rationales

N°	Research question	Rationale
RQ1	When and where have the studies on vulnerabilities and open issues on SCs been published?	This question is to understand whether there are specific Publication sources for these studies, and when they have been published
RQ2	What are the problems/issues with respect to SCs?	This question identifies what are the problems/issues directly related to the use of SCs in the Blockchain context
RQ3	How are the problems/issues categorized or classified?	This question investigates how the problems/issues related to the use and development of SCs have been categorized/classified, checking if there is a proper taxonomy
RQ4	What are the proposed solutions to the problems/issues identified within the context of SCs?	The aim is to determine what are the proposed solutions to the problems/issues identified in RQ2
RQ5	Are there methods and tools available for the identified solutions?	This question verifies, among the methods and tools identified in RQ4, what are the limits of using each one in terms of practical purposes

4.2 Study Selection

Source and Search String. The search was performed in the Scopus[2] repository. We chose this database because, in the initial searches, Scopus returned a larger set of results compared to others. It was also noted that many papers found on other platforms were also returned in Scopus, considering the same search string. Finally, we defined the following search string for this mapping:

TITLE-ABS-KEY("smart contract" w/12 (problem OR issue))

The main focus of the search was to identify papers that addressed problems/issues about smart contracts used in the Blockchain context. During the test of search strings, it was clear that the results of the string *("smart contract" AND (problem OR issue))* contemplated the papers being looked for. With the goal of narrow the resulting papers to our scope, we used the proximity operator available in Scopus, expressed by *w/x*, where x is the maximum number acceptable of words between two terms, regardless of the order of the terms. Using *x=12* it was possible find previously defined control papers.

Inclusion and Exclusion Criteria. The selection criteria are organized into two inclusion criteria (IC) and eight exclusion criteria (EC). The inclusion criteria are: (IC1) Study must include the use of smart contracts in Blockchain

[2] http://www.scopus.com.

context; and (IC2) Study must include problems or solutions related to the use of smart contracts. The exclusion criteria are: (EC1) Study has no abstract; (EC2) Study is just an abstract or extended abstract without full text; (EC3) Study is not a primary study, such as editorials, summaries of keynotes, and tutorials; (EC4) Study is not written in English; (EC5) Study is a copy or an older version of another publication already considered; (EC6) No access to the full paper; (EC7) Study with a publication date prior to 2015; and (EC8) Study should contain problems related to smart contracts itself, problems about smart contract applications aren't enough.

Data Storage. We used a spreadsheet to gather all the relevant data from the returned studies in the searching phase (e.g., id, title and bibliographic reference), cataloging and storing each publication appropriately.

Assessment. Before conducting the mapping, we checked our protocol. We defined a pre-selected set of papers, which should be present in the returned set of studies. Regarding the review process, stage 1 was conducted by one of the authors, and the papers not excluded were validated by the other authors. Stage 2 was conducted by all authors, and we equally divided the remaining papers. Throughout the review process, in cases of doubt, the papers were not excluded, and, in a meeting, the authors reached a consensus on whether or not to include the studies.

4.3 Data Extraction and Synthesis

The search resulted in 341 publications from Scopus. We followed a selection process with 2 main stages in this SLM (Fig. 1).

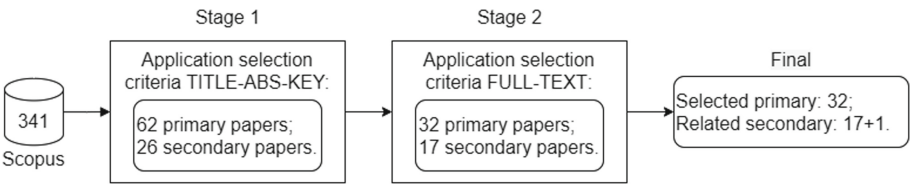


Fig. 1. Search and selection SLM process

In stage 1, we applied the selection criteria (inclusion and exclusion criteria) over title, abstract and keywords, resulting in 62 papers (reduction of approximately 82%). 1 paper was eliminated by EC1 (Study has no abstract); 10 by EC2 (Study without full text); 34 by EC3 (Study is not a primary study); 8 by EC4 (Study is not written in English) 3 by EC5 (Study is a copy or an older version of another study already considered); 2 by EC7 (Study with publication date prior to 2015); and 221 by EC8 (Study should contain problems related to smart contracts itself, problems about smart contract applications aren't enough). In

stage 2, we applied the selection criteria considering the full text, resulting in 32 studies (a reduction of approximately 48%). 1 paper was eliminated by EC2, EC3 and EC5; and 26 by EC8. From the first stage on, secondary works were also selected among the EC3 excluded papers. We separated those that didn't fit other exclusion criteria, obtaining 17 papers at the end of the 2nd stage. One more was included outside the search results because it was indicated by an expert due to its relevance, totaling 18 papers. The related work supported the determination of answers to our research questions through taxonomies, classifications and other contributions.

5 Results and Discussions

The SLM study was carried out according to Sect. 4. Here we show the results for each of the research questions and discuss them. To answer the questions, an id was determined for each of the articles, as in [11].

To assist in the analysis and systematize the extracted data, categories for classifying the studies were defined, one for each research question. We considered the characteristics of the selected studies, both reusing categories already considered in the literature and defining new categories when necessary. It is possible that the same paper fits multiple categories. In cases of doubts about whether an article belongs to a certain category, we simply didn't include it.

5.1 Question 1: When and Where Have the Studies Been Published?

Over the years, publications on smart contracts for Blockchain, had a growth similar to quadratic for both the primary and secondary papers, as presented in Fig. 2. We conducted our search in January 2021. So, we have only two papers this year, but considering that the period was less than a month, the number was not bad. Even with this growth rate, we find that the maturity of this field is still low. The majority, 72% of all 50 papers were from conferences, and only 26% from journals, is that in absolute terms it is still new compared to fields already established in the Information Technology. This is expected since it has not been even a decade since the first platform with support for SCs was inaugurated.

We count the frequency of the countries from where the papers were written. They are shown in Fig. 3. Countries that appeared only once (3.67% each) were Italy, Malta, Saudi Arabia, Hong Kong, Thailand, Qatar and Morocco; twice (6.67% each) were Austria, Switzerland, United Kingdom, France, Japan and Russia; three times (10%) was Germany; four times (13.33% each) were Singapore and Australia; seven times (23.33%) was the USA; and eleven times (36.67%) was China.

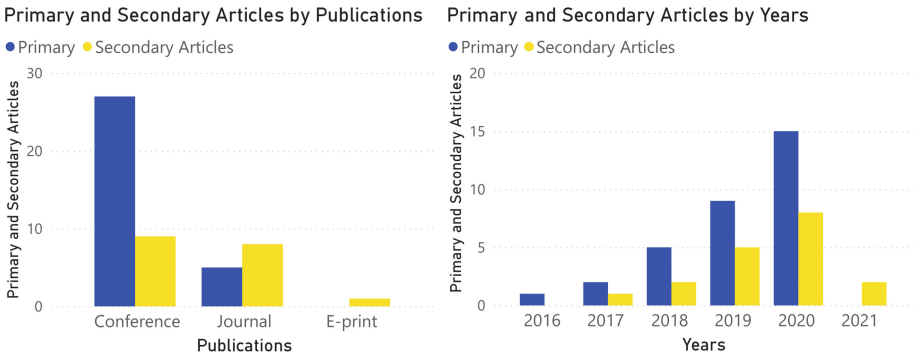


Fig. 2. Frequency of year and source of publications of primary and secondary papers

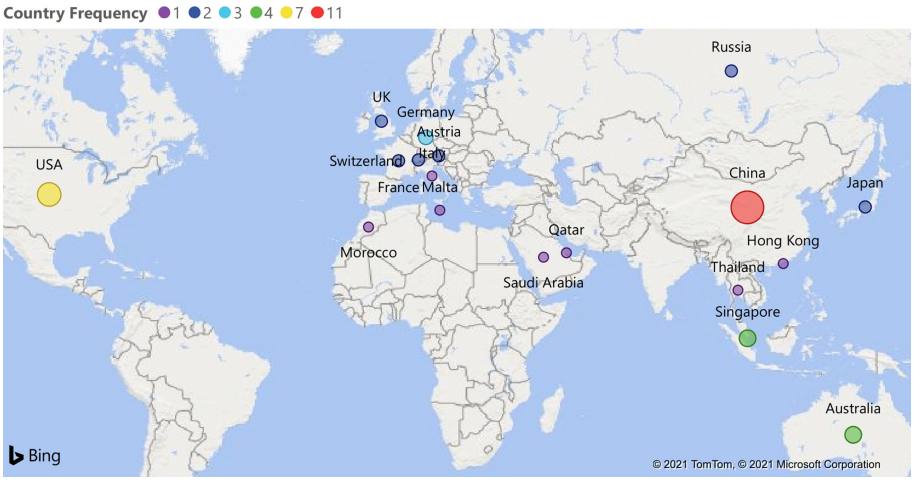


Fig. 3. Countries of institutions where the authors of the primary papers are affiliated

5.2 Question 2: What Are the Problems/Issues with Respect to Smart Contracts?

Most of the problems/issues pointed out in the literature were the exploitation of specific vulnerabilities in Ethereum. Solidity, the most used language in Ethereum, has several known problems[3].

In general, the articles seek to automatically find vulnerabilities by analyzing the source code, identifying the use of critical instructions (e.g., transfer funds), and analyzing the possibility of the code reaching these critical instructions.

The most common critical instructions mentioned were reentrancy, integer overflow, withdrawn and unprotected SELFDESTRUCT instruction. Besides

[3] The address https://swcregistry.io/ has an updated list of known vulnerabilities.

security issues, some papers mention issues related to privacy [@8, @9, @11], performance [@5, @8, @9], governance and legality [@11].

Some articles studied problems not directly related to the source code of the SCs. They deal with external conditions of application execution, such as malicious or irrational use of applications, redesign of software engineering techniques to suit better the context of Blockchain, or monitoring of the use of decentralized applications (auditing).

5.3 Question 3: How Are the Problems/Issues Categorized or Classified?

There was no common proposal to classify problems. Among primary papers, Luu*et al.* [#1], for example, organized vulnerabilities based on their origin or cause, namely: dependence on the order of transactions, dependence on time, inadequate handling of exceptions and indentation. [#15] used a similar classification. Another article by Petar Tsankov [#4] organized the vulnerabilities as follows: insecure coding, unsafe transfers, unsafe inputs, transaction reordering, and reentrancy issues. Groce*et al.* [#19] prefer to classify the vulnerabilities by their severity and the difficulty of being exploited.

Regarding secondary papers, Wang*et al.* [@17] provides an interesting taxonomy over SC security issues, also categorizing respective solutions for them, being the main problems: abnormal contract, program vulnerability and exploitable habitat. Huang*et al.* [@4] overview security themes from an SC lifecycle perspective. In each phase, vulnerabilities may be introduced or exploited, just as methods can be used to avoid them, being the phases: security design, security implementation, testing before deployment and monitoring and analysis. The systematization of 10 SC vulnerabilities classes based on Common Weakness Enumerator, provided by [@15], is worth mentioning.

Even though all the proposed problem classifications are different, they can be the basis to design a more complete taxonomy for the critical issues in SC. Also, there are examples where that issue and solution classifications fit well together [@17]. It might be worth considering combining them.

5.4 Question 4: What Are the Proposed Solutions to the Problems/Issues Identified Within the Context of Smart Contracts?

The material extracted for this question was organized considering related work that had categories or taxonomies for solutions [17] [@4, @13, @17]. Based on the resulting papers, we defined the categories as in Table 4, showing their distribution.

Table 4. Distribution over the proposed solutions to the problems/issues identified

Methods		#Primary articles	Total (%)
Design	Modeling	1, 2, 3, 4, 5, 7, 8, 10, 12, 14, 15,16, 20, 24, 25, 28, 29, 31, 32	59.38
	Specification	4, 7, 14, 15, 16, 18, 24, 25	25.00
Implementation	Design pattern	6, 8, 11, 17	12.50
Verification	Theorem proving	1, 5, 10, 18, 26, 27, 30	21.88
	Model checking	7, 15, 31	9.38
	Abstract interpretation	2, 24	6.25
	Symbolic execution	1, 5, 10, 13, 14, 16, 18, 19, 26, 30	31.25
	Runtime verification	8	3.13
	Software testing	6, 16, 19, 21, 32	15.62
	Machine learning	9, 16, 22, 23	12.50
	Manual auditing	19	3.13

Regarding the proposed solutions, we only included papers that were certain of their classifications. The dominant category was Design Modeling with 19 papers, among them: Event-B, a set-based method [#31]; State-Transition Systems like Colored Petri nets [#28], Markov decision processes [#16], Kripke structure [#15], Dynamic Automata with Events [#8] and state machines [#7]; Abstract Syntax Tree-Level Analysis [#12, #25, #29]; Control Flow Graph [#1, #2, #4, #5, #10, #12, #14, #16, #20, #24, #32]; and Agent-based model [#3], borrowed from Game Theory. Yet in Design realm, Specification has 8 papers, among them: logic like Computation Tree Logic [#7] and Linear Temporal Logic [#15]; Horn Clauses [#24]; Hoare-Style Properties [#25]; and Program Path-Level Patterns like execution traces [#14, #16, #18] and datalog [#4].

In the Verification aspect, the most often approach is Symbolic Execution with 10 articles [#1; #5, #10, #13, #14, #16, #18, #19, #26, #30] followed by: Theorem Proving like Z3 [#1, #5, #10, #18], Isabelle/HOL [#27] and Yices2 [#30], with 7 articles in total; Software Testing, with approaches like unit testing [#19], fuzzing [#16, #19], mutation testing [#21] and Test Case Generation [#32], with 5 papers; Machine Learning, among the approaches there are Random Forest [#9], Imitation Learning [#16], degree-free graph convolutional neural network [#22], temporal message propagation network [#22] and bidirectional long-short term memory with attention mechanism [#23], totaling 4 papers; Model Checking, leveraging tools like NuSMV [#7, #15] and ProB [#31], with 3 papers in total; Abstract Interpretation with 2 papers [#2, #24]; Runtime Verification [#8] and Manual Auditing [#19], each of them with 1 paper.

Note that it is common for Symbolic Execution approaches to leverage a satisfiability modulo theories solver, from the Theorem Prover category. Another consideration is that not only proposals for approaches were included in Table 4. We also include leveraged methods and tools used with experimental proposals. Hence, the Verification category contains 23 (71.875%) papers, Design has 20

(62.5%) and Implementation 4 (12.5%) papers [#6, #8, #11, #17], all belonging to its only subcategory, the Design Pattern.

5.5 Question 5: Are There Methods and Tools Available for the Identified Solutions?

The material extracted for this question was schematized considering that the tool(s)/method(s): (i) are proposed; (ii) have implementation available (studies that no mention how to access or obtain the implementation, even if it exists, wasn't included); (iii) even if not proposed in the paper, are evaluated in experiments (reproducible cases will be mentioned); (iv) aren't proposed, but have relevant contributions; and (v) the possibility of conflict of interest is considerable (studies with less than half of the authors being part of a for-profit organization wasn't included). Table 5 shows the distribution over this scheme.

Table 5. Distribution over methods and tools available for the identified solutions

ID	#Primary articles	Total (%)
(i)	1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 32	87.50
(ii)	1, 4, 5, 12, 16, 23, 24, 26, 30	28.13
(iii)	1, 2, 5, 8, 9, 10, 12, 13, 14, 16, 17, 18, 19, 21, 22, 23, 24, 28, 30, 32	62.50
(iv)	6, 18, 19, 29	12.50
(v)	7, 9, 12, 19, 29	15.63

As category (i) has a dominant frequency, with 29 papers (87.50%), the category (iv) has the lowest representativeness, with only four papers (12.50%). Even though the proposal for solutions is a category with a larger number, there are considerably fewer papers with available implementations, with only nine articles (28.13%) in that category. About category (iii), it's worth mentioning papers that facilitate the reproduction of experiments/analyses [#1, #5, #12, #16, #18, #23, #24, #28]. We emphasize that we consider only those papers that have made available/indicated a dataset and tools/methods used to allow public access, being potentially reproducible.

Regarding the contributions presented in (iv) we highlight: the importance of Software Engineering in the area of Blockchain and SCs, being mentioned testing, design patterns and best development practices [#6]. Regarding software testing, an important limitation pointed out in SC are the few options for execution environments for testing. In the case of the Ethereum platform there are only the main (the real), test (for developers) and simulation (local) networks [#6]; In [#18], we mention as the main contributions the security analysis framework to classify security issues in on-chain wallet contracts and the reproducible

experiments; and, in [#19], a very interesting flaws categorization, involving 22 categories, is used to outline 246 flaws found in audits. In addition, each of the flaws found is classified according to its severity (potential impact) and difficulty of exploitation. After an in-depth examination of empirical evidence, the authors came to interesting conclusions, such as: unit tests are ineffective in identifying failures, as the correlation between the number of pre-existing unit tests and the audit results was weak; there's a trade-off between cost/degree of automation and flaws detection with the exploitation of high severity and low difficulty; and many failures can be solved by adopting the ERC20 standard.

In relation to category (v), it is clear that identifying cases of conflict of interest having as criterion that at least half of the authors of an article are part of a for-profit organization is not really a good decisive criterion. Identifying cases like these is not a simple task, after all, conflicting interests are not necessarily financial, even though the existence of a conflict of interest in some cases may be evident. On the other hand, how much this affects the content is generally not clear. Then, category (v) indicates only the possibility of conflict of interest and might be worth remembering this detail during reading.

5.6 Discussion

Summarizing what was seen in the studies addressed, we can understand that the observed vulnerabilities have as common cause the difficulty of detecting incompatibilities between the intended and the real behavior. This is largely due to the platform's immaturity, which still has high-level resources with complicated and unexpected low-level behaviors for conventional developers. But the technology is considerably new, and it is expected to mature.

Beyond this scenario, in which there is an increase in attacks exploiting vulnerabilities in SCs, and the existence of already published vulnerable contracts is known, Almakhour *et al.* [@13] mention three reasons to apply formal specification and verification to SCs before their deployment: once an SC is published, any vulnerabilities in its code will become permanent; programmers' lack of knowledge about proper programming semantics to reflect high-level workflow may lead to misconception issues, resulting in "unfair contracts"; and, many programming paradigms used to develop SCs were not designed to be used in the context of the Blockchain environment [8] [@13]. Besides, Liu and Liu [@5] recommended carrying out more research to design more complete formal verification tools, combining formal verification methods with vulnerability analysis methods that complement each other. Observing the results presented in Table 4, Modeling and Specification have been extensively studied. The solutions least explored in the research, that were related to Formal Verification are: Runtime Verification (3.13%), Abstract Interpretation (6.25%), and Model Checking (9.38%). It's an indicative that little was exploited of such techniques to deal with vulnerabilities in SCs, which points us to a direction to be investigated.

A limitation often mentioned in relation to some methods, e.g. symbolic execution and model checking, is path/state explosion [@13, @18], a situation predominantly caused by unbounded iterations. It's important to remember that,

in practice, there are no unbounded iterations because there is a well-defined limit: the gas. Therefore, an adequate modeling of the gas usage by SCs is a promising way to mitigate this limitation of the methods. In this case, simple heuristics can be applied to define values for variables such as the balance of SCs and the cost per instruction.

Regarding to software testing approaches, a considerable part of them have a dynamic nature, requiring the execution of system under test. We known that there is no suitable environment for dynamic analysis. Those that are available range from simulated or real network environments, but each of them with own drawbacks. The resulting papers that mention software testing often wasn't clear if the context was static or dynamic, so we suggest that future work in this topic should address these details.

With respect to reasoning on security vulnerabilities, attacks often are based on exploitation of more than one vulnerability. In this sense there is a certain similarity between the attacks and the propagation [@15]. In addition, it is highly likely that new vulnerabilities will be discovered only after they have already been exploited, and only after that, low-level properties may be defined to mitigation. So, one may consider anticipating new attacks to handle them. One possibility may be to consider the issue of finding new attacks as an optimization problem where the search space are the infinite combinations of vulnerabilities, applying some metrics, like gas cost.

6 Threats to Validity

We clarify some of the possible limitations of this mapping. During the process of selecting studies and extracting data, there was the matter of subjectivity. One of the solutions to avoid subjectivity was that all the authors participated in defining the search string. When the papers were selected to be fully read, the authors conducted a check by reading full papers of the other three authors.

The use of only one electronic database causes a limitation in the set of relevant papers that can be obtained as a result. However, as mentioned previously, Scopus has been selected as it also contains most of the publications from other databases. One other limitation is the lack of snowballing (forward and backward) [6] which could have brought some more relevant publications to complement and to bring a more qualified analysis. Looking directly at the publications of some research groups also is a valid search to enhance our process. These two aspects might have limited the input of more papers. Also, regarding the tertiary analysis provided, a limitation is the fact that we did not use specific strings to find secondary papers, so some relevant articles may have been left out. However, this is the initial study being performed and definitely these aspects will be considered in the continuity of this mapping.

7 Conclusions

A mapping study gives an idea, in the early phases, of shortcomings in existing evidence, which becomes a basis for future investigations [10]. This paper

presented a systematic mapping on vulnerabilities and open issues of Smart Contracts in the Blockchain context. We have analyzed 32 primary references and 18 secondary articles. The findings from this study are expected to contribute to the existing knowledge about the vulnerabilities of SCs. In summary, we concluded that: (i) critical instructions that lead to problems directly related to the correct writing of contracts are: reentrancy, integer overflow, withdrawn and unprotected SELFDESTRUCT instruction; (ii) It was not possible to identify a standard proposal for classification of problems, both in primary and secondary works. Even though the classification is different, it is possible to use them as a basis to design a complete taxonomy regarding the critical issues in SCs; (iii) the papers showed that the proposed solutions to the identified problems are: Design, Implementation, and Verification, the most used methods, Modeling (present in 59.38% of the papers), which belongs to the Design category and Symbolic Execution (present in 31.25% of the papers), which belongs to Verification. Also, some studies point to the need for more research to design formal verification tools, combining formal verification methods with vulnerability analysis.

Thus, as future work, we intend to update this SLM to include more electronic databases and include more stages in the selection process, such as snowballing and research groups. Furthermore, extending the scope of the mapping to more categories related to SCs security problems/issues, as in the categorizations proposed by works like [@16, @17] and traditional software vulnerability analysis [7,15] is a promising path. As future work, we can also suggest: research in the perspective of machine learning on the application of automatic vulnerability detection for SCs, since it is a promising vulnerability detection technique even for traditional software [7,15], does not rely heavily on rules defined by human experts and has obtained competitive results; research proposing solutions involving dynamic analysis or software testing, seeking to deal with the limitations of the execution environment.

Conflicts of Interest. The authors declare that they have no conflicts of interest.

References

1. Alharby, M., Aldweesh, A., van Moorsel, A.: Blockchain-based smart contracts: a systematic mapping study of academic research (2018). In: 2018 International Conference on Cloud Computing, Big Data and Blockchain (ICCBB), pp. 1–6. IEEE (2018)
2. Alt, L., Reitwiessner, C.: SMT-based verification of solidity smart contracts. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11247, pp. 376–388. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03427-6_28
3. Argañaraz, M., Berón, M., Pereira, M.J., Henriques, P.: Detection of vulnerabilities in smart contracts specifications in ethereum platforms. In: 9th Symposium on Languages, Applications and Technologies (SLATE 2020), vol. 83, pp. 1–16. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2020)
4. Buterin, V., et al.: Ethereum: a next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-White-Paper7 (2014)

5. Gelvez, M.: Explaining the DAO exploit for beginners in solidity (2016)
6. Jalali, S., Wohlin, C.: Systematic literature studies: database searches vs. backward snowballing. In: Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 29–38. IEEE (2012)
7. Ji, T., Wu, Y., Wang, C., Zhang, X., Wang, Z.: The coming era of alphahacking? A survey of automatic software vulnerability detection, exploitation and patching techniques. In: 2018 IEEE third international conference on data science in cyberspace (DSC), pp. 53–60. IEEE (2018)
8. Kalra, S., Goel, S., Dhawan, M., Sharma, S.: Zeus: analyzing safety of smart contracts. In: Ndss, pp. 1–12 (2018)
9. Keele, S., et al.: Guidelines for performing systematic literature reviews in software engineering. Tech. rep, Citeseer (2007)
10. Kitchenham, B.A., Budgen, D., Brereton, O.P.: Using mapping studies as the basis for further research-a participant-observer case study. Inf. Softw. Technol. **53**(6), 638–651 (2011)
11. Matsumura, G.d.S., dos Santos, L.B.R., da Conceição, A.F., Vijaykumar, N.L.: Resulting articles: selected primary and related secondary papers. https://bit.ly/3tNkIG6 (2021)
12. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Tech. rep, Manubot (2019)
13. Nikolić, I., Kolluri, A., Sergey, I., Saxena, P., Hobor, A.: Finding the greedy, prodigal, and suicidal contracts at scale. In: Proceedings of the 34th Annual Computer Security Applications Conference, pp. 653–663 (2018)
14. Schueffel, P., Groeneweg, N., Baldegger, R.: The crypto encyclopedia. Growth publisher, Tech. rep. (2019)
15. Shen, Z., Chen, S.: A survey of automatic software vulnerability detection, program repair, and defect prediction techniques. Security and Communication Networks v2020 (2020)
16. Szabo, N.: Formalizing and securing relationships on public networks. First monday (1997)
17. Tolmach, P., Li, Y., Lin, S.W., Liu, Y., Li, Z.: A survey of smart contract formal specification and verification. arXiv preprint arXiv:2008.02712 (2020)
18. Wood, G., et al.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Pap. **151**(2014), 1–32 (2014)
19. Zhang, P., Xiao, F., Luo, X.: A framework and dataset for bugs in ethereum smart contracts. In: 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 139–150. IEEE (2020)